SYSTEMS ENGINEERING
Research Center

Assessing Impact of Distributions and Dependencies in Analysis of Alternatives of System of Systems

A013 - Final Technical Report SERC-2013-TR-035-3

December 19, 2013

Principal Investigator: Dr. Daniel A. DeLaurentis, Purdue University

Co-Principal Investigator Dr. Karen Marais

Team Members

Navindran Davendralingam, Senior Research Associate, Purdue University

Seung Yeob Han, PhD Candidate, Purdue University

Payuna Uday, PhD Candidate, Purdue University

Cesare Guariniello, PhD Candidate, Purdue University

Zhemei Fang, PhD Student, Purdue University

Ankur Mour, Graduate Research Assistant, Purdue University

Contract Number: H98230-08-D-0171

WHS TO0029; RT 44b

Report No. SERC-2013-TR-035-3
December 19, 2013

UNCLASSIFIED

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
19 DEC 2013	**Final**	

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Assessing Impact of Distributions and Dependencies in Analysis of Alternatives of System of Systems, Phase-3	**H98230-08-D-0171**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
DeLaurentis /Dr. Daniel A.	**RT 44-7**
	5e. TASK NUMBER **WHS TO0029**
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Stevens Institute of Technology Purdue University	**SERC-2013-TR-035-3**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
DASD (SE)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release, distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Work under this phase of the project initiative builds upon our earlier efforts in RT-36 and extends the theoretical and practical underpinnings of a suite of methods used to support decision-making in evolving SoS architectures. It is one of three projects active under SERC's Enterprise as Systems and System of Systems (ESoS) SERC thrust area. The research shows promise in evolving towards a real-world deployable toolset that is applicable across multiple SoS domains. The goal is to provide a set of tools and methods within an Analytic Workbench setting that reduces the complexities of decision-making in SoS environments that typically overwhelm the immediate mental faculties of the SoS practitioner. The work in this report demonstrates key concepts and value-added through application of the methods to a collection of SoS pertinent scenarios to illustrate this strength.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**UU**	**100**	

This material is based upon work supported, in whole or in part, by the U.S. Department of Defense through the Systems Engineering Research Center (SERC) under Contract H98230-08-D-0171. SERC is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology

ABSTRACT

The development of a System-of-Systems (SoS) remains a highly challenging endeavor due to the complex interdependencies between systems that often exhibit managerial and operational independence, yet, must work cohesively to achieve an overarching set of capabilities. Current guidelines set forth by the Department of Defense SoS System Engineering guide present SoS SE as a set of seven core elements which are connected to the 16 technical and management processes in the Defense Acquisition Guidebook (DAG). This guide, however, as well subsequent frameworks (such as the Wave Model), are primarily meant to raise awareness of the key issues and products involved. A need exists to create and mature decision support tools to support the decision making process of evolving SoS architectures; including the need for properly assessing the impact that potential disruptions can have, and the analysis of alternatives SoS constructs.

Trades between facets of capability and various measures of risk are essential decisions that must be addressed for SoS capability planning. Existing tools for such trades, where they exist, can be ineffective and non-intuitive when size and/or interdependency complexity is high. These features create a tradeoff space between development risk and capability potential of a system. Prior work under RT-36 centered on seven analytical methods that have been adapted to support SoS architecting decisions and systems engineering of constituent systems. Since no single method or tool can fulfill all technical and managerial needs, the exploration of methods in this report use generic forms of problems faced by practitioners, focusing on inputs, outputs, and limitations in the context of support for the "Wave" model for SoS architectural evolutions.

Work under this RT-44b initiative builds upon our earlier efforts in RT-36 and extends the theoretical and practical underpinnings of a suite of methods used to support decision-making in evolving SoS architectures. It is one of three projects active under SERC's Enterprise as Systems and System of Systems (ESoS) SERC thrust area. The research shows promise in evolving towards a real-world deployable toolset that is applicable across multiple SoS domains. The goal is to provide a set of tools and methods within an Analytic Workbench setting that reduces the complexities of decision-making in SoS environments that typically overwhelm the immediate mental faculties of the SoS practitioner. The work in this report demonstrates key concepts and value-added through application of the methods to a collection of SoS pertinent scenarios to illustrate this strength.

This Page Intentionally Left Blank

TABLE OF CONTENTS

FIGURES AND TABLES

1 SUMMARY

Work in this report has focused on a suite of analytical tools that address core issues faced by the SoS practitioner in making decisions on evolving an SoS architecture. These issues include (1) identification of risks that cascade across the SoS network of interconnected systems, (2) quantification of performance and consequences of various risks at the SoS level, and (3) framework strategies that allow for effective trade-offs to be made across metrics associated with cost, performance and schedule in evolving an SoS. Trades between overarching capability and risk are essential decisions that must be addressed for SoS capability planning. Existing tools for such trades, where they exist, can be ineffective and non-intuitive when size and/or interdependency complexity is high. These features create a tradeoff space between development risk and capability potential of a system. The initial methods being explored in our Analytic Workbench support artifacts of the Wave model; they are listed and described as the following:

- Robustness using Stand-In Redundancy

The method employed here uses robust strategies in evaluating and constructing SoS networks of assets that have the ability to mitigate the effects of constituent systems being compromised.

- Approximate Dynamic Programming

This method employs a multi-stage perspective that extends the portfolio-based framework where decisions in prior steps now affect future decisions in fulfilling requirements and maximizing capabilities.

- Interdependency and Resilience Analysis using Bayesian Networks

This method uses statistical/distribution information from data driven SoS networks to address issues of network capability resilience and analysis of interdependencies in developmental networks.

- Functional (Developmental) Dependency Network Analysis (F(D)DNA)

This method extends the Markov network approach to analysis of both functional and developmental dependencies between constituent systems in an SoS architecture.

- SoS Architecture Decision Analysis using Robust Portfolio Optimization

This method provides a decision support framework for the identification of portfolios of interdependent systems that fulfill requirements and capabilities.

Our work also includes development of mechanism-based design strategies for managing incentivized behaviors within an SoS; this is demonstrated within the context of bandwidth allocation for a tactical data network.

This report is organized as follows: Section (2) of the report is the introduction and covers the background motivation behind the research in supporting SoS architectural evolution from the perspective of a SoS SE practitioner. Section (3) details each analytic method, as listed above, to further support Wave model artifacts in evolving SoS architectures. Subsections of Section (3) reflect progress made for each method individually and demonstration for various application scenarios These scenarios include a prior developed Naval Warfare Scenario (NWS), orbit servicing scenario and technology integration for the National Air Space (NAS) NextGen concept. Section (4) closes with summary notes and our plans for extending this work through subsequent efforts in RT-108.

2 INTRODUCTION

The Department of Defense (DoD) has released a SoS SE guidebook, which presents SoS SE as seven core elements that are mapped to the original 16 technical management processes within the Defense Acquisition Guidebook (DAG) (Defense, Defense Acquisition Guidebook, 2008). The guidebook serves as a concerted effort in understanding how SoS architectures work and address a holistic view of what frameworks are necessary in tackling SoS SE challenges. However, the guidance provided in the document is still in need of more comprehensive guidelines and complementary technical tools to enable effective SoS SE management and support.

2.1 PROBLEM STATEMENT

While the Wave model establishes a concise framework for addressing SoS architectural evolutions, there is nevertheless a pressing need for adequate tools to support each decision epoch, and to navigate the complexities associated with decision making in such environments. Trades between capability and risk are essential during analysis of alternatives for SoS capability planning. Existing tools for such trades, where they exist, can be ineffective and non-intuitive when size or interdependency complexity is high. These features create a tradeoff space between development risks, cost and capability performance of a SoS, for which new tools/methods must be developed.

2.1.1 RESEARCH STRATEGY & RELATION TO SERC CORE COMPETENCIES

The research, conducted under the Enterprise as Systems and System of Systems (ESoS) SERC thrust area, relates directly to the SERC's core competency number 2 with its motivation from the challenges of systems engineering and acquisition when a system-of-systems capability is the primary objective. In these settings, systems must be able to interoperate and therefore have dependencies during the analysis of alternatives and concept development phases. It is well known that interoperability between systems may be necessary for SoS capabilities, but the underlying interdependencies in system development, if not addressed, can be the source of cascading modes of disruption as well. The tools to be derived from the methods research are in concert with several need areas identified in the "Trapeze Model" developed and described in the SoS SE Guidebook. In particular, the "*Understanding Systems*" element that seeks information about systems that impact SoS, both technical and programmatic, is addressed. Also, our approach for uncovering emergent outcomes in networked systems (resulting from interdependencies and topology features) is clearly congruent with the SERC's second Research Thrust in *Systems Science and Complexity* and also is congruent with the "*Assess Performance*" element in the "Wave" construct which seeks to identify potential

undesirable emergent behaviors of the SoS and single system dependencies essential to SoS capabilities.

2.2 Research Objectives

Our research presents a multidisciplinary effort to establish an analytic workbench of computational tools to facilitate better-informed decisions on SoS architectures. The work is motivated by the idea that SoS practitioners have relevant information (data) and archetypal questions that reflect desired outcomes at the SoS level. These archetypal questions in turn are mapped to a modular suite of analytic tools detailed in this report. The analytic workbench (tools and studies) is oriented to help practitioners to answer such questions as: How are vulnerabilities identified and traced to architecture features and interdependence structure? What can be done (e.g., new systems, transitioned architecture) to maximize capability gain (or resilience)? How to understand the alternatives in the context of cascading effects and dynamics? How can investments address current needs but also longer term evolution goals? While not exhaustive, these questions also reflect the domain-agnostic nature of the workbench in addressing SoS level architectural challenges.

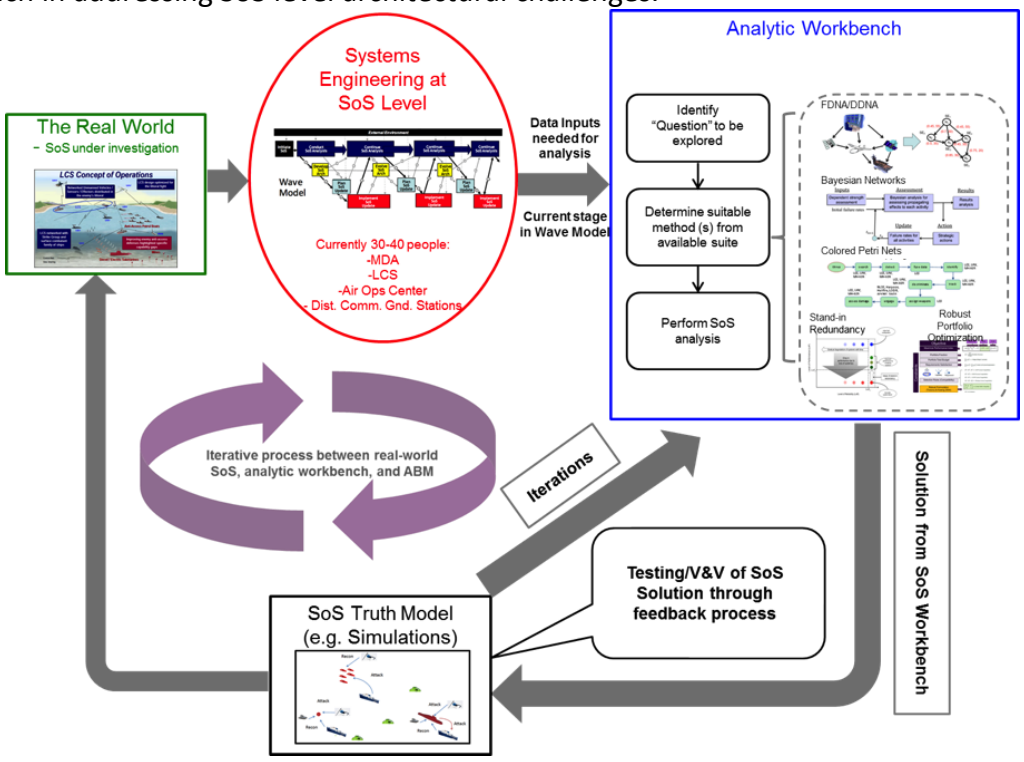


Figure 1: SoS Analytic Workbench concept of use

Figure 1 shows the concept of use of our analytic in support of evolving a SoS. The main idea is that data /information on the current state of systems in an SoS (following stage definitions within the Wave model) can be brought into an *Analytic Workbench*. The suite of tools of the

workbench addresses archetypal analysis on evolving SoS architectures. Figure 2 below maps some of the main archetypical forms of analysis that can be mapped to tools in the current analytic workbench.

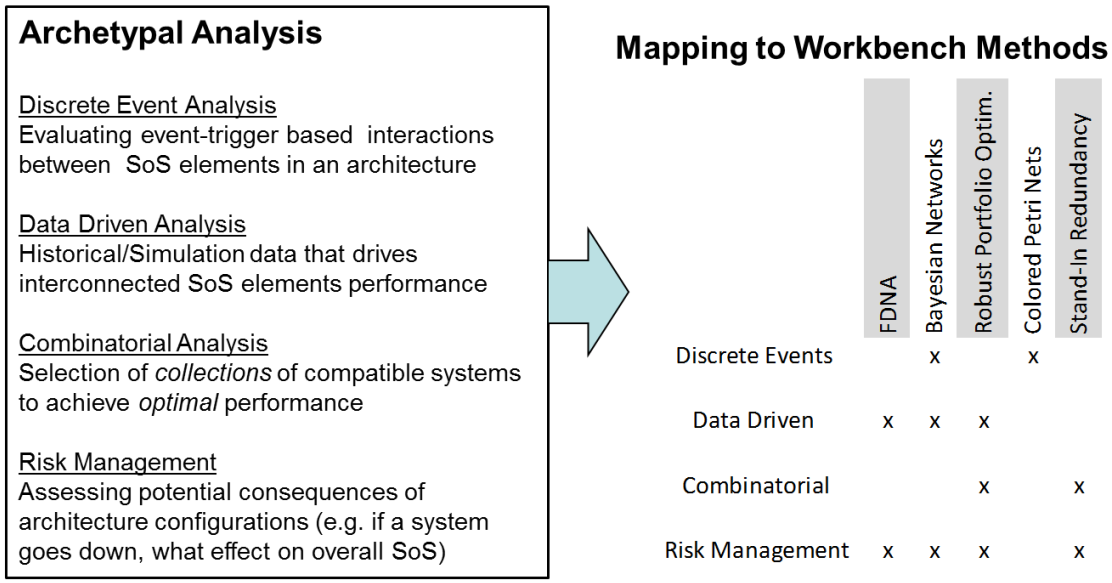


Figure 2: Archetypal mapping to methods

The iterative process of Figure 1 , addressing the archetypal analyses of Figure 2, is executed in concert with available 'truth models' (e.g. computational simulations, field testing) in providing preliminary verification of the next SoS evolution *solution.* The *solution* in this case refers to suggested architectural changes (e.g. addition/removal of systems and/or links) towards fulfilling target SoS capabilities, while preserving acceptable risk (operational or developmental) and cost.

2.1.1 Research Outreach and Engagement

Our efforts under RT-44b have initiated collaborative efforts with three groups; these being the Naval Surface Warfare Center Dahlgren (NSWCDD), USAF Space Command (El Segundo, CA), and MSCI (Always On Demand -US Army). Our initial exchanges with these groups have resulted in valuable insights into the applicability, strengths and potential improvements of the methods for the Analytic Workbench. Our working relationships with these groups continue in our subsequent work under RT-108 towards refining the MPTs under the Analytical Workbench for a larger scale demonstrative deployment.

3 ANALYTIC METHODS

3.1 RESILIENCE-BASED SYSTEM IMPORTANCE MEASURES FOR SYSTEM-OF-SYSTEMS

This research investigates one crucial aspect of SoSs: their ability to recover from disruptions, or their *resilience*. A family of system importance measures (SIMs) that rank the constituent systems based on their impact on the overall SoS performance is developed in this research. Specifically, this set of SIMs guides design and operational decisions by providing specific information on where an SoS is lacking resilience (or has excess resilience) and hence on where improvements are needed (or where downgrades are possible.

3.1.1 INTRODUCTION

All systems are subject to change over their lifetimes. Resilience is the ability of a system to survive and recover from these changes. Implementing resilience is a challenging task because it is highly context-dependent. Systems may be resilient to certain types of disturbances but vulnerable to others. Long-lasting systems, such as infrastructure networks (e.g., energy, transportation, or communications), may initially be resilient to certain disruptions, but as time passes after systems are fielded, changes in the operating environment may make the networks less resilient to both old and new types of threats. Once a failure occurs, resilience is the inherent ability of a system to *survive* and *recover* from this disturbance. And so, resilience is represented as a combination of survivability and recoverability, as shown in Figure 3. This notional representation is widely used in the literature (Tierney and Bruneau, 2007; Castet and Saleh, 2012; Ayyub, 2013) to depict the fundamental ideas behind resilience. While it appears easy to represent resilience conceptually, it is much harder to define, assess, and design resilient systems.

Trade-space analyses are standard practice in systems engineering, but conducting trades on a system-of-systems (SoS) resilience is difficult because, to date, no reliable and consistent metrics have been developed for SoS resilience. Several metrics have been proposed, but these measures assume homogenous networks, ignoring one of the key features of SoS: the combination of heterogeneous systems (e.g., airports and aircraft) to achieve a common goal (e.g., transport). Rather than attempting to create a single metric that glosses over the complexities of an SoS, here a family of System Importance Measures (SIMs) that capture different aspects of SoS resilience is presented. Analogous to component importance measures in reliability theory, the SIMs provide a way to rank or prioritize the constituent systems of an SoS based on different threats. Specifically, these SIMs provide analysts and designers with informative guidance on where an SoS is lacking resilience (or has excess resilience) and hence on where improvements are needed (or where downgrades are possible).

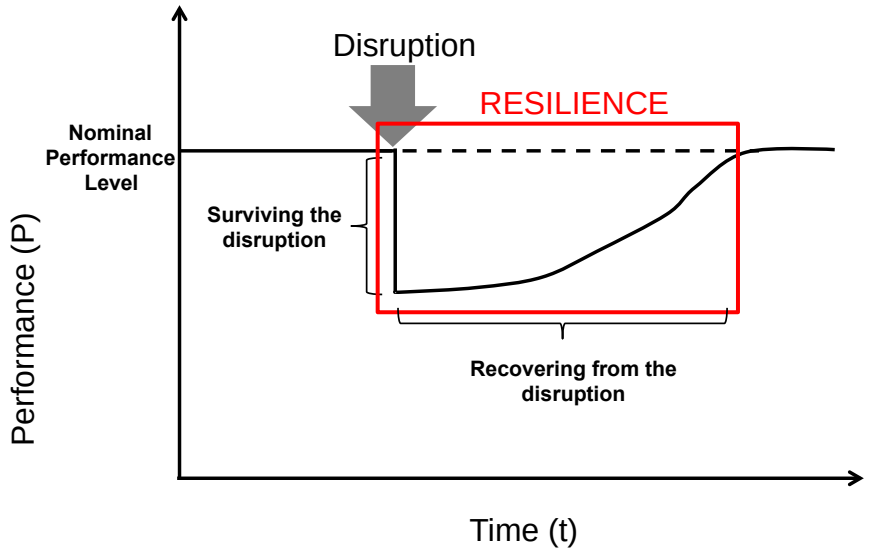


Figure 3: Notional SoS resilience following a disruption

3.1.2 System Importance Measures

Measuring resilience is a critical first step in any framework that aims at addressing or improving resilience. However, establishing a single, all-encompassing resilience metric will be challenging, if not impossible. Since a two-dimensional representation of resilience (see Figure 3) is necessary to capture the main aspects of this attribute, a single metric to measure resilience could be insufficient. Given the two dimensions (time and performance), there will always exist cases where a single-dimensional metric will yield the same result for two different curves. For example, a metric based on area would not distinguish between a curve with a quick recovery to low performance gain and one with a slow recovery to a high performance gain. Further, while a single overall metric may enable overall comparisons between different SoS architectures, a single metric provides little, if any, information regarding specific areas within each SoS that need attention. To address this gap, this research develops a family of System Importance Measures (SIMs) that captures different aspects (time and performance) of and contributors to SoS resilience. Analogous to Component Importance Measures (CIMs) in reliability theory, SIMs provide a way to rank the constituent systems of an SoS based on their impact on the overall SoS performance during disruptions.

Component importance measures (Rausand and Hoyland, 2004; Elsayed, 1996; Ramirez-Marquez and Coit, 2007; Van der Borst and Schoonakker, 2001) combine system structure and component reliability to assess the importance of a particular component to the overall reliability. They indicate, for example, whether improving a particular component will improve the overall reliability, or, conversely, whether a component can be downgraded without significantly impacting the overall system reliability. CIMs include Birnbaum's measure, risk achievement worth, risk reduction worth, and Fussell-Vesely's measure.

There have been some attempts to modify the component importance measures to analyze the resilience of networks. Barker et al. (2013) developed two resilience-based CIMs for networks, but the analysis and subsequent metrics are only applicable to networks with homogenous nodes. In addition, emphasis is placed on network flow (that is, link resilience) rather than to nodes. While this approach may be beneficial in addressing network resilience, it appears to be useful only for networks where the flow between mostly similar nodes is of concern rather than the particular functions carried out at the nodes themselves.

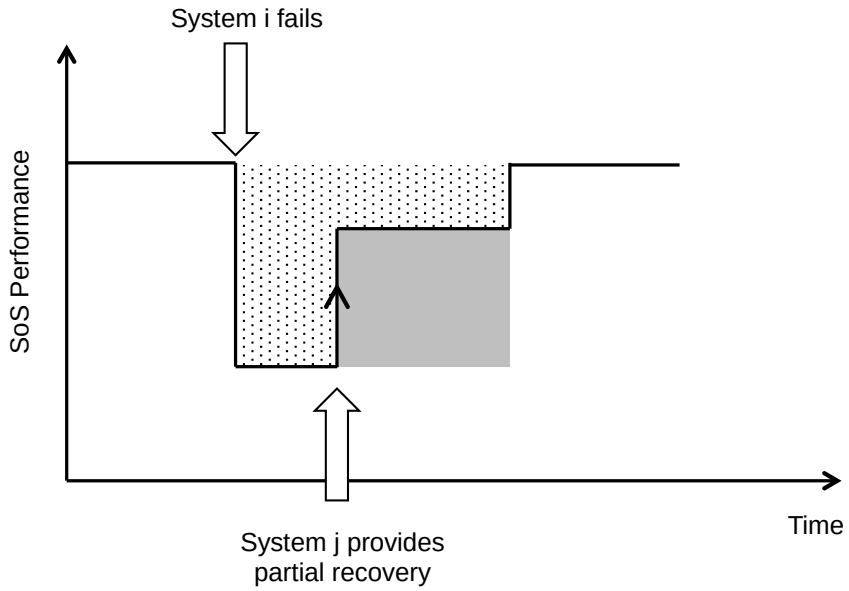


Figure 4: Resilience curve indicating failed System *i* and recovery System *j*

This research focuses on developing importance measures specifically for SoS that are characterized by diversity in nodes and functions. Similar to the CIMs described above, system importance measures help identify and rank the systems that have the most and least impact on the overall SoS resilience. Consider Figure 4. Once a constituent system, say System *i*, fails, the performance of the SoS drops from its nominal performance level to some degraded level. In the absence of any recovery measure, the SoS performance stays at this lower level for the duration of the disruption, till the failed system is repaired or replaced. However, if some recovery measure is employed, such as having another System *j* take over some of the lost functionality, then the SoS performance is raised to a higher level of performance (between the nominal and degraded level) and stays at this level till the original failure been addressed and overall performance is brought back to the nominal level. The two SIMs presented below, System Recoverability Importance (SRI) and System Disruption Importance (SDI) capture both the impact on the SoS of system failures, as well as the importance of using systems to recover SoS performance.

System Recoverability Importance (SRI)

The first measure, System Recoverability Importance (SRI), answers the question: How important is a system to SoS recovery? Thus, $SRI_{i,j}$ measures how important System *j* is to SoS recovery when System *i* fails. Based on Figure 4, we define $SRI_{i,j}$ as:

$$SRI_{i,j} = \frac{grey\ area}{total\ (grey + dotted) area} \tag{1.1}$$

$$E(SRI_{i,j}) = SRI_{i,j}.P_{avail}(j) \tag{1.2}$$

The larger the value of $SRI_{i,j}$, the more important the System *j* is to mitigating any disruption impact on the SoS due to the failure of System *i*. Now, $SRI_{i,j}$ depends on the availability of System *j* to actually provide this recovery. Hence, the expected $SRI_{i,j}$ is calculated using equation (2). The summation of these expected $SRI_{i,j}$ values yields the overall contribution of System *j* to SoS recoverability. Specifically, $\sum_{i=1}^{n} E(SRI_{i,j})$ indicates how important System *j* is to overall SoS recovery when the other systems fail.

System Disruption Importance (SDI)

The second measure, System Disruption Importance (SDI), answers the question: What is the impact of a system failure on the overall SoS? Thus, $SDI_{i,j}$ measures the impact of the failure of System *i*, given the ability of System *j* to provide recovery, on the overall SoS performance. Again using Figure 4, $SDI_{i,j}$ is given

$$SDI_{i,j} = \frac{dotted\ area}{total\ (grey + dotted) area} \tag{1.3}$$

$$E(SDI_{i,j}) = P_{fail}(i|D).SDI_{i,j}.P_{avail}(j) \tag{1.4}$$

Thus, a high value of $SDI_{i,j}$ represents high importance of System *i* since the recovery measure does not adequately reduce the impact of the disruption on the SoS. Now, $SDI_{i,j}$ depends on: (a) the conditional probability that System *i* fails given a disruption D occurs, and (b) the availability of System *j* to actually provide this recovery. Hence, the expected $SDI_{i,j}$ is calculated using equation (4). The summation of these expected $SDI_{i,j}$ values yields the overall expected impact of a disruption on the SoS when System *i* fails. Specifically, $\sum_{j=1}^{n} E(SDI_{i,j})$ indicates the impact of System *i* failure, given that other systems are available for recovery, on the overall SoS.

3.1.3 APPLICATION OF SRI AND SDI: ILLUSTRATIVE EXAMPLE

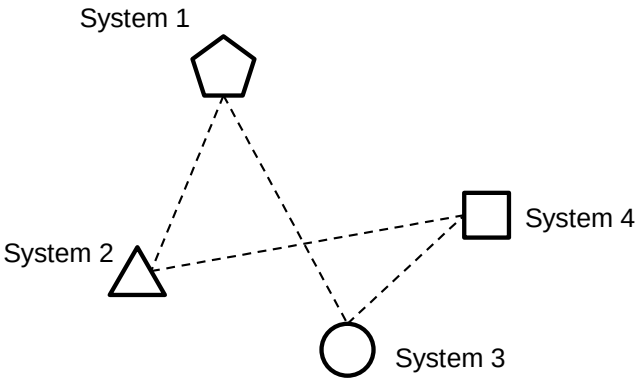


Figure 5: Four-node notional SoS

Consider a simple four-node SoS (see Figure 5). Each constituent system performs one or more functions, and collaborations between these systems enable higher SoS-level capabilities. The failure of each constituent system results in a corresponding drop in SoS performance from 100% to some degraded level. This degraded level depends on the failed system as shown in Table 1. In the absence of any resilience capability, it is assumed that the SoS performance level is raised from the degraded level to 100% only by the repair or replacement of the failed system.

The SRI and SDI measures are now applied to analyze two cases of SoS resilience: (1) stand-by redundancy, and (b) stand-in redundancy. Stand-by redundancy is the traditional technique of having an identical secondary system, called a "back-up", on stand-by for each constituent system. So, if a system fails, the SoS performance level drops to the corresponding degraded level for a small (10% of total disruption time for each system) duration of time before the back-up raises the performance level back to 100%.

Table 1: Impact of system failures on SoS (baseline case with no resilience measures available)

Failed system	Degraded SoS performance level	Time duration of failure
System 1	50%	20 units
System 2	30%	30 units
System 3	10%	10 units
System 4	20%	50 units

On the other hand, stand-in redundancy is a way to compensate for a loss of performance in one constituent system by re-tasking the remaining systems. Specifically, as one entity, or node in an SoS, experiences degraded performance or a failure mode, other entities can alter their operations to compensate for this loss. In the four-node example SoS, stand-in redundancy is implemented as follows:

- When System 1 fails, Systems 2 and 3 can enable partial recovery as follows: System 2 raises the performance level to 75% after 10 time units, while System 3 raises the SoS performance level to 55% after 5 time units;

- When System 2 fails Systems 1 and 4 can enable partial recovery as follows: System 1 raises the performance level to 90% after 5 time units, while System 4 raises the SoS performance level to 40% after 10 time units;
- When System 3 fails, Systems 1 and 2 can enable partial recovery as follows: System 1 raises the performance level to 80% after 2 time units, while System 2 raises the SoS performance level to 65% after 5 time units; and
- When System 4 fails, only System 3 can raise the SoS performance level to 65% after 40 time units

Table 2 shows the probabilities used to compute the expected values of $SRI_{i,j}$ and $SDI_{i,j}$. A key assumption for this example is that each disruption can only affect one system at a time. Thus, if one system fails, other systems that can stand-in for the failed one have relatively high probabilities of availability as they themselves are not impacted by the disruption. The probability that each back-up system (1_b, 2_b, 3_b, and 4_b) is available is 1 (not shown in table).

Table 2: (P_{fail}/D) and P_{avail} for Systems 1, 2, 3, and 4

System	Probability system fails given disruption D occurs (P_{fail}/D)	Probability system is available to stand-in when other systems fail (P_{avail})
System 1	0.6	0.99
System 2	0.5	0.98
System 3	0.02	0.90
System 4	0.01	0.99

Analysis of stand-by redundancy

Table 3 shows the results of applying *SRI* and *SDI* metrics to the stand-by case. The rows of this table indicate the system that has failed and the columns represent the systems that are used for recovery. In the stand-by situation, the failure of each system can only be compensated for by the use of its corresponding back-up. Thus, when System 1 fails, only System 1_b can provide recovery; when System 2 fails only System 2_b can provide recovery; and so on. Wherever applicable, each cell in the matrix comprises a set of values in parenthesis ($E(SRI_{i,j})$, $E(SDI_{i,j})$). These values are calculated using equations (1.2) and (1.4). Summing the expected $SRI_{i,j}$ values along each column, and then normalizing them with the maximum expected recoverability for each column, provides the expected contribution of the back-up systems to overall SoS recoverability. A high value indicates that the system contributes significantly to recovery when other systems fail, and conversely, a low value indicates that the system does not impact overall SoS recovery.

Similarly, summing the expected $SDI_{i,j}$ values and normalizing them along each row gives the overall expected impact of a failure, in the presence of a back-up, on the SoS. Here, a high value indicates that the SoS is impacted severely by the loss of the corresponding system, while a low value indicates that the system failure has a low impact on the SoS. The back-up systems have a high contribution to the overall recoverability, however this contribution is possible only when

the corresponding primary system fails. Also, given the presence of these (costly) back-up systems, the overall impact of failures on the SoS is very low.

Table 3: SRI and SDI for stand-by redundancy

		System used for recovery →				
		System 1$_b$	System 2$_b$	System 3$_b$	System 4$_b$	$\sum_{j=1}^{4} SDI_{i,j} / 1$
Failed system ↓	System 1	(0.9, 0.06)	0	0	0	0.06
	System 2	0	(0.9, 0.05)	0	0	0.05
	System 3	0	0	(0.9, 0.002)	0	0.002
	System 4	0	0	0	(0.9, 0.001)	0.001
	$\sum_{i=1}^{4} SRI_{i,j} / 1$	0.9	0.9	0.9	0.9	

Analysis of stand-in redundancy

Table 4 shows the results of applying *SRI* and *SDI* metrics to the stand-in case. The rows of this table indicate the system that has failed and the columns represent the systems that are used for recovery. As explained previously, in the stand-in situation, the failure of each system can only be compensated for re-tasking other systems in the SoS Here too, wherever applicable, each cell in the matrix comprises a set of values in parenthesis (E($SRI_{i,j}$), E($SDI_{i,j}$)), calculated using equations (1.2) and (1.4). Summing and normalizing the expected $SRI_{i,j}$ values along each column provides the expected contribution of each system to overall SoS recoverability. System 1 plays a key role in recoverability as it can provide substantial recovery when Systems 2 or 3 fail. On the other hand, Systems 3 and 4 are not useful to recovery. Although System 3 can stand-in partially when Systems 1 or 4 fail, the actual amount of recovery it provides is very low.

Summing and normalizing the expected $SDI_{i,j}$ values along each row provides the overall expected impact of a failure, in the presence of a back-up, on the SoS. Failure of System 1 has a relatively large impact on the SoS, while failure of Systems 3 and 4 do not impact the SoS significantly. Unlike the expected $SRI_{i,j}$ values, the expected $SDI_{i,j}$ depends on the probability that a system will actually fail. From Table 2 it is seen that the probability of failure for Systems 3 and 4 are low. As a result, even though the recovery measures in place for when these two systems fail are inadequate, the impact of their failures on the overall SoS is low.

Table 4: SRI and SDI for stand-in redundancy

System used for recovery →

Failed system	System 1	System 2	System 3	System 4	$\frac{\sum_{j=1}^{4} SDI_{i,j}}{3}$
System 1	(0,0)	(0.2450, 0.4410)	(0.0675, 0.4995)	(0,0)	0.3135
System 2	(0.7071, 0.1414)	(0,0)	(0,0)	(0.0943, 0.4479)	0.1964
System 3	(0.6160, 0.0075)	(0.2994, 0.0136)	(0,0)	(0,0)	0.0070
System 4	(0,0)	(0,0)	(0.1013, 0.0080)	(0,0)	0.0026
$\frac{\sum_{i=1}^{4} SRI_{i,j}}{3}$	0.4410	0.1814	0.0562	0.0314	

These initial results demonstrate the use of SIMs in the analysis and design of resilient SoSs:

1. Using these importance measures, areas of the SoS have excess or inadequate resilience can be determined. While an overall metric could provide some estimate of SoS resilience under both stand-in and stand-by redundancy cases, these SIMs provide specific information about: (a) systems that have *excess recoverability*, and (b) systems that have *inadequate recoverability* and hence, need more attention (resources).
2. SIMs also provide specific information to guide design decisions. For example, the results showed which type of redundancy proved better for each system. Specifically considering the expected $SDI_{i,j}$ values for stand-by and stand-in cases, the failure of System 3 or System 4 has marginally higher impact on the SoS when stand-in redundancy is employed instead of stand-by redundancy. If these systems are expensive to back-up, then incurring a slightly higher initial investment in enabling other systems to perform some of System 3 and 4's functions may be a more cost-effective option to achieve essentially the same level of resilience. This observation highlights the importance of cost implications in resilience analyses. Future work will incorporate financial considerations with these SIMs to guide design decisions.

3.1.4 Conclusion and Future Work

The primary aim of this research is to provide a rigorous quantitative basis to make informed decisions about SoS resilience as opposed to the existing ad-hoc approaches. System Importance Measures (SIMs) are suggested as one way to analyze resilience with a focus on ranking resilience-critical systems. The mathematical formulation behind these SIMs was first presented, and then their use was demonstrated with two cases.

In current work, two additional SIMs are being developed: System Recovery Time Importance (SRTI) and System Performance Importance (SPI). The importance measures presented in this document, that is, SRI and SDI, do not place a relative value on time versus performance. For example, the resilience curves for a system that provides a rapid, but low

recovery, and a system that provides a slower, but greater recovery can have the same SRI and SDI values. The additional importance measures will explicitly account for how fast a system can provide recovery as well as how much performance gain can be obtained. Also in current work, the importance measures are being applied to a study of the Littoral Combat Ship SoS.

In future work, the SIM initial formulation presented here will be refined and expanded to a framework that evaluates resilience-cost trade-offs and provides guidance on designing SoS resilience. Specifically, this research will help identify areas in the SoS where greater investment of resources will considerably improve the resilience of the overall SoS, or conversely, areas where additional capital need not be spent, as these systems do not significantly impact the overall SoS.

The key contribution of this research is to provide decision-makers with improved information and tools to make SoS-level decisions. The use of system importance measures (SIMs), and their resulting upstream effects on development policies, costs and risks, can be used by decision-makers to quantitatively assess the resilience of SoSs and by designers to better allocate risk resolution resources.

3.2 Dynamic Planning of System of Systems Architecture Evolution

The dynamic planning and development of a large collection of systems or a 'System of Systems' (SoS) poses significant programmatic challenges due to the complex interactions that exist between constituent systems. Decisions to add, remove, or reconstitute connections between systems can result in repercussive failures across operational and developmental dimensions of an SoS. Research in this section develops a tool that adopts an operations research based perspective to SoS level planning based on metrics of cost, performance, schedule and risk. Specifically, our work employs an Approximate Dynamic Programming approach that is well suited to address issues of computational tractability of the resulting dynamic planning optimization problem. This approach allows for identification of near-optimal multi-stage decisions in evolving SoS architectures. A Naval Warfare Scenario SoS example problem illustrates application of the method.

3.2.1 Introduction

The US Department of Defense (DoD) has recognized the importance of a 'System of Systems' view to the acquisition and development of military assets (OUSD(AT&L), 2008); this recognition means that SoS capabilities being sought are a direct consequence of the interactive effects of their constituent systems. These constituent systems are operationally and managerially independent, yet interact on various levels to give rise to an overarching SoS level capability. Decisions to support the development of these monolithic entities require acquisitions using systems engineering based policies that can better account for the complexities associated with SoS architectures. To this end, Defense Acquisition Guidebook

(DAG) (DoD, 2008) has been developed to aid the understanding and implementation of DoD acquisition practices including evolutionary acquisition strategies that are the norm for SoS capability evolution. Consistent with DAG, the Systems Engineering Guide for Systems of Systems (SoS SE) examines the SoS challenges and provides a 'Trapeze' model to give a good conceptual view of the SoS SE core elements, their interrelationships and SoS decision-making artifacts (OUSD(AT&L), 2008). Dahmann, et al. (2010) unwind the trapeze model to a more familiar and intuitive time-sequenced "Wave" model and identifies information critical to decision making in SoS evolution. Figure 6 illustrates the Wave model and the original SoS SE core elements.

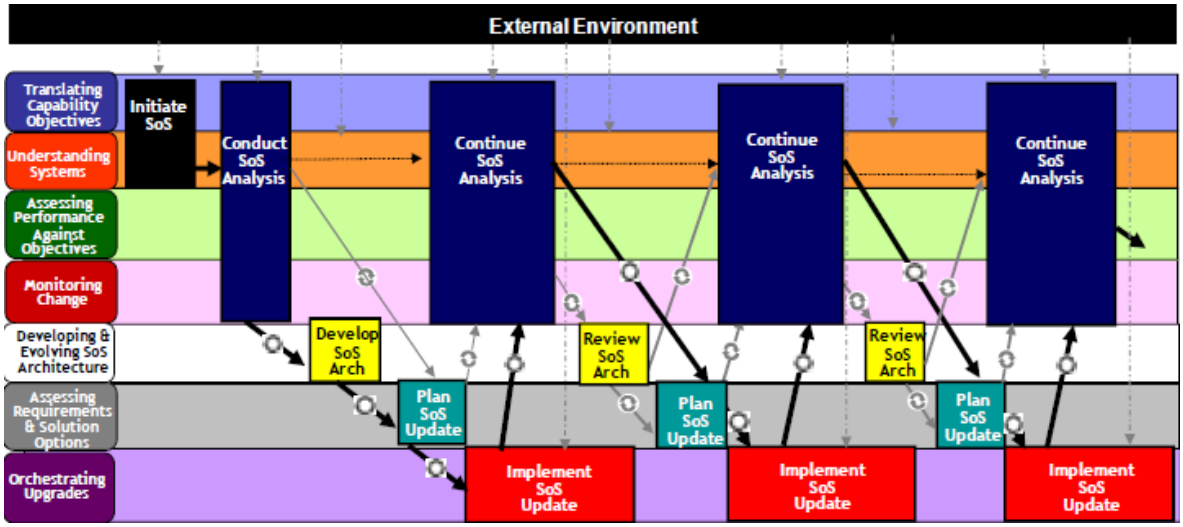


Figure 6: Wave Model and Related SoS SE Core Elements (J. Dahmann, et al. 2010)

The systematic procedures as shown in Figure 6 provide guidance to SoS practitioners to make decisions properly; in addition to the logical procedures, an analytic solution framework to objectively quantify the state and outcome of consequent actions to evolve an SoS architecture is required by SoS practitioners. Key decisions points within the Wave model are reflected by the , 'Plan SoS Update' and 'Implement SoS Update' artifacts as shown in Figure 6. Accordingly, actions may involve a sequence of decisions that include adding new systems, retiring old systems, or upgrading systems, for 'Plan SoS Update' that could provide policy makers decision sets for achieving optimal or near-optimal SoS capability over a time period. Operational decisions for 'Implement SoS Update' might be integrated in the meantime to provide prompt feedbacks to developers. Unlike traditional production, investment, or supply chain planning problems, that are typically faced at system level, the dynamic planning in an SoS exhibits a multitude of distinguishing features that must be carefully addressed. Typical questions could include:

- how to deal with the interactions between decisions from multiple independent organizations,
- how to deal with the diverse time scales occurring in an SoS (such as investment decisions every five year versus operational deployment every few months),

- how to deal with the complexity resulting from the sheer number of uncertain variables involved

The sequential development process and the objective of maximizing the overall capability for a given finite time period makes dynamic programming a natural choice to address the problem. However, the characteristics of an SoS such as the large number of systems that may be involved, multi scale decisions and significant uncertainties lead to state, decision and sample explosion respectively, make dynamic programming challenging. *Approximate Dynamic Programming (ADP)* is an umbrella term covering various methods and techniques, aiming to solve curses of dimensionality by approximating the future value functions. ADP allows us to formulate the SoS architecture evolution process into a dynamic planning problem and address the complexity resulting from multi timescale decisions and uncertainties.

3.2.2 Literature Review

Researchers have come up with several different frameworks and methods to tackle architecting and evolution of systems and SoS. Ross and Donna (2008) proposes an Epoch-Era Analysis for system design comparison and selection to deliver sustained value to stakeholders in the face of a rapidly changing world. Epoch-Era Analysis uses natural value-centric time scales, where an Epoch refers to a period with a fixed context, characterized by static constraints, available design concepts, available technology, and articulated attributes while an Era is generated by stringing together sequences of epochs given the likelihood of switching between given epochs and the durations of each epoch. This work has been extended to the valuation of changeability under the framework (Fitzgerald and Ross, 2012). The current application of changeability focuses on complex system design using qualitative indicators while the applicability to SoS has not been fully addressed. Overall, Epoch-Era Analysis provides a conceptual framework that needs to be combined with other analytical techniques such as options theory to guide the evolution.

Real Options Analysis (ROA) is an approach that values flexibility when certain physical decision options are embedded to cope with future uncertainties (Neufville, 2003; Chaize, 2003; Mikaelian, 2009). Simply put, a real option gives the right (but not the obligation) to undertake some business initiatives that include abandoning, deferring, or expanding projects. De Weck, O., Neufville, R., Chaize, (2004) employ ROA to investigate the benefits of the staged deployment of communications satellite constellations in low earth orbit under demand uncertainty, which shows the usefulness of ROA for initial architecture selection embedded future options. However, the discrete number of decision tree options grows significantly with the number of available options involved. For an SoS, this number is a combinatorial artifact of the number of current and yet-to-be introduced candidate systems involved and thus is prone to the curse of dimensionality.

Portfolio theory has been applied to decision-making in an SoS environment by treating the collection of existing and potential future systems as a portfolio of “asset” systems, which combine to deliver a desired SoS level goal. The most used form of portfolio theory involves the

application of mathematical programming methods in identifying optimal collections of investment assets that balance reward against risk, given an investor's specific tolerance for risk. Prior work in SoS architectural analysis has used robust portfolio optimization techniques in addressing the acquisition of constituent systems in an SoS (Davendralingam, 2011). Other work employs multiple objective value analysis, mathematical optimization, cost benefit analysis, mean-variance approach and so forth to support portfolio decision analysis (Burk and Parnell, 2011). However, these classes of portfolio methods do not directly translate to multi-stage portfolio problems as readily for SoS architecture evolution.

Dynamic programming was proposed as one of the most promising methods to formulate the SoS management problem by Maier (2005). However, due to the large number of systems and inherent uncertainties involved in an SoS, dynamic programming suffers from computational complexity. *0Approximate dynamic programming* presents a powerful modeling and algorithmic strategy that can address a wide range of optimization problems that involve making decisions sequentially in the presence of different types of uncertainties (Simao, Day 2008; Powell et al., 2011). It employs a variety of approximation techniques in addressing issues of computational tractability due to the curse of dimensionality. Bertsekas and Tsitsikilis (1996) applied neural network concepts (also termed Neuro-Dynamic Programming) to approximate the value function and named Neuro-Dynamic Programming (NDPPowell (2010) developed approximation strategies based on post-decision state variables which avoid computing the expectation of uncertainties. Wide applications of ADP exist in literature on dynamic resource allocation problems. Powell et al have applied ADP techniques to real world problem that include military airlift operations under uncertainty (Powell, Ayari, et al., 2010), fleet management for locomotives, business jets, etc (Powell and Topaloglu, 2002), R&D portfolio optimization for solid oxide fuel cells problem (Hannah, Powell and etc, 2010), Other research work such as modeling global climate policy under decision-dependent uncertainty (Webster et al., 2011), and multi-stage investment management (Keles and Hartman, 2007), has also been investigated using ADP. The prior cited applications of ADP have shown its potential in addressing issues of computational tractability and sequential decision-making under conditions of uncertainty. The decision variables involved reflect the same kind of resource allocation decisions that are exercised in SoS architectural decision making; however, none of them have taken into account the SoS characteristics such as individual systems making their own decisions and complex interactions among myriad heterogeneous systems.

3.2.3 Technical Approach

We formulate the process of SoS architecture evolution shown in Figure 6 as a dynamic programming model. Figure 7 demonstrates a hierarchical framework with multiple time-scales to solve the SoS architecture evolution as a dynamic planning problem. The objective is to maximize the overall SoS capabilities subject to a set of resource constraints (e.g., budget and manpower), over a finite time horizon, under uncertainty. The sequential decision variables are in the form of addition, removal and continuation of systems. The approach translates the

hierarchical and coupled nature of interconnected systems within an SoS into the language of mathematical programming that equivalently describes characteristics of the problem within the context of an optimization problem. Once we formulate the problem into a dynamic programming model, various approximation strategies including aggregation, parametric model and non-parametric model can be applied to the resulting multi-stage problem to address complexity and computational tractability.

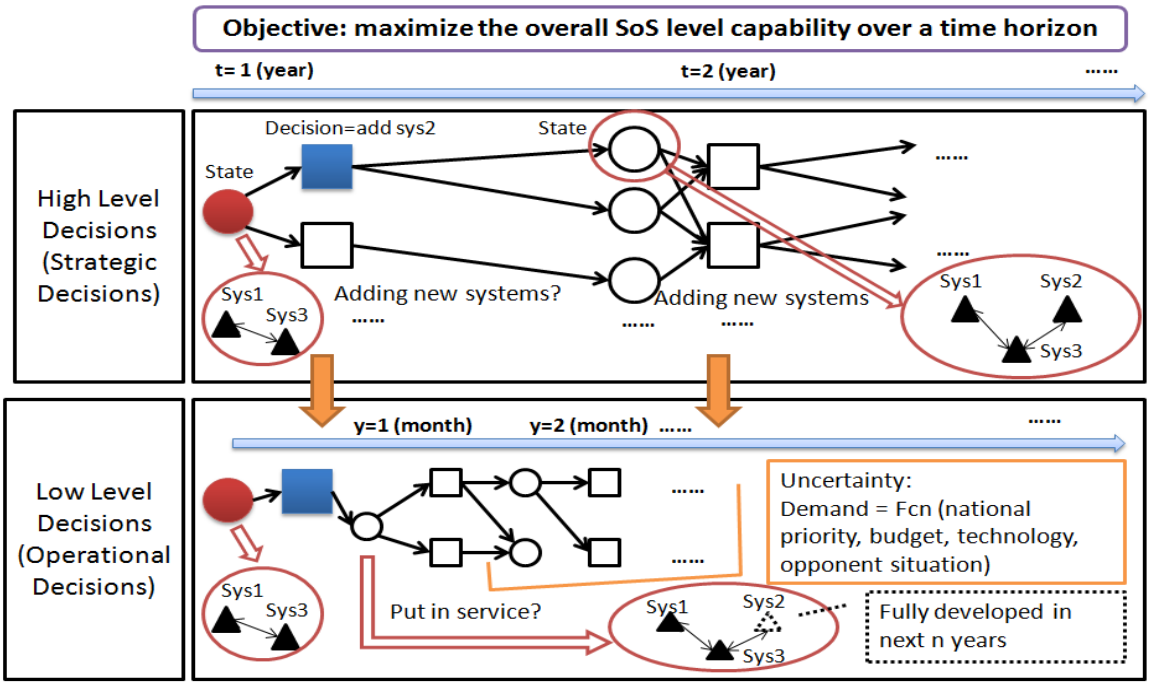


Figure 7: Overall Framework for the Sequential Decisions

As illustrated in Figure 7, the overarching objective is to maximize a given SoS level capability index over a time horizon. The capability index can be translated into the number of threats being engaged, the number of surviving aircraft or ships after being attacked, etc. High level strategic decisions address long time scale decisions such as investment or acquisition of new aircrafts, ships, satellites, which all participate in a particular SoS. Multiple years' effort is needed to obtain the final products after current decisions. Sequential decisions are preferable as they offer the chance of evaluating and learning the current state such as technology maturity, budget situation and so forth from the environment. Low level decisions can either represent operational decisions or decisions within constituent systems like scheduling. In this work, operational decisions act as low level decisions with short time scales. For instance, during a military deployment, decision makers have to sequentially decide when to put systems in service, when to put systems in maintenance and when to keep systems idle. The high level strategic decisions provide a resource pool for the low-level operational decisions; new invested systems can become available after a few years, by which time operational decision-makers will have additional choices and more advanced systems at their disposal. Low level short time scale decisions are required for two reasons: one is to satisfy short time

requirements; the other, is to provide more accurate feedback and new information to high level decision-makers to facilitate learning. Large errors occur if new information is counted and learnt after a few years without considering effect of short time scale decisions. Moreover, a variety of sources of uncertainties or exogenous information exist in SoS, such as national priority, technology, budget, market, climate change, all of which have completely different timescales. For example, national priorities may have major changes every five years while the budget situation might change every year and market demand might vary every week.

Therefore, this framework aims to provide policy makers a sequence of architecture alternatives at different stages and time scales (time interval between decisions), given the individual system capabilities and resource constraints. The incorporation of multiple timescales leads to a rapid increase in the number of decision variables involved in an SoS. If we consider a 10 year long SoS program as an example, annual strategic decisions and monthly operational decisions will give us 120 decision variables. Besides, the large number of systems included in an SoS along with the heterogeneity of systems, generates a large number of architecture alternatives, not to mention the effect of complex uncertainties. Hence, all these pose significant challenges to computational efficiency. Our approximate dynamic programming framework provides a path to address such type of issues through approximation of the value functions and many other techniques, as depicted in the next section.

3.2.4 Approximate Dynamic Programming

The logic of ADP is to step forward through time instead of working backwards as classical dynamic programming does. The forward stepping requires an appropriate approximation of value function $\bar{V}_t(S_t)$, which represents the expected future value of being in state S_t; this replacement is the essence of approximate dynamic programming. In the absence of future information, a sequence of sample realizations of random exogenous information must be generated, which is usually obtained in three ways: real world data, computer simulations, and sampling from a known distribution. Following the sample path, the value function approximation can be iterated and updated. Thus the computational cost only stems from the production of iteration number and stage number, which gives great efficiency when the problem grows large, although the efficiency needs to sacrifice some accuracy in the results.

In this report, two ADP strategies are employed: linear value function approximation as a special form of parametric model and the concept of post-decision state variables, as used in literature (Powell, 2010). Classical dynamic programming recursively computes the Bellman equation which is the essence of dynamic programming as following:

$$V_t(S_t) = \max_{x_t \in X_t} \left(C_t(S_t, x_t) + \gamma E\{V_{t+1}(S_{t+1}) \mid S_t\}\right) \quad (2.1)$$

where S_t represents the state variables, x_t represents the decision variables, C_t is the current contribution, γ is the discount factor and $V_{t+1}(S_{t+1})$ is the expected value of being in state S_{t+1}.

To avoid looping over all possible states, the approximated value $\bar{V}_{t+1}$ is used instead of the exact value function. A generic structure for the value function approximations is:

$$\bar{V}_t(S_t) = \sum_{f \in F} \theta_{tf} \varphi_f(S_t) \tag{2.2}$$

where $\{\varphi_f(S_t): f \in F\}$ are often referred to as features that capture the important characteristics of the resource state vector from the perspective of capturing the total expected contribution in the future. θ_{tf} represents adjusting parameters that allow us to obtain different value function approximations. A variety of methods such as linear and piece-wise linear value function approximations are used to calculate θ_{tf} and $\varphi_f(S_t)$.

To avoid the calculation of expectation, post-decision state variables are introduced, by which Bellman equation can be written as:

$$V_{t-1}^{x}(S_{t-1}^{x}) = E\{\max_{x_t \in X_t} C_t(S_t, x_t) + \gamma \bar{V}_t^{x}(S^{M,x}(S_t, x_t)) \mid S_{t-1}^{x}\} \tag{2.3}$$

where S_{t-1}^{x} represents post-decision state vector. In this equation, the expectation can be dropped by using a sample realization of the uncertainties $W_t(w)$; then the equation becomes:

$$\tilde{V}_{t-1}^{x}(S_{t-1}^{x}) = \max_{x_t \in X_t} C_t(S_t, x_t) + \gamma \bar{V}_t^{x}(S^{M,x}(S_t, W_t(w), x_t)) \mid S_{t-1}^{x} \tag{2.4}$$

Given a particular realization of $W_t(w)$, the above equation becomes a deterministic optimization problem, which solves the *curse of dimensionality*.

3.2.5 Illustrative Example and Results

The Littoral Combat Ship (LCS) is a relatively new naval system that, together with its related components, may be viewed as a naval warfare SoS. LCSs are outfitted with three different mission packages: surface warfare (SUW), anti-submarine warfare (ASW), and mine warfare (MIW), aimed at countering mines, small boats and submarines in littoral waters (Ronald, 2013). The SUW module that is designed to detect and engage multiple surface contacts in a littoral environment consists of LCS, UAV, and MH-60R. We demonstrate the applicability of our approach by applying it to a simplified SUW module. A capability index can be converted from percentage of systems surviving an attack and for the sake of simplicity, notional numbers based on expert judgments are used. In this context, we assume that at strategic level, this is a three-year program and a system can be completely developed in one year. Decision makers wish to be aware of which architecture from different combinations of LCS, UAV and MH-60R should be developed at the beginning of each year. Once systems are available to enter the operational domain, we assume the deployment of these systems are seasonal and decision makers wish to know whether to put existing systems in service or out of service (like maintenance) to prevent uncertain attacks. The feedback from the low level decisions can influence the strategic decisions for next year.

A pre-requisite to formulating the problem into a mathematical programming format is to identify the basic elements of dynamic programming: state variables, decision variables,

transition function, exogenous information (uncertainty) and objective function. The objective is to maximize the expected sum of SoS capability at operational level after being attacked during each stage, with the constraints of budget on the investment of new systems at the beginning of each year. It is assumed that the capability of each system is additive towards obtaining SoS level performance. Low level decisions are assumed to be binary. Accordingly, the formulation can be written as:

$$\begin{aligned} &obj: \max \quad E(\sum_{t=1}^{3}\sum_{s=1}^{4} c^p (R_{t,s}^{in} + y_{t,s})(1-\hat{w}_{t,s+1}) \\ &st: c^s x_t \le B_t \quad (t=1..3) \quad \text{and} \quad y_{t,s} \le R_{t,s}^{out} \quad (t=1..3,\ s=1..4) \end{aligned} \tag{2.5}$$

where c^p represents a row vector of notional index of individual system (LCS, UAV, MH-60R) capability and is assumed to be constant over time. $\hat{w}_{t,s+1}$ as uncertainty coming from time s to $s+1$, refers to a column vector of binary results from the production of probability of threat occurring and probability of successful attack. c^s denotes development cost of each system while B_t means budget limits at each year. The state variable R_{ts} represents a vector of numbers of systems with different attributes at season s of year t, specifically, R_{ts} consists of R_{ts}^{new}, R_{ts}^{in}, and R_{ts}^{out} where R_{ts}^{new} is for new systems under development, R_{ts}^{in} is for systems in service while R_{ts}^{out} is for systems out of service. The decision variable x_t reflects developing systems at the beginning of each year (for the ease of representation, it means the new available systems that are developed one year ago) while $y_{t,s}$ means putting available systems in service. Transition functions can be formulated as follows:

$$\begin{aligned} &new\ systems\ available: R_{t,s=1}^{out} = R_{t-1,s=4}^{out} + x_{t,s=1} \\ &put\ in\ service: \ R_{t,s+1}^{in} = (R_{t,s}^{in} + y_{t,s})(1-\hat{w}_{t,s+1});\ \ R_{t,s+1}^{out} = R_{t,s}^{out} - y_{t,s} \end{aligned} \tag{2.6}$$

We assume that the initial number of available systems in the architecture is zero, thus the resource pool for low level decisions within one year results directly from initial strategic level decisions. Thus the problem can be further simplified as:

$$\begin{aligned} &obj: \max \quad E(\sum_{s=1}^{12} c^p (R_s^{in} + y_s)(1-\hat{w}_{s+1})) \\ &st: c^s \sum_{s=4(t-1)+1}^{4(t-1)+4} y_s \le B_t \quad (t=1,2,3) \ \ where\ x_t = \sum_{s=4(t-1)+1}^{4(t-1)+4} y_s \quad (t=1,2,3) \end{aligned} \tag{2.7}$$

To validate the results from ADP, a regular binary integer programming is employed to obtain the optimal solution for comparison. To easily compare the results from ADP and optimal solution, incoming attacks are assumed to be deterministic and known as a prior. Under this experiment setting, results can be obtained as displayed in Figure 8 and Figure 9. Note that the primary reason for employing ADP relies on its potentials of computational scalability to solve problems with large number of states and uncertainties involved where optimal values are usually difficult to compute. Since this example is not a large problem, the advantages of ADP are not obviously demonstrated. Results in Figure 8 and Figure 9 primarily

aim to illustrate that reasonable objective can be obtained by ADP in small examples and it is validated to further apply to large problems.

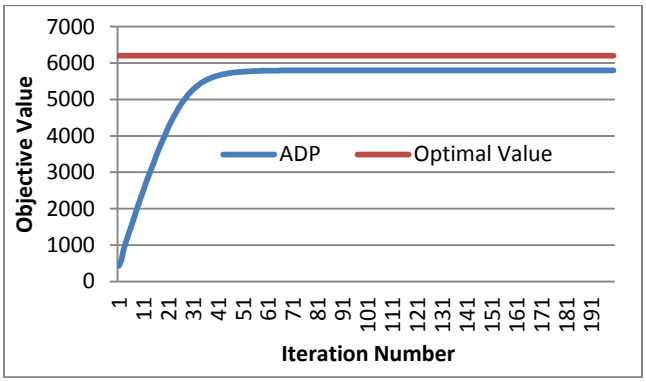


Figure 8: ADP Objective Value and Optimal Objective Value

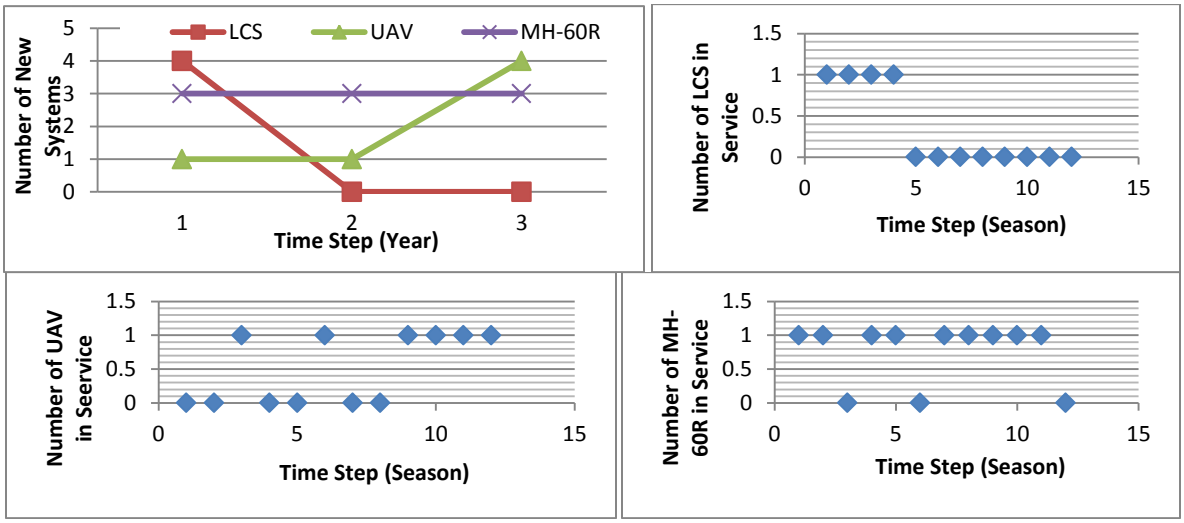


Figure 9: Strategic Level Decisions and Operational Level Decisions

Figure 8 shows the comparison between objective value obtained by ADP and optimal objective value. Performance stabilizes after around 50 training iterations of value function approximations. The ADP objective value is 6% lower than the optimal value and the acceptance of this sub-optimal result is dependent on the decision makers' preference of computational efficiency and accuracy of the results. The computational cost of ADP linearly scales with the product of iteration number and time stages while regular linear or integer programming exponentially scales when system states and uncertainty increase. Overall ADP provides suboptimal decision set with a reasonable range; when systems involved becomes large, time scale grows and uncertainties are present, ADP exhibits a clear advantage in computational efficiency. As shown in Figure 9, sequential decisions of architecture alternatives at strategic and operation level can be provided for decision makers and it gives policy makers a global sense of decisions and resulting impacts towards the SoS.

3.2.6 Conclusion and Future Work

This research work addresses decision-making in the dynamic planning of an SoS architecture using approximate dynamic programming techniques. A notional Naval Warfare SoS is used to illustrate the applicability of the method. Deterministic assumptions are made to compare the solution of the ADP algorithm to the optimal solution. The results indicate a sequence of architecture alternatives over time for SoS practitioners while a small amount of difference exists between the objective value from ADP algorithm and optimal objective value. Future work will explore application of algorithmic advances in the ADP formulation so as to enable efficient incorporation of forward state information, learning potential and quantification of uncertainties related to the value in being in particular states (architectures). Additionally, future work will explore computational efficiency using a more realistic, large scale SoS concept problem.

3.3 An Approach for Evaluating System-of-Systems Operational Benefits of a New Decision using Interdependency Analysis

There are two ways to develop a new SoS: 1) integrate only existing off-the-shelf systems or 2) deploy nascent and inchoate systems with existing ones. In many cases, the latter is selected for the sake of the advancement of SoS wide capability. For example, the current air transportation system has adopted new technologies such as Automatic Dependent Surveillance-Broadcast (ADS-B) to improve the performance of the current system.

SoS system engineers desire to maximize the capability of a new SoS, but this desire conflicts with the realities of constrained environment such as tight budgets and restricted schedules. Before implementing the proposed SoS capability, SoS system engineers must understand the relationships between operational impact and resource allocation. With continuing budget cuts, SoS system engineers have had a hard time making program choices. Sometimes, they should reallocate resources and make a new decision to ensure that it meets both budget constraint and operational needs of stakeholders. SoS system engineers should ask this question when designing a new SoS: which alternative should they choose to ensure that they achieve the highest level of operational effectiveness given their limited resources?

To answer this question we propose an approach which facilitates early decisions and planning. In this section, we use the next generation air transportation system (NextGen) to illustrate the proposed approach. The proposed approach helps SoS system engineers to select the alternatives to maximize the performance of the NextGen given their limited resources.

3.3.1 Introduction to the Next Generation Air Transportation System (NextGen)

Continuously rising demand in the National Airspace System (NAS) is one of the major contributing factors that aggravate system-wide congestion and delays. Therefore, mitigating congestion at the NAS has become one of several priorities for the Federal Aviation

Administration (FAA). To solve this, the FAA proposed new operating concepts and technologies to achieve the Next Generation air transportation system (NextGen).

The NAS is a complex system characterized by a large number of system components interacting with one another such as airlines, passengers, airplanes, airports, organizations, and air traffic control systems. In addition to many diverse components in the NAS, uncertainty factors such as oil price change and the cyclic economic conditions make the NAS even more complex. Due to this complexity on the NAS it is a challenging endeavor to analyze and assess the impact of the introduction of new technologies or polices on the NextGen.

Much research has been done on validating the efficiency and safety of integrating NexGen technologies into the NAS. Long, et al. (2011) proposed a new concept (System-Oriented Runway Management (SORM)) to enhance airport performance (e.g., runway capacity, throughput, and flight time and fuel savings). SORM uses two distinct technologies: runway configuration management and combined arrival/departure runway scheduling. Hemm, et al. (2012) proposed automated NextGen concepts (called ground-automation controlled concept) to improve the separation assurance safety risk. These researches only focus on one segment (e.g., surface flow) within the flight profile. Thus, they could not capture the impact of new technologies on the overall NAS. Hasan, et al. (2012) proposed new air traffic management concepts and technologies to support a near-term vision of trajectory based operations which is one of the NextGen capabilities. They validated the efficiency and safety of five new technologies into the NAS: Direct-To algorithm (D2), Arrival Manager (AMAN), Multi-Climb, Corridor Integrated Weather System (CIWS), and Convective Weather Avoidance Model (CWAM). The results proved that the new concepts can save fuel burns and flying time by improving airspace efficiency. Even though this study considered all segments in the flight profile (e.g., surface and en-route flows) to validate efficiency of five new technologies on the NAS, they validated the efficiency of an individual technology one by one when assessing the benefits of the technologies. Therefore, the method could not capture interdependencies between technologies.

All components in the NAS are dependent on each other. Therefore, the introduction of a single technology might not be enough to achieve an objective of the NextGen. To fully realize the benefits of the NextGen effort (e.g., fuel saving and reduced delays and emission), a set of new technologies in the NAS should be integrated together at the right time. In this context, it is important to answer these questions: what technology is the most critical in terms of improving the NAS performance? how much NAS performance can be improved by a set of technologies? which technology is delivered first (or when) to have the optimal increase of the NAS performance (e.g., order of technology development)? To answer these questions the SoS system engineers should deal with interdependencies not only among new technologies, but also between new technologies and the NAS performance. Previous works mentioned above only focus on the impact of a new technology on the performance in the whole or the part of the NAS. Thus, they could not address the impact of a set of technologies in the NAS. Mark, et al. (2013) developed the modeling and analysis framework to support risk-informed decision-

making for the NextGen. They estimated deliverable time of new technologies with probabilities but did not provide the criticality of new technologies. Therefore, there is a need of developing a means to analyze interdependencies not only between new technologies, but also between new technologies and the NAS performance. The proposed method will solve these challenges.

In this case study, any technical development or modification of the NAS is defined as a "technology", exemplified by new Air Transportation Management (ATM) infrastructure, new operational concepts, and new system components.

3.3.2 WHAT IS THE NEXTGEN?

The Federal Aviation Administration (FAA) has ongoing efforts to introduce the next generation air transportation system (NextGen) technologies in order to achieve a future air transportation system. The vision of the NextGen is to build on near- and mid-term (through 2018) systems developed by the FAA and other government partners, to improve performance and capacity of the National Airspace System (NAS) necessary to meet 2025 requirements (JPDO 2011). More specifically, the NextGen will allow more aircraft to safely fly closer together on more direct routes, reducing delays and providing benefits for the environment and the economy through reductions in carbon emissions, fuel consumption and noise. The main difference between the current air transportation system and the NextGen 2025 is that the NextGen 2025 is a satellite-based system enabling a more dynamic traffic flow management with ground automation. For example, in the current system pilots have to depend on their own vision information for separation assurance. In 2025, the NextGen will allow pilots to use cockpit display of traffic information which provides the location of the preceding aircraft for separation assurance.

The FAA has proposed a key work plan that includes 7 solution sets:

- Trajectory Based Operations (TBO)
- High Density Airports (HD)
- Flexible Terminals and Airports (FLEX)
- Collaborative Air Traffic Management (CATM)
- Reduce Weather Impact (RWI)
- Safety, Security and Environment (SSE)
- Transform Facilities (FAC)

These solution sets provide the key capabilities necessary to enhance airspace efficiency and safety. Figure 10 illustrates how the NextGen concept can create improved capabilities for each flight phase in a typical flight profile.

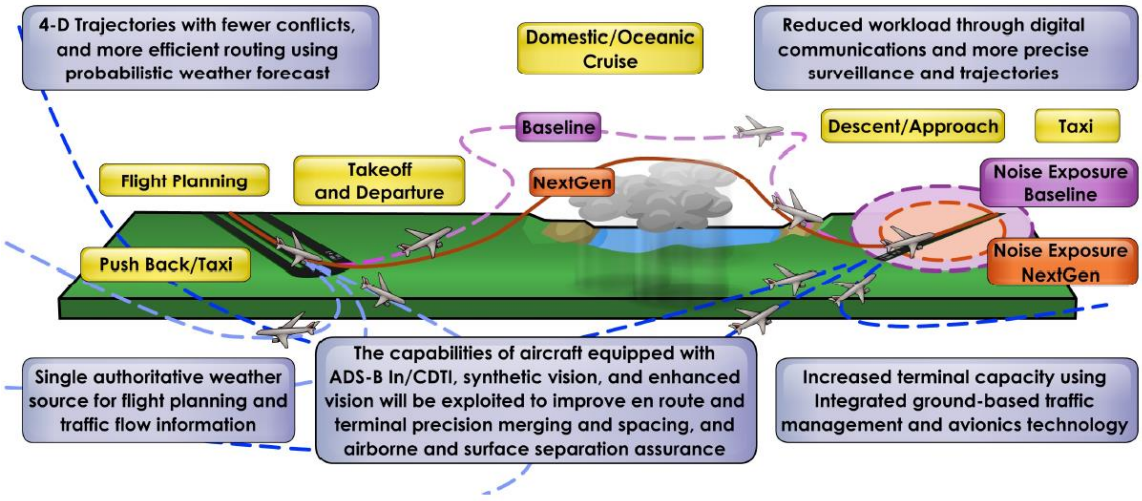


Figure 10: NextGen 2025 flight profile (JPDO 2011)

The following section briefly explains the new capabilities arising from the NextGen technologies in flight phases.

1. Flight planning phase
   NextGen technologies (e.g., four-dimensional weather products, net-centric operations, and etc.) allow collaborative decision making due to the availability of more up-to-date information on the status of the national airspace system such as weather conditions and projected demand. As a result, flight plans will be more accurate.

2. Push back and taxi-out phase
   NextGen technologies (e.g., aircraft moving map, Automatic dependent surveillance-broadcast (ADS-B), cockpit display of traffic information (CDTI), ground-based augmentation system (GBAS), and etc.) will change the communication method between pilots and traffic controllers from verbal to data communication. The new communication protocol will reduce the possibility of communication errors and misunderstanding. In addition to data communication, innovative tools for integrated surface management will sequence the aircraft departure and arrival efficiently to minimize taxi-out time.

3. Takeoff and departure phase
   Wake vortex mitigation tools and redesigned airspace will reduce delay in takeoff and departure phase by allowing more efficient departure sequencing and metering of flights. Wake vortex mitigation tools will reduce arrival/departure separation distances and increase capacity on closely-spaced runways. The use of area navigation (RNAV) and required navigation performance (RNP) procedures will increase throughputs in airspace of high-density terminal.

4. En-route cruise phase

The high accuracy of flight position information from ADS-B will allow flights to fly more precise flight trajectories. Thus traffic controllers can increase the capacity of en-route airspace by using improved metering and reducing the required minimum separation distance.

5. Descent and approach phase
   Time-based airborne merging and spacing enabled by ADS-B will allow more accurate time of arrival. RNP procedure will also increase terminal airspace capacity by allowing optimal profile descents. Synthetic vision and enhanced vision technologies will increase throughputs in low-visibility conditions.

6. Taxi-in phase
   The integrated surface automation system will provide an efficient and conflict-free path to pilots via data communication system. The efficient path will allow reduction of taxi time to the gate.

Research activities on NextGen technology development, integration, implementation and safety must be accomplished to achieve the benefits mentioned above. The interdependencies that exist between the various pairs of NextGen technologies warrant analysis of not only individual implementations but of the interacting conglomerate of technologies. Therefore the model should have an ability to analyze not only the interdependencies of technologies, but also the impact of a single technology or set of technologies at the system level.

3.3.3 Hierarchical Representation of NextGen

Complex interdependencies between the NextGen technologies create challenges when assessing the impact of a set of technologies on the NAS. We use hierarchical representation of the NextGen to understand how the capability of the NextGen varies when requirements or the NextGen technologies change. Then NextGen hierarchy is decomposed into three primary levels of: NextGen capability, requirements, and NextGen technologies.

The NextGen technologies are intended to improve the airspace flow management necessary to meet 2025 requirements by allowing reduction in flight delay, increase of throughputs, and reduction in environmental emissions, while maintaining and improving the safety level of the NAS. In this case study, we focus on reduction in flight delay. Thus, we define ‘provide service in the NAS with lowest possible mean dealy’ as the NextGen capability and can measure the performance of the NextGen by estimating the percent of ideal delay reduction obtained.

The capability of the NextGen can be achieved by implementing three requirements: improvement of departure flow management, airborne merging and spacing management, and arrival flow management. Figure 11 shows the decomposition of the NextGen capability into requirements and necessary NextGen technologies. Each requirement is satisfied by a set of NextGen technologies, and several NextGen technologies contribute to the fulfillment of more

than one requirement. The hierarchical representation shown in Figure 11 provides the backbone for analyzing interdependencies between the NextGen technologies when assessing the Bayesian Network. As shown in Figure 11, all NextGen technologies and requirements are strongly dependent on each other to achieve the objective of the NextGen. For instance, in the arrival flow management, there might be a situation that the airport has unexpected traffic congestion in taxiway/runway of an airport due to bad weather. At that moment the flights that are supposed to arrive at that airport have en-route delay by being hold in en-route, even though traffic conditions in en-route are good.

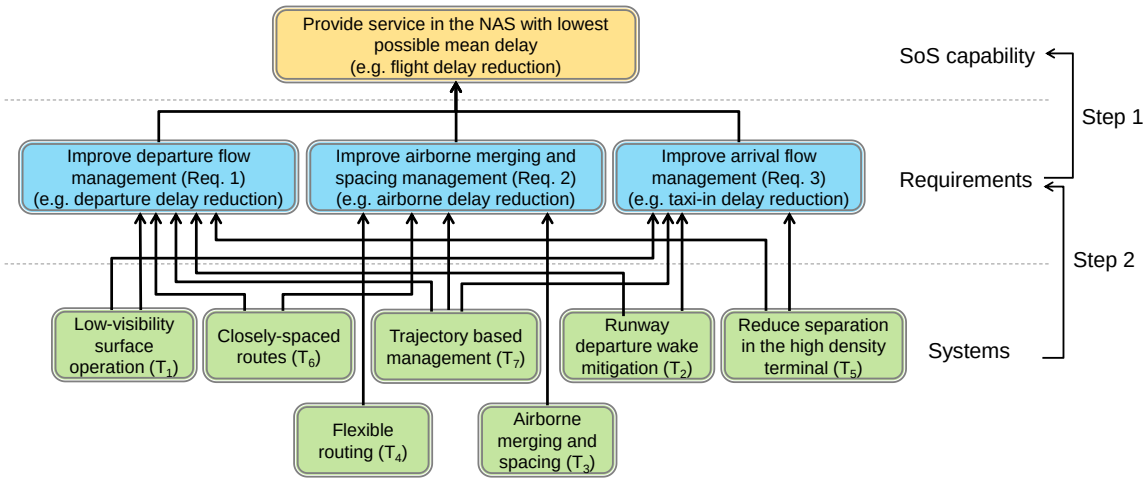


Figure 11: Hierarchical representation of the NextGen

3.3.4 Interdependency Analysis between NextGen Capabilities and Technologies

Implementing interdependency analysis between the NextGen capability (e.g., % of ideal delay reduction obtained) and technologies is a two-step process. The first step is to assess the impact of requirement state conditions on the NextGen capability. In other words, the first step quantifies how much the improvement of departure/en-route/arrival flow management affects the reduction in average arrival delay. The second step is to assess the impact of the NextGen technologies on requirement states. The following sub-sections discuss these two steps.

3.3.4.1 Step 1 – Build model of relationship between NextGen capability and requirements using BNs

Assessing the impact of the NextGen technologies on the NAS requires a modeling & simulation tool. We adopt Bayesian Networks as our tool of choice due to suitability of the method in estimating interactions among delays between airports. More specifically, the proposed BN provides the means to estimate average arrival delay under certain conditions such as congestion levels in the terminal and taxiway of the departure airport, en-route, and

the taxiway of the arrival airport. For example, the BN estimates average arrival delay given departure delay at the departure airport, en-route delay, and taxi-in delay at the arrival airport.

The NextGen technologies allow an efficient flow management in all flight segments. We can interpret an efficient flow management as the reduction in delay in each segment. For example, the Airport Moving Map (AMM), which is one of the NextGen technologies, with Global Positioning System (GPS) data can reduce taxi-out delay on the airport surface by allowing pilots to identify and anticipate the location of their/other airplanes. In this case study we use the reduction of departure, en-route, and taxi-in delays as input variables to generate the distribution of average arrival delay.

Two of the busier airports are selected for this simulation: Chicago O'Hare International Airport (ORD) and Hartsfield Atlanta International Airport (ATL). ORD and ATL are used as the departure and arrival airports respectively. The proposed BNs focus on quantifying how flight delays in taxiway and en-route affect flight arrival delays. In order to estimate parameters in the BN we use the data from the FAA ASPM database Quarter Hour Report from January 2012 to December 2012. The reports selected for this study are Analysis By Quarter Hour Airport Report, Daily Weather By Quarter Hour Report, and City Pair By Quarter Hour Report between ORD and ATL. Total number of data for this study is 8,906.

We define the variables in the BN model according to the ASPM database definitions. In the definition, all time is in minutes. The number of operations at ORD (*Opts_ORD*) refers to the total number of departures/arrivals from ORD (or any origin) to any destination (or ORD) during a given 15 minute period. It represents how busy the ORD airport is during a given 15 minute period. On the other hand, the number of operations at ATL (*Opts_ATL*) represents the total number of departures/arrivals from ATL (or any origin) to any destination (or ATL) during a given 15 minute period. Weather condition includes two states: VMC (Visual Meteorological Conditions) and IMC (Instrument Meteorological Conditions). Gate departure delay (*Gt_dep_dly*) refers to the difference between actual gate out time and the flight plan gate out time. Taxi-out delay (*Tx_out_delay_ORD*) refers to the difference between taxi-out time and unimpeded taxi-out time which is selected by airport, carrier, and season at ORD. Departure delay (*Dep_dly*) means the actual wheels off minus the flight plan gate out plus the unimpeded taxi-out time. Airborne delay (*Arb_delay*) is the difference between actual airborne time and the flight plan estimated time en-route from ORD to ATL. Arrival delay (*Arri_delay*) is the difference between actual gate-to-gate time and scheduled gate-to-gate. Taxi-in delay (*Tx_in_delay_ATL*) denotes the difference between taxi-in time and unimpeded taxi-in time at ATL.

The proposed BN model was created based on the BN structure provided by Xu et al. (2005). They developed the BN structure based on expert judgment and validated it against empirical data to investigate and visualize propagation of delays among airports. Figure 12 shows the proposed BN structure. The circles highlighted in green describe the relationship among the total number of operations, weather conditions, taxi-out delay, gate departure delay, and

departure delay during a given 15 minute period at ORD airport. Similarly, blue circles represent the same information at ATL airport. Pink circle denotes the airborne delay from ORD to ATL airports. These three groups (e.g., green, pink, and blue circles) are used to represent the state of the three requirements 1, 2, and 3 respectively. Three outputs from these three clusters are connected to the *Arri_delay* node. The *Arri_delay* node represents the arrival delay of flights from ORD to ATL airports. The arrival delay of flights will be used to capture the SoS capability by calculating reduction of arrival delay of a flight.

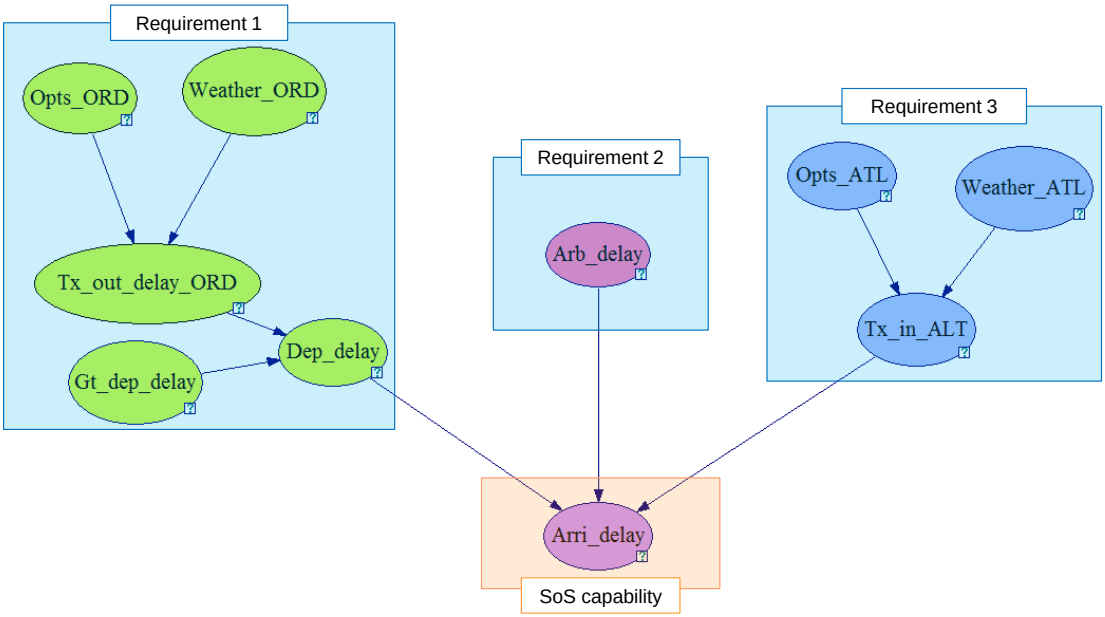


Figure 12: The proposed BN structure

The proposed BN also allows for the estimation of the arrival delay given other node conditions. Figure 13 and Figure 14 represents the impact of weather on arrival delay of a flight. When weather conditions change from VMC to IMC at both airports, the distributions in arrival delay node also change as shown in Figure 13 and Figure 14. In addition to the change of arrival delay distributions, the expected arrival delay changes from 6.1 to 6.9 minutes because bad weather increases the elapsed time of departure and arrival.

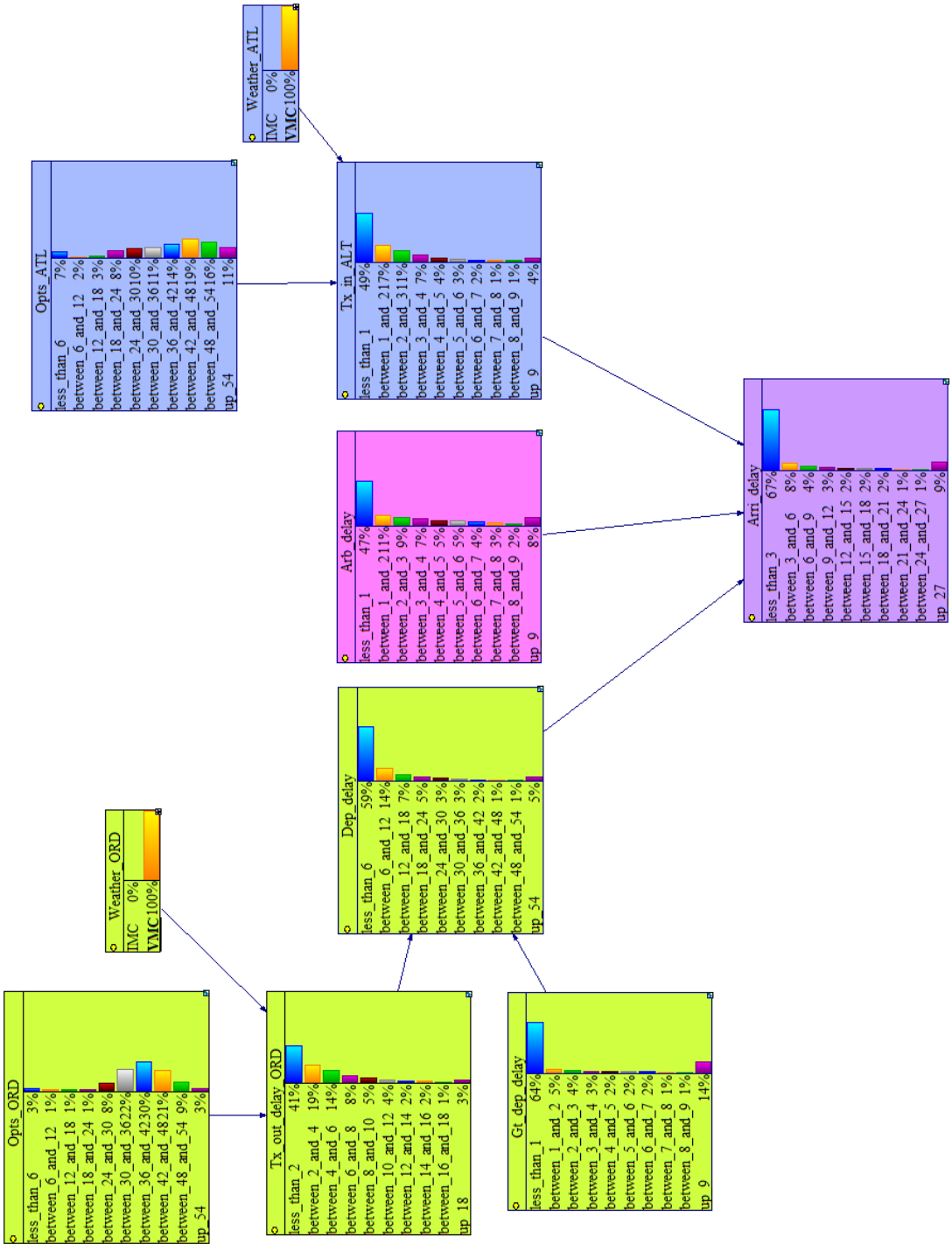


Figure 13: Weather effects on arrival delay of a flight under VMC

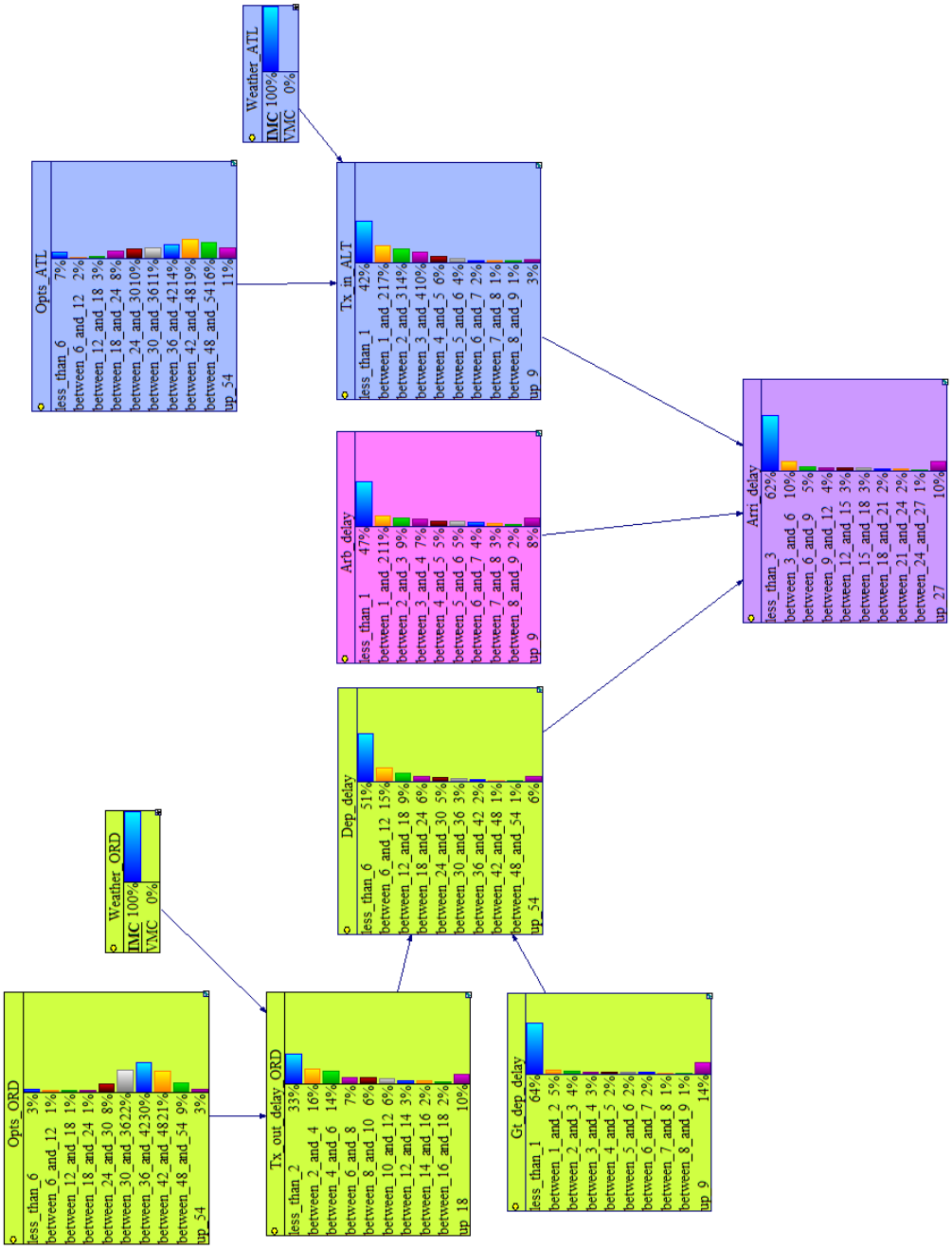


Figure 14: Weather effects on arrival delay of a flight under IMC

We use K-fold cross-validation technique to validate the proposed BN model. In K-fold cross-validation, the original data is randomly divided into *k* equal size subsamples. A single subsample is included in the testing group among *k* subsamples to test the BN model and to evaluate its prediction accuracy. The remaining (*k* – 1) subsamples are included in the training group to build the BN model and to estimate the parameters of it. Then the cross-validation process is repeated k times by changing the testing group to the next subsample. In this study, we use 10-fold cross-validation for the model validation.

The BN model is validated using GeNIe's built-in testing tool to evaluate the prediction accuracy of the model (GeNIe&SMILE 1998). Once the parameters in the BN model are estimated by cases in the training group, all cases in the testing group are tested one by one. For each case, GeNIe reads the values of nodes except the value of arrival delay. Then, the value predicted by GeNIe is compared with the true value of arrival delay node. The prediction accuracy is measured using a 'confusion matrix' and error rate. The columns of the confusion matrix are the predicted values of arrival delay in minutes, and the rows are the actual values. If the BN model is performing well, then the entry values along the main diagonal will be large (i.e., many occurrences of predicted matching actual) compared to those off diagonal. The error rate is the ratio of the number of incorrectly predicted cases to the number of all testing cases, shown and computed in (5).

$$Error\ rate = \frac{\#\ of\ incorrectly\ predicted\ cases}{\#\ of\ all\ testing\ cases} = \frac{1880}{8906} = 0.21 \qquad (3.1)$$

Table 5 represents the confusion matrix for arrival delay predictions. The proposed BN model presents 79% accuracy rate, showing potential use for predicting arrival delay.

Table 5: Confusion matrix for arrival delay prediction

		Predicted value									
		Less than or equal to 3	3 to 6	6 to 9	9 to 12	12 to 15	15 to 18	18 to 21	21 to 24	24 to 27	Greater than 27
Actual value	Less than or equal to 3	5698	79	21	23	24	3	2	1	0	4
	3 to 6	393	108	20	20	23	13	3	3	0	2
	6 to 9	204	52	40	30	21	9	11	5	0	5
	9 to 12	117	35	23	48	34	12	7	6	0	2
	12 to 15	46	23	12	25	58	13	9	12	4	11
	15 to 18	32	23	7	15	21	45	16	13	7	10
	18 to 21	17	5	11	14	21	12	50	14	4	22
	21 to 24	7	5	3	4	6	11	20	38	7	19
	24 to 27	4	2	0	3	12	7	8	13	21	28
	Greater than 27	19	9	3	10	14	10	11	14	5	920

We use sensitivity analysis to determine the impact of the variable factors (e.g., departure, airborne, taxi in, and gate departure delays) on the result of the model (e.g., the expectation arrival delay). The objective of this analysis is to indicate which of the factors affect the result of the model most. In other word, we identify the level of sensitivity of each variable factor to the result of the model. The result of the sensitivity analysis is shown in Figure 15. It is shown that

the higher the slope of departure, airborne, taxi in, and gate departure delays, the more sensitive to the expected arrival delay. The gate departure delay is the most sensitive to the expected arrival delay and following departure delay. Interesting observation is that airborne and taxi in delays is not sensitive until 8.5 minutes, but after that time these values are sensitive to the expected arrival delay. We can see that 8.5 minutes is the threshold of these two variables for increase of the expected arrival delay.

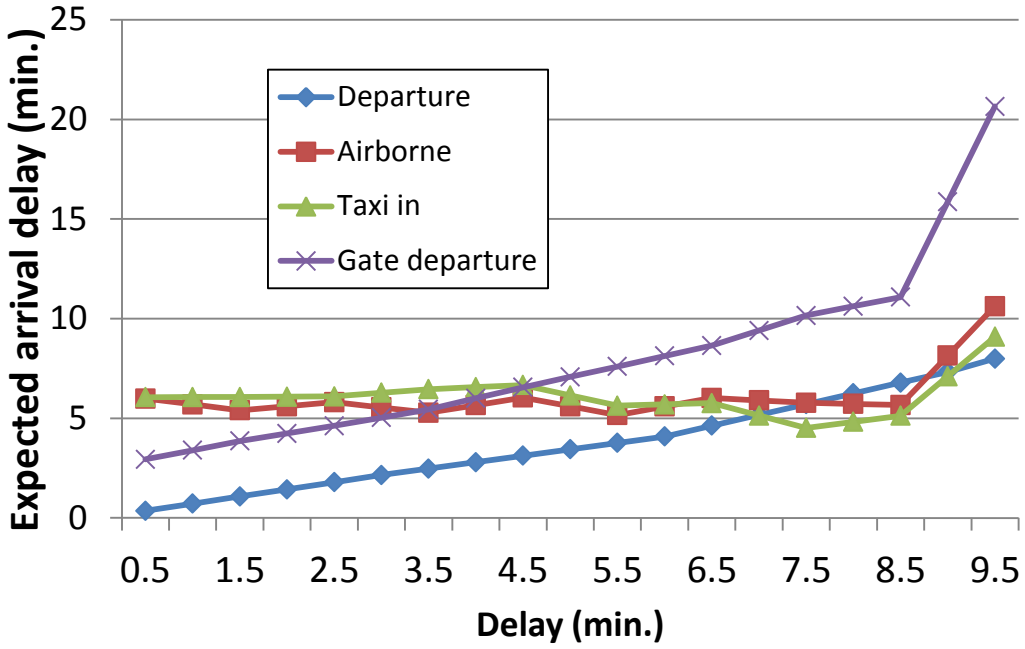


Figure 15: Sensitivity of delay of each of the four segments to expected arrival delay

3.3.4.2 Step 2 – Build model of relationship between requirements and NextGen technologies

We select seven different technologies to illustrate the impact of combined NextGen technologies on departure, airborne, and taxi-in delays. Table 6 shows the NextGen technologies used for this study and their respective technical impact on departure, airborne, and taxi-in delays. To estimate the technical impact values, we can use the results from other studies which focus on the analysis of impact of a specific technology on one of three areas (e.g., departure, airborne, and taxi-in). For example, Mayer (2006) analyzed departure efficiency benefits of terminal Area Navigation (RNAV) operations at Dallas-Fort Worth International Airport. The RNAV procedure is one of the influencing factors behind the separation reduction in high density terminal areas. The author compared pre- and post-implementation departure delays of the RNAV departure procedure. The average departure delay reduces from 4.8 minutes in conventional operations to 3.5 minutes in the RNAV implementation operation (i.e., departure delay reduction of 27%). Yousefi, et al. (2010) assessed the benefits of the user-preferred routes (in other words, flexible routing), which is one of the NextGen concepts, using corridor networks. They created contiguous corridors by connecting the high density and usage corridor elements to induce high traffic flow rate. Then

they computed the delay reduction when using corridor networks as compared to the baseline no-corridor case. When the total length of the corridor is 52,000 nm, the delay is reduced by about 60%. Due to lack of full information on technology impact values, we use notional values to illustrate the assessment of the impact of combined NextGen technologies on the NAS to demonstrate the proposed approach.

Even though the technical impact values are available from various studies, the values still might be subject to uncertainty due to different simulation settings and assumptions. Thus we represent the technical impact values using normal distributions in the proposed simulation to address uncertainty. The values (or ones in parenthesis) in Table 6 represent mean values (or standard deviations) of technology impact. Accounting for uncertainties in the technology impact values allow us to generate robust results by providing mean values with 95% confidence interval.

Table 6: NextGen technologies of interest and their technical impact on delay reduction

NextGen technologies	Reduction in departure delay at origin	Reduction in airborne delay	Reduction in taxi-in delay at destination
T_1: Low-visibility surface operation	-15% (σ=2)	-	-15% (σ=2)
T_2: Runway departure wake mitigation	-10% (σ=1)	-	-10% (σ=1)
T_3: Airborne merging and spacing	-	-20% (σ=3)	-
T_4: Flexible routing	-	-25% (σ=3)	-
T_5: Reduce separation in high density terminal	-20% (σ=2)	-	-20% (σ=2)
T_6: Closely-spaced routes	-25% (σ=4)	-15% (σ=4)	-25% (σ=4)
T_7: Trajectory-based management	-10% (σ=2)	-10% (σ=2)	-10% (σ=2)

The impact of any combined set of technologies can be greater or less than the sum of the impacts of all individual technologies. These impact values of any combined technologies depend on the relationship between technologies. For example, if technology A and B can collaborate with each other, then they can create a higher impact than the sum of two individual impacts due to the effects of synergy. Therefore, we need to define a parameter for each set of technologies to gauge the synergistic relationship. Since we have three categories (e.g., departure, airborne, and taxi-in) which can be affected by technologies, we use three different parameters (e.g., α for departure, β for airborne, and γ for taxi-in) to denote the synergistic relationships. These parameters can be estimated using a technology influence map proposed by Pinon et al. (2011). The technology influence map allows for estimating relationships that exist between technologies and operational improvements. Then, we define the combined impact of technologies (e.g., T1, T2) on departure delay as follows:

$$T_{1,2,\dots} = \alpha_{1,2,\dots}(T_1 + T_2 + \cdots) \tag{3.2}$$

For example, if we want to know the impact of T_1 and T_2 as shown in Table 6, we can easily calculate it with a parameter $\alpha_{1,2}$ using (6). If $\alpha_{1,2}$ is 0.8, the impact of T_1 and T_2 on departure delay is $-0.8 \times (10 + 5)\% = -12\%$. For this study, notional values for all parameters for synergistic relationship are used and set equal to 0.8.

3.3.4.3 Impact of combined NextGen technologies on NextGen capability (e.g., reduction of average arrival delay)

In this study, the performance of the NAS is defined as the percentage of average arrival delay reduction. We choose the nominal distributions of departure, airborne, taxi-in, and arrival delays of all domestic U.S. operations from ASPM data as the baseline. At every simulation run, the distributions are updated based on combined technology impacts. For example, if combined technologies can reduce departure delays by 20%, all departure delays in the ASPM data are updated to the reduced delay values. Then we can obtain new distributions of average departure, airborne, and taxi-in delays. These new distributions are used as inputs to the BN in the step 1.

The process of obtaining the performance of the NAS given a set of the NextGen technologies can be outlined as follows. First, we estimate the reduction in departure, airborne, and taxi delays brought about by the NextGen technologies using Table 6. Then, the new delay values are generated by applying technology impacts to the nominal delay values of 8,906 data. The new delay values of departure, airborne, and taxi-in are used as inputs to the Bayesian Network analysis as we discussed in Section 3.2.4.1 to estimate expected average arrival delays. Once we obtain the expected average arrival delay, we can calculate the percentage of average arrival delay reduction by comparing it with the expected average arrival delay obtained from the baseline case. We identify the critical NextGen technologies, in terms of the reduction of average arrival delay. Monte Carlo sampling method is used to address uncertainty of combined technology impacts on the expected average arrival delay. 10,000 samples are derived from the Monte Carlo sampling method to generate mean values and standard deviations of the expected average arrival delay.

We calculate the increase in NextGen performance by the addition of a particular NextGen technology. We apply all seven NextGen technologies to the NAS one by one, and then calculate the performance increase in the NAS by the addition of a particular technology. The technologies corresponding to higher values of percentage of average arrival delay reduction indicate the critical systems. As shown in Figure 16, T6 has the largest mean of percentage of average arrival delay reduction and has high uncertainty due to larger standard deviations on technology impact of T6. In conclusion, the results from Figure 16 can assist SoS system engineers in identifying the most effective NextGen technologies that contribute to increased expected NextGen performance.

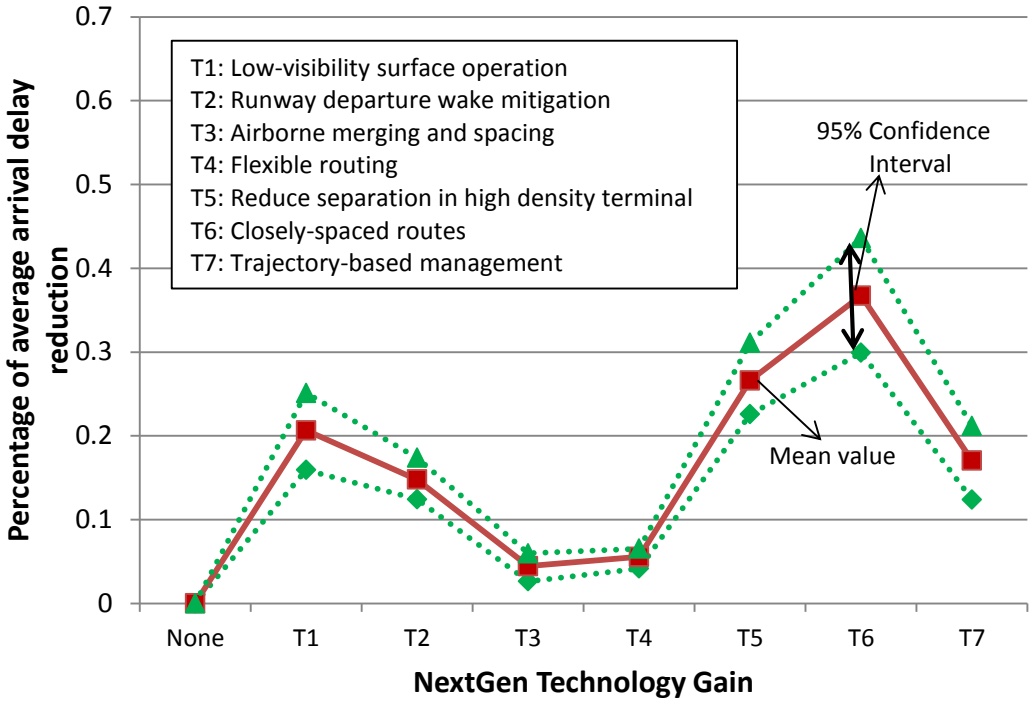


Figure 16: Increase of NextGen performance with one NextGen technology addition

3.3.5 Comparison of All Possible Development Processes of the NextGen Technologies

For optimal decision making, the SoS system engineers should update the development plan after a certain period of time. If all development actives proceed in accordance to their own schedule and there is no change in the budget or the NextGen requirements, then the development plan need not be updated. However, in the real world this may not be the case. When updating the development plan, the SoS system engineers need to know the impact of combined technologies on the NAS performance. Figure 17 shows the range of possible impacts of combined technologies. X-axis represents number of NextGen technologies in a development set and y-axis denotes percentage of average arrival delay reduction. Thus, in the first set there should be seven dots. The results from this figure can support decision making for updating the development plan. Whether or not a set of combined technologies is necessary depends on the value the decision-maker assigns to the system performance.

There could be a situation when some technologies have been already developed and incorporated within the NAS. In this case, when updating the development plan, the SoS system engineers should consider these available technologies. If the SoS system engineers know all possible alternatives given certain developed technologies, it supports the decision making process in updating the development plan. Figure 18 shows one possible example when T6 is already developed.

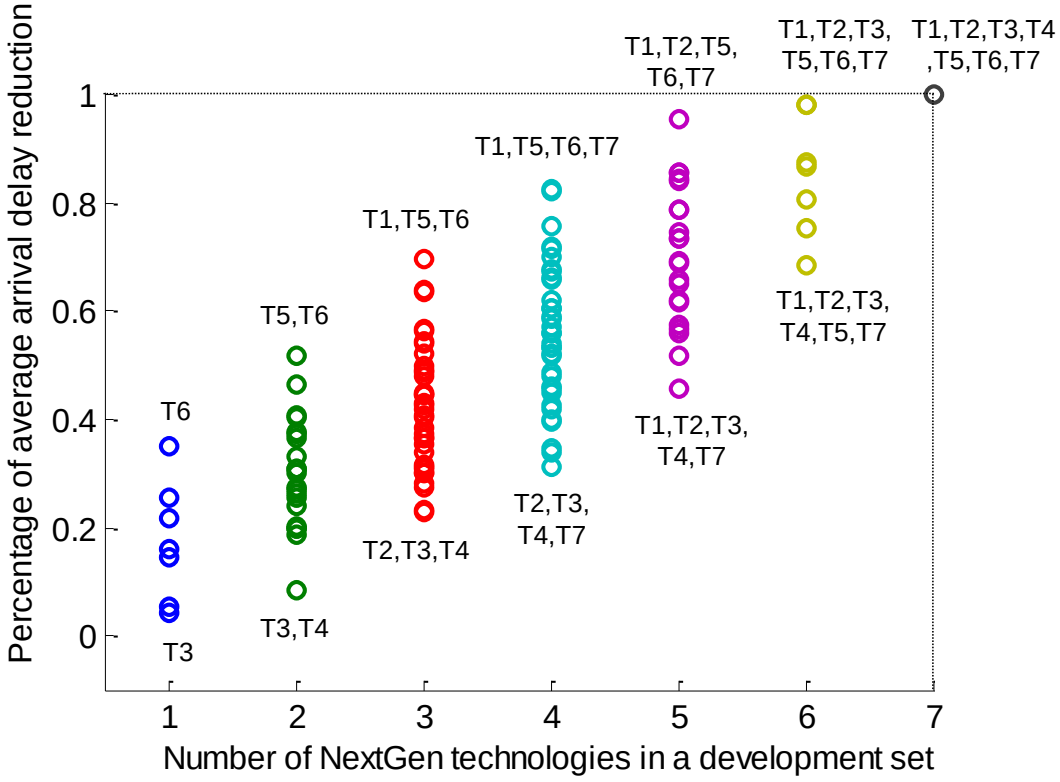


Figure 17: Benefit of average arrival delay with all possible sets of NextGen technology

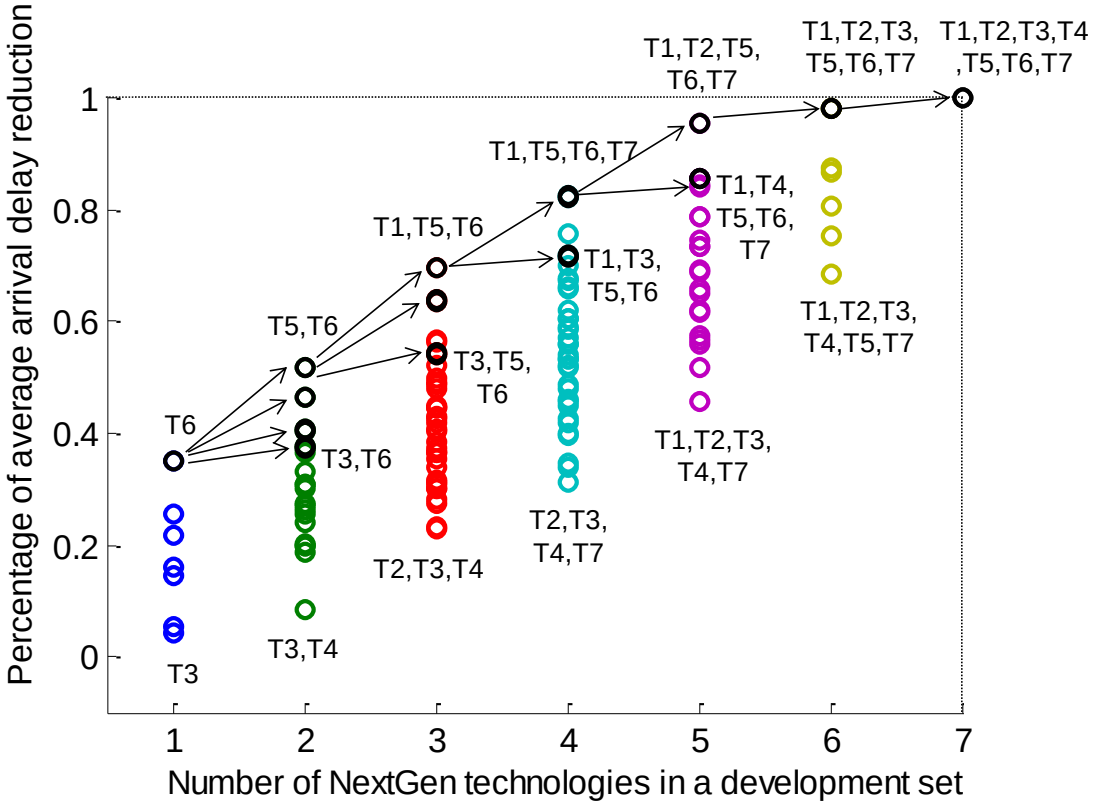


Figure 18: All possible development processes given a developed technology of T6

3.3.6 Cost-Benefit Analysis of Combined NextGen Technologies

In the previous section, we showed how our approach can be used to provide the best development scenarios for high NextGen performance and all possible development processes. These analyses provide only performance information. However to make an informed decision when adding new NextGen technologies, the SoS system engineers should consider not only the improvement of NextGen performance but also the cost incurred due to the new technologies. The cost of adding NextGen technologies can be approximated by the acquisition and operating costs. Since operating costs are very difficult to estimate before implementing the new technologies, we restrict our attention to the estimated acquisition cost. Each new NextGen technology can be achieved by completing a set of acquisition programs (Dillingham 2008), (JPDO 2011). For example, completion of five acquisition programs (e.g., airport surface detection equipment, automatic dependent surveillance broadcast, traffic flow management infrastructure, airport movement area safety system, and traffic management advisor) will allow low-visibility surface operation. Table 7 shows core acquisition programs and estimated acquisition cost for NextGen implementation (GAO 2012). Table 8 shows the list of required acquisition programs and total acquisition cost to implement NextGen technologies.

Table 7: Estimated cost of individual acquisition programs (US$M, FY June 2008)

Program	Estimated acquisition cost at completion	Program	Estimated acquisition cost at completion
P_1: Free flight traffic management	$135.5	P_2: Airport surface detection equipment	$550.1
P_3: En route automation modernization	$2,154.6	P_4: Next Generation air-to-ground communication	$324.7
P_5: Standard terminal automation replacement	$2,719.2	P_6: Airport surveillance radar	$696.5
P_7: Aviation surface weather observation network	$384.3	P_8: Integrated terminal weather system	$286.1
P_9: Instrument flight procedures automation	$50.8	P_{10}: Terminal automation modernization replacement	$139.5
P_{11}: Automatic dependent surveillance broadcast	$1,678.2	P_{12}: Traffic flow management infrastructure	$398.1
P_{13}: System-wide information management	$96.6	P_{14}: En route control center system modernization	$167.9
P_{15}: Airport movement area safety system	$151.7	P_{16}: Traffic management advisor	$135.5
P_{17}: Precision runway monitor	$145.8	P_{18}: En route communication gateway	$315.1

Table 8: List of required acquisition programs and total acquisition cost to implement NextGen technologies

NextGen technologies	Required acquisition programs to achieve NextGen technology	Total estimated acquisition cost for NextGen technologies (US$M, FY June 2008)
T_1: Low-visibility surface operation	P_2, P_{11}, P_{12}, P_{15}, P_{16}	$2,913.6
T_2: Runway departure wake mitigation	P_5, P_6, P_7, P_8, P_{10}, P_{12}, P_{16}, P_{17}	$4,905.0
T_3: Airborne merging and spacing	P_3, P_4, P_9, P_{11}, P_{12}, P_{13}, P_{14}, P_{16}, P_{18}	$5,321.5
T_4: Flexible routing	P_1, P_3, P_4, P_{11}, P_{12}, P_{13}, P_{14}, P_{16}, P_{18}	$5,406.2
T_5: Reduce separation in high density terminal	P_4, P_5, P_8, P_9, P_{10}, P_{11}, P_{12}, P_{16}	$5,732.1
T_6: Closely-spaced routes	P_3, P_4, P_9, P_{11}, P_{12}, P_{16}	$4,741.9

T_7: Trajectory-based management	P_2, P_4, P_6, P_{11}, P_{12}, P_{13}, P_{16}, P_{17}	$4,025.5

Figure 19 shows the increase in the NextGen performance against the cost incurred due to the addition of new NextGen technologies. The X-axis represents the estimated acquisition cost incurred due to new NextGen technologies and the Y-axis denotes percentage of average arrival delay reduction as compared to the baseline case. For example, adding technology T1 increases the NextGen performance by 22% while adding combined technologies T1 and T7 increases the NextGen performance by 24%. The NextGen performance benefit of both cases is almost similar, but the first case is less costly.

Each circle represents one of the possible NextGen technology combinations. The cost of a NextGen technology combination can be estimated by the cost of the set of acquisition programs required for implementing the NextGen technology combination. Since many NextGen technologies require common acquisition programs as shown in Table 8, there are many cases that different NextGen technology combinations have the same cost due to the same acquisition program required.

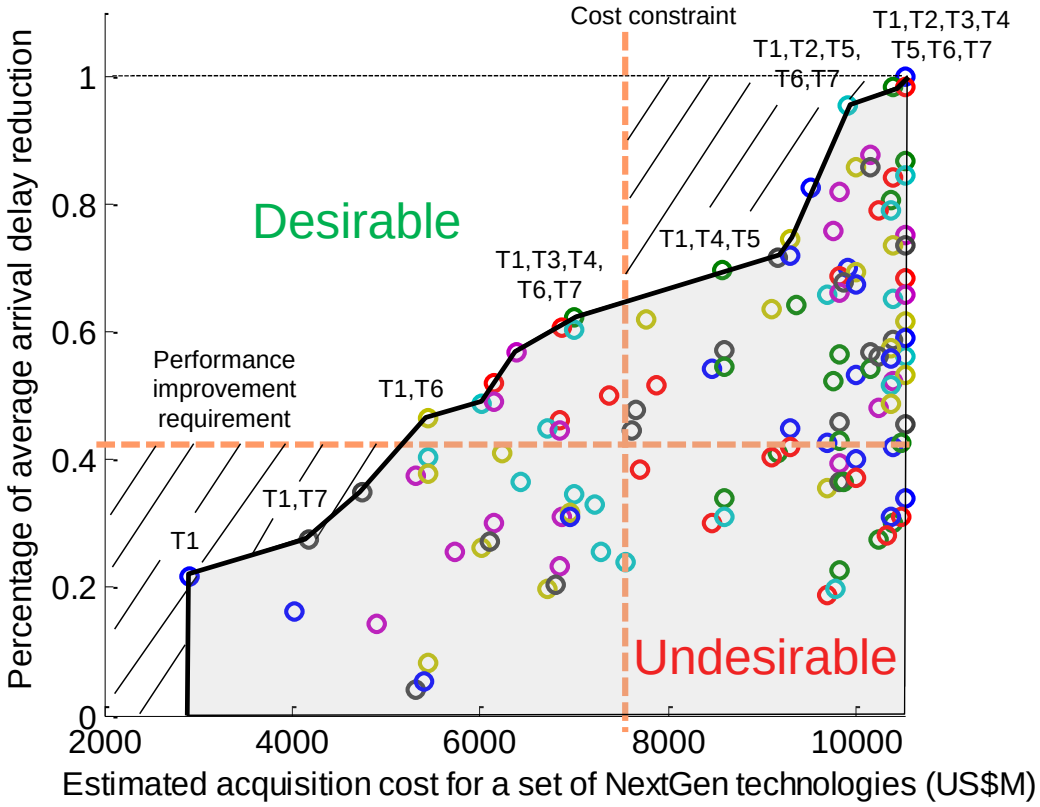


Figure 19: The cost-benefit analysis of combined NextGen technologies

Whether or not new NextGen technologies are necessary depends on how much the improvement of the performance is worth to a given decision maker. Figure 19 shows an example of how the space can be divided into two regions: desirable and undesirable. We use Pareto frontier to identify these two regions. The area inside the Pareto frontier is undesirable due to high cost and low performance improvement. There might be cost constraints (e.g., vertical dotted line in the figure) and performance improvement requirements (e.g., horizontal dotted line in the figure) when adding new NextGen technologies. With these constraints, the desirable design space is reduced to just the top-left quadrant. In this case, the combined technologies in the top-left quadrant and on the frontier line are desirable alternatives. This example illustrates the different alternatives available to the decision maker based on performance and cost.

As shown in Figure 19, technology T1 (e.g., Low-visibility surface operation) is included in all sets of NextGen technologies on the Pareto frontier. It means that technology T1 not only contributes to improvement of delay reduction alone but also allows for generating synergy effects as working together with other technologies.

To summarize, we have analyzed the interdependencies between the NextGen capability and technologies using BNs. The main result of this case study is to estimate the impact of combined NextGen technologies on the NAS system. Furthermore, the proposed approach provides the ability to identify the most critical technologies, in terms of the reduction of flight arrival delays, and all possible development scenarios using the entire design space. The main purpose of the case study is demonstration of the proposed approach in a useful context. For this case study, we used notional values for input variables and made many assumptions due to lack of input information. Therefore the results in this case study may not have enough quality that could/should be acted upon by FAA.

3.3.7 Conclusion and Future Work

In this section we propose an approach which facilitates early decisions and planning in the domain of the NAS. In this case study we analyze the interdependencies between the NextGen capability and technologies using BNs. The main result of this case study is to estimate the impact of combined NextGen technologies on the NAS system. Furthermore, the proposed approach provides the ability to identify the most critical technologies, in terms of the reduction of flight arrival delays, and all possible development scenarios using the entire design space. Cost-benefit analysis provides a feasible design space given cost and performance improvement constraints. The design space allows SoS system engineers to select the best SoS alternative for the given conditions.

The main purpose of the case study is demonstration of the proposed approach in a useful context. For this case study, we used notional values for input variables and made many assumptions due to lack of input information. Therefore the results in this case study may not

have enough quality that could/should be acted upon by FAA. In future work, we will develop an approach which guides us to select input values reasonably.

3.4 Mechanism Design Approach for Bandwidth Allocation in Tactical Data Links

This work focuses on improving the quality and accuracy of the common operating picture of a tactical scenario through the efficient allocation of bandwidth in the tactical data networks among self-interested actors, who may resort to strategic behavior dictated by self-interest. We propose a two-stage bandwidth allocation mechanism based on modified strictly-proper scoring rules, whereby multiple agents can provide track data estimates of limited precisions and the center does not have to rely on knowledge of the true state of the world when calculating payments. In particular, we emphasize the importance of applying robust optimization techniques to deal with the data uncertainty in the operating environment. We apply our robust optimization based scoring rules mechanism to an agent-based model framework of the tactical defense scenario, and analyze the results obtained.

3.4.1 Introduction

The defense sector is undergoing a phase of rapid technological advancement, in the pursuit of its goal of information superiority. This goal depends on a large network of complex interconnected systems – sensors, weapons, soldiers – linked by heterogeneous tactical data networks. Our research focuses on improving the quality and accuracy of the common operating picture through the efficient allocation of bandwidth in the tactical data links. The problem of bandwidth allocation is compounded by the self-interested behavior exhibited by the military commanders of each platform, who are more concerned with the well-being of their own platforms over others. Individual platforms benefit from receiving data from other platforms but have no incentive for sharing it. Thus, we can expect a tendency for platforms to under-represent the quality of their data so that the bandwidth is allocated to the transmission of data by others (Rogers et al., Klein et al.). Against this background, we propose a mechanism that efficiently allocates the flow of data within the tactical data network to ensure that the resulting global performance maximizes information gain of the entire system, despite the self-interested actions of individual actors.

In this section, we consider a multi-flag, multi-platform military scenario where a number of military platforms have been tasked with the goal of detecting and tracking targets. The platforms must share and exchange tactical data from their onboard sensors in order to establish and maintain a common operating picture (COP) of the tactical situation. The track data exchanged among sensor platforms encapsulates the sensor's own position as well as estimates of the position and dynamics of the observed targets. The exchange of tactical data among the platforms is facilitated over a standardized radio network, known as a TActical Data Information Link (TADIL). The sensors onboard the military platform have a partial and inaccurate view of the COP and need to make use of data transmitted from neighboring sensors over the bandwidth-constrained TADIL to improve the accuracy of their own measurements. The mission outcome can be significantly affected by decisions made in real time about which data to share. Ad-hoc bandwidth allocation can have serious repercussions and can even jeopardize a mission.

Reporting Responsibility (R2) rules is a minimal precedence based mechanism which permits only the unit with the best quality data (position, velocity, etc.) to report a surveillance track on the data link. This strategy prevents multiple track reports on the link for a single object, thus minimizing the data latency. However, it precludes any possibility of collaboration in building the COP by disallowing the redundant reporting of a single object. In our work we consider the R2 minimalist approach as our point of departure. We start with the premise that additional communication per network cycle can significantly improve the quality of the combined data and by enough to warrant the additional latency that comes from a longer cycle time. Thus, we seek to design a mechanism which can efficiently allocate a finite bandwidth, beyond what is used in the R2approach, to enhance the quality of the common operating picture. We encapsulate the desired features of the mechanism in Figure 20.

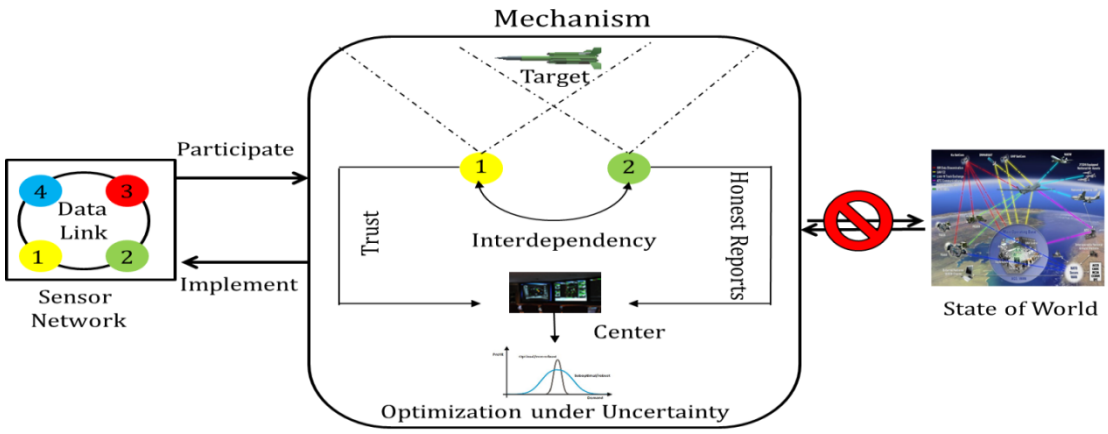


Figure 20: Framework for Bandwidth Allocation in Tactical Data Networks

The heart of the mechanism, which we need to design, resembles a portfolio optimization problem. The portfolio problem assumes that a portfolio needs to be constructed consisting of a set of stocks. Each of the stocks has a return and a risk value associated with it and the objective is to determine the fraction of wealth to be invested in each stock to maximize the portfolio value. In reference to our problem scenario, the stocks represent the observations made by the sensors. The return value of the stock can be regarded as the information content of each observation; the risk indicates the uncertainty in the reported data while the total wealth represents the bandwidth to be allocated on the tactical data link. Thus the objective is to determine which track information to select for transmission, given the fixed bandwidth available to maximize the total information content. The challenges of interdependency of the reported data, selfish behavior, constrained resources and dynamic uncertain environments dictate that our mechanism needs to go beyond a simple portfolio optimization. We highlight these challenges below:

- Voluntary Participation – Since sensor platforms are individually owned by different stakeholders, the mechanism must ensure that platforms participate voluntarily in lieu of some incentive of participation.

- Honest Reports (Incentive Compatibility) – The sensor platforms may resort to self-interested behavior and optimize their own gain from the network at a cost to the overall network performance. Hence the mechanism has to incentivize the platforms to truthfully reveal their private information (track data).

- Interdependency - In tactical sensor networks, since the observations made by the sensors are polluted by uncertainty and noise, the information content of a sensor's observation will be affected by the observations made by other sensor platforms. The mechanism must account for this information interdependency in the reported observations.

- Lack of access to the state of world – The mechanism should work even when the center has no access to the true state of world. The dynamic and uncertain nature of the operating environment means that the track data evolves between the time the information is reported and the time when it can be observed. Thus the center needs to evaluate the received reports without any knowledge of the true outcome.

- Optimization under uncertainty - The mechanism needs to account for the possibility that given the dynamic operating environment, there might be some uncertainty in the reported data.

- Implementation - The mechanism has to ensure that once the sensor platforms have been allocated to their respective targets, the selected platforms invest all their resources to track their assigned targets.

3.4.2 Previous Work

In order to address the requirements of private information and selfish behavior, mechanism design has been used in literature for achieving globally optimal behavior. The field of mechanism design lies at the intersection of economics and game theory and is concerned with designing protocols, and institutions that are mathematically proven to satisfy certain system-wide objectives under the assumption that individuals interacting through such institutions act in a self-interested manner and may hold private information that is relevant to a required decision. Mechanism design finds application in problems involving the allocation of scarce resources where both human and computational entities are inclined to resort to strategic behavior dictated by guile and self-interest; examples of this includes allocation of network bandwidth, storage capacity and power. Rogers et al. have studied the use of tools and techniques from computational mechanism design for information fusion within Sensor Networks. Klein et al. proposed an interdependent value mechanism design for bandwidth

allocation in Tactical Data Networks. Both these bodies of literature use a modified version of the Vickrey-Clarke-Groves (VCG) mechanism to achieve efficient bandwidth allocation and ensure truthful reporting by conditioning payments on the realized value for data shared between agents. The VCG mechanism is a sealed-bid second-price auction in which all bidders submit sealed bids individually but the winner pays the second-highest bid rather than their highest winning bid. The VCG mechanism suffers from well-documented vulnerabilities of bidder collusion and spiteful bidding and doesn't address our requirements of optimization under uncertainty, lack of access to the state of world and implementation. Given the shortcomings of auction-based mechanism we shift our focus from the realms of auction based models, to another promising alternative approach within the Mechanism Design research domain, in the form of scoring rules.

3.4.3 SCORING RULES

Scoring Rules have been proposed as a methodology to address the shortcomings of auction-based mechanism design for expected value maximization. Scoring Rules are used to assess the accuracy of probabilistic forecasts, by awarding a score based on the forecast and the event that materializes. Scoring rules provide a framework wherein the agents are incentivized to invest their resources in making accurate, high-quality assessments and reporting them truthfully. In our work, we are interested in strictly proper scoring rules. A strictly proper scoring rule is the one in which an actor can maximize his score by reporting exactly his or her true beliefs about the event. We shall restrict our discussion to the four most popular strictly proper scoring rules – quadratic, spherical, logarithmic and parametric – as we can analytically derive and express their expected values in closed forms.

One of the drawbacks of the auction-based Mechanism Design was that it did not account for agents not investing all their available resources in generating the observations. Miller et al. combat this issue through the introduction of scaling parameters. They show that the affine transformation of the scoring rules, does not affect the inherent properties of the scoring rules, like, incentive compatibility. We model an agent's noisy private measurement,x, as Gaussian random variable, $x \sim N(x_0, \theta^{-1})$where, x_0 is the true state of the observable and θ is the information content of the observation. If we denote the scoring rule by the function $S(x_0; x, \theta)$ and the expected score as $\bar{S}(\theta)$ then we can formulate the expected payment and utility as

$$\bar{P}(\theta) = \alpha\bar{S}(\theta) + \beta \qquad \bar{U}(\theta) = \alpha\bar{S}(\theta) + \beta - c(\theta) \tag{4.1}$$

where α and β are the scaling parameters and $c(\theta)$ is the cost of generating an observation with precision θ.

We can now compare the four different scoring rules – quadratic, spherical, logarithmic and parametric – for Gaussian probability density function $N(x_0; x, \theta^{-1})$. An important property of the strictly proper scoring rules is the concavity of the expected scoring rules to incentivize an

agent to produce truthful observations. Hence the parameter k for the parametric scoring rule family is restricted to the space $(1, 3)$ to ensure concavity. We also calculate the expected values along with the parameter expressions for the strictly proper scoring rules in Table 9.

Table 9: Scoring Rules for Gaussian distributions

	Quadratic	Spherical	Logarithmic	Parametric
$S(x_0; x, \theta)$	$2N - \left(\frac{\theta}{4\pi}\right)^{1/2}$	$\left(\frac{4\pi}{\theta}\right)^{1/4} N$	$\log N$	$kN^{(k-1)} - \frac{k-1}{\sqrt{k}}\left(\frac{2\pi}{\theta}\right)^{(1-k)/2}$
$\bar{S}(\theta)$	$\left(\frac{\theta}{4\pi}\right)^{1/2}$	$\left(\frac{\theta}{4\pi}\right)^{1/4}$	$\frac{1}{2}\log\left(\frac{\theta}{2\pi}\right) - \frac{1}{2}$	$\frac{1}{\sqrt{k}}\left(\frac{2\pi}{\theta}\right)^{(1-k)/2}$
α	$4c'(\theta_0)\sqrt{\theta\pi}$	$4c'(\theta_0)(4\pi\theta^3)^{1}$	$2c'(\theta_0)\theta_0$	$\frac{2c'(\theta_0)\theta_0\sqrt{k}}{k-1}\left(\frac{\theta_0}{2\pi}\right)^{(1-k)}$
β	$c(\theta_0) - 2\theta_0 c'(\theta_0)$	$c(\theta_0) - 4\theta_0 c'(\theta_0)$	$c(\theta_0) - 2\theta_0 c'(\theta_0)\left(\frac{1}{2}\log\left(\frac{\theta_0}{2\pi}\right) - \frac{1}{2}\right)$	$c(\theta_0) - \frac{2\theta_0 c'(\theta_0)}{k-1}$

Papakonstantinou et al. extended the concept of *modified* scaled strictly proper scoring rules to handle the lack of knowledge of the outcome while preserving the property of incentive compatibility. In modified strictly proper scoring rules, the trusted center fuses the observations from all the other agents and excludes the agent whose reported observation is being evaluated. In the absence of access to the true outcome, the center uses this fused set of observations to evaluate the agent's reported observations. Thus, based on the modified strictly proper scoring rule, an agent can maximize its expected score and by extension, their expected payments by truthfully reporting its observations, assuming that other agents in the system also honestly report their observations. This makes truthful revelation a Nash equilibrium and the optimal strategy for all agents in the system.

3.4.4 INTERDEPENDENT VALUATION

In tactical sensor networks, individual sensors have a limited and partial view of the common operating picture and produce uncertain and noisy observations. The value of one sensor platform for an allocation of bandwidth depends on private information held by other platforms, namely on the quality of their observed track data. The resulting information structure is one of interdependent valuations. A naïve extension of the VCG mechanism is known not to work in the case of interdependent valuations. Jehiel & Moldovanu have showed that, in an interdependent valuation setting, no standard one-stage mechanism can achieve both efficiency and incentive compatibility for procurement of estimates from multiple sources. Mezzetti addressed this challenge to a certain extent and showed that an efficient allocation with multidimensional types is possible, if two-stage mechanisms can be adopted in which the payments are made contingent on realized values reported in a second stage. Mezzetti

designed a two-stage mechanism: in the first stage the agents would submit their reports to the center, which would in turn, determine the allocation of the items among the bidding agents. In the second stage, the agents report their observations and receive the final payments from the center. Accordingly, we design a two-stage mechanism based on modified strictly proper scoring rules.

3.4.5 Robust Optimization

A two-stage mechanism can be constructed based on modified scaled strictly proper scoring rules, which selects a set of sensor agents to provide observations for a target. Since there are numerous targets in the system, we end up with different sets of sensor – target pairs. However we can only allocate a limited bandwidth for transmitting information over the tactical data network. Hence, we need a methodology to decide which sensor-target pairs to select for transmission to ensure the recovery of the highest gain in information for a given quantum of bandwidth. The problem is compounded by the inherent uncertainty in the information content of the observations. Deterministic optimization techniques that rely on nominal data, no longer work in these settings. Robust techniques provide an attractive choice in addressing the feasibility and optimality of the optimization solution, given the uncertainty in the data. We formulate our problem as a robust portfolio optimization problem.

$$\begin{aligned} &maximize && \sum_{k \in K, j \in L_k} \theta_{jk} z_{jk} \\ &subject\ to && \sum_{k \in K, j \in L_k} z_{jk} = N_{auc} \\ & && z_{jk} \in \{0,1\} \end{aligned} \tag{4.2}$$

where,

$\boldsymbol{K}$: Set of targets in the system

$\boldsymbol{L_k}$: Set of agents selected through the proper-scoring rules algorithm for target k

$\boldsymbol{\theta_{jk}}$: Quantification of covariance (information content) of the reported observation made by agent j of target k

$\boldsymbol{N_{auc}}$: The total number of agent-target pairs that can selected for transmission

$\boldsymbol{z_{jk}}$: Binary decision variable corresponding to which sensor-target pair is selected

The information content which is a calculated using the covariance of the reported observation is assumed to be uncertain. In other words, we model the information content as a random variable $\tilde{\theta}_{jk}$ that has a symmetric distribution in the interval $[\theta_{jk} - \hat{\theta}_{jk}\ ,\ \theta_{jk} + \hat{\theta}_{jk}]$. θ_{jk} is the expected information gain, while $\hat{\theta}_{jk}$ is the measure of the uncertainty of the information content. We adopt the robust linear framework proposed by Bertsimas & Sim to solve the portfolio problem. The Bertsimas-Sim framework is based on the premise that given a set of

uncertain dara elements, only a small subset of the elements takes their worst-case values at the same time. The formulation provides a protection-level Γ to control the degree of robustness of the solution. The parameter Γ guarantees a feasible solution for instances in which fewer than Γ parameters take their worst-case values. The approach even provides a probabilistic guarantee, that if more than Γ parameters change, the robust solution will still be feasible to a high degree of probability. The linear nature of the problem makes it extensible to discrete optimization problems.

$$
\begin{aligned}
&maximize && \sum_{k \in K, j \in L_k} \bar{\theta}_{jk} z_{jk} - \beta(z_{jk}, \Gamma) \\
&subject\ to && \sum_{k \in K, j \in L_k} z_{jk} = N_{auc} \\
&\beta(z_{jk}, \Gamma) = && \max_{\{S \cup \{t\} | S \subseteq J, |S| = \lfloor \Gamma \rfloor, t \in J \setminus S\}} \left\{ \sum_{jk \in S} \hat{\theta}_{jk} z_{jk} + (\Gamma - \lfloor \Gamma \rfloor) \hat{\theta}_{jk_t} z_{jk_t} \right\}
\end{aligned}
\tag{4.3}
$$

3.4.6 PROPOSED ALGORITHM

We design a two-stage mechanism based on the modified strictly proper scoring rules and robust optimization. In the first stage, the center preselects $\boldsymbol{M}$ of the $\boldsymbol{N}$ available agents based on the reported cost functions through a single $(\boldsymbol{M}+\boldsymbol{1})^{\boldsymbol{th}}$ sub-auction. In the second stage, the center announces the modified scaled strictly proper scoring rules and asks the $\boldsymbol{M}$ preselected agents to produce their observations. Each of the preselected agents produce and report their observations to the center, which in turn, calculates their payments based on the announced scoring rule. The center then selects the final sensor-target pairs based on the robust portfolio optimization and the selected sensor platforms report the observations on their allocated targets on the data link.

1. First Stage

1.1. The trusted center asks $N \geq 2$ sensor agents to report their cost functions $\hat{c}_i(\theta)$ and their maximum information content $\hat{\theta}_i^c$, $\forall i \in \{1,2,\dots,N\}$

1.2. The center selects M $(1 \leq M < N)$ sensor agents with the lowest costs, associates them with the $(M+1)^{th}$ cost and discards the rest of the sensor agents.

2. Second Stage

2.1. The center asks sensor agent j, selected in Step 1.2, to generate the observations and presents it with a modified strictly proper scoring rule with parameters α_j and β_j

2.2. Each of these sensor agents will produce an estimate x_j with information content θ_j and report $(\hat{x}_j, \hat{\theta}_j)$ to the centre which, in turn, issues the payments to all the sensor agents.

2.3. The center solves the robust portfolio optimization to select target-sensor pairs for transmission. The selected sensors are asked to broadcast the observations on their allocated targets.

3.4.7 RESULTS

In order to study the application of mechanism design in a practical context, we need a surrogate model for the real-world operation which exhibits the necessary fidelity and complexity. To this end, we leverage the Discrete – Agent Framework (DAF) developed at Purdue University to design an Agent-Based Model (ABM). We conduct multiple runs of the ABM by changing the starting positions and dynamics of the targets in the scenario and analyse the results of applying our modified scaled strictly proper scoring rules based mechanism to the simulation model.

3.4.7.1 Maximum Number of Preselected Sensor Platforms (M)

In the first stage of the proposed mechanism, the trusted centre preselects M sensor platforms from the N available sensor platforms with the lowest cost functions through one single reverse $(M+1)^{th}$ auction. In our simulation scenario, since we consider four sensor platforms and the R^2 tracks have already been pre-assigned, there are only $N = 3$ sensor platforms available for selection for transmitting non – R^2 track data. Thus $M \in [1,3]$ dictates the maximum number of sensor platforms that can track any one target. For example, $M = 2$ indicates that a maximum of 3 platforms can be assigned to one target; one for R^2 - track data and two for non-R^2 track information.

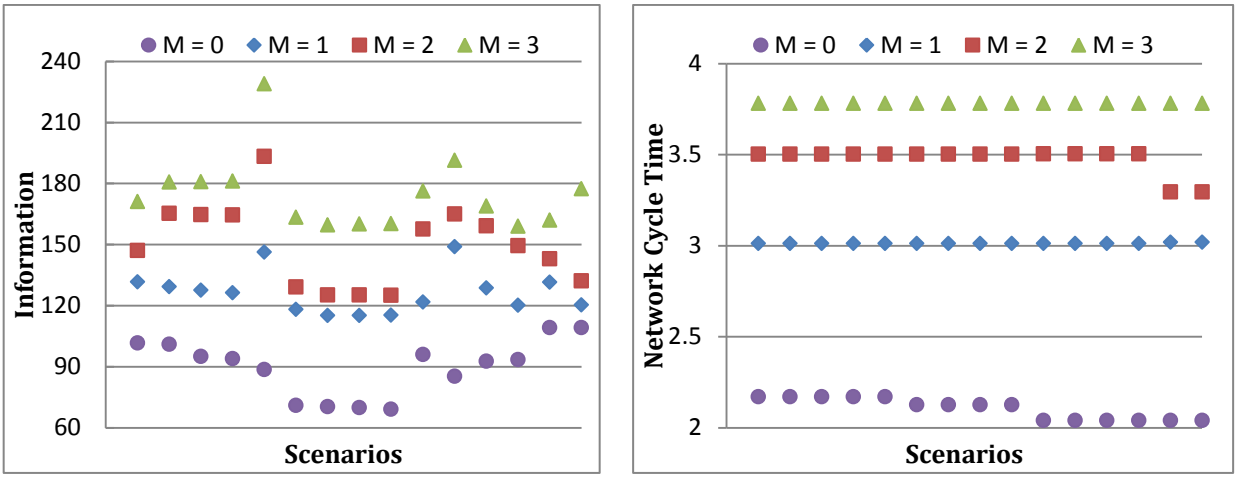


Figure 21: (a) Information for different values of M, (b) Network Cycle Time for different values of M

Figure 21(a) and 21(b) represents the variation of the transmitted information and the net cycle time as the maximum number of pre-selected platforms (M) is varied from 1 to 3. The x-axis represents the diverse simulation scenarios (runs) with different starting positions and dynamics of the targets. The baseline case of R^2 reporting is represented as $M = 0$ and corresponds to the lowest information flow with the minimum Network Cycle Time. As the value of M increases from 1 to 3, more platforms are selected to transmit non – R^2 track data and the information flow in the network increases. However this increased situational awareness comes at the cost of information latency, as the Network Cycle Time (NCT) increases with M. This represents an intuitive result of the tradeoff between information content and information latency. Increasing the value of M allows additional track data to be transmitted over the tactical data network, though it also results in increased latency between successive track updates.

3.4.7.2 Scoring Rules

In order to facilitate the discussion on the comparison of the four strictly proper scoring rules – Quadratic, Spherical, Logarithmic and Parametric - we generate the plots of the total expected payment and the minimum payment made by the center for the parameter space of $k = (1, 3)$. We present these results in Figure 22.

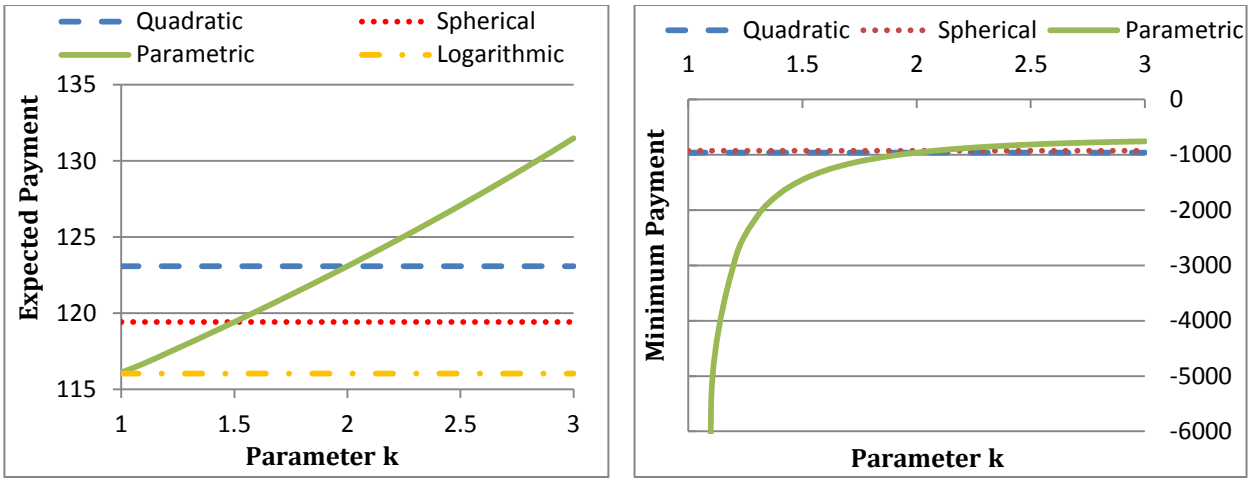


Figure 22: (a) Expected Payment for different values scoring rules, and (b) Minimum Payment for different values scoring rules

Figure 22(a) illustrates that the payment scheme based on the logarithmic scoring rule and the parametric scoring rule for $k \to 1$ results in the center making the lowest expected payments to the sensor platforms. Another distinctive trait is that the expected payment resulting from the logarithmic, spherical and quadratic scoring rules is the same as those based on the parametric scoring rule, for values of the parameter $k \to 1$, $k = 1.5$ and $k = 2$ respectively. This result serves as a validation for the analytical derivation where the parametric scoring rule takes the same expression for the expected payments as the logarithmic, spherical and quadratic scoring rules, for values of the parameter $k \to 1, k = 1.5\ and\ k = 2$ respectively. Figure 22(b) plots the lower bounds of the payment of the parametric scoring rule for the parameter space $k = [1.1, 3)$ against the spherical and quadratic scoring rule. The logarithmic scoring rule and the limiting case of the parameter scoring rule family $(k \to 1)$ results in large negative payments when the sensor platforms produce imprecise observations and hence are omitted from the figure. The effect of the platform's imprecise estimate can be minimized for the parametric family by choosing the parameter carefully. From Figure 22 it appears that a value of $k \in [1.1, 1.5]$ is a judicious compromise between the different factors. This set of parameter values produces low expected payments close to the ones obtained from the logarithmic scoring rule, and at the same time, imposes a finite lower bound on the minimum payments.

3.4.7.2 Protection Level (Γ)

Next, we solve the robust portfolio optimization problem in the second stage by using the Bertsimas - Sim formulation for different values of the protection level (Γ). Figure 23(a) shows the decrease in the expected information flow and the uncertainty-adjusted information flow in the mechanism as the value of Γ increases. The uncertainty-adjusted information value represents the difference between the expected information flow and the risk function

(uncertainty) when at most Γ variables are allowed to take their worst values. The figure illustrates the phase transitions that occur as the value of Γ increase and the transition points for the expected information flow coincides with the protection levels where the composition of the portfolio changes. The Bertsimas-Sim framework provides probabilistic bounds of constraint violation, i.e. a theoretical bound on the fraction of portfolios with information values which fall below the threshold value of the uncertainty adjusted information. We plot this probability of underperforming as a function of the protection level Γ in Figure 23(b). For low protection levels, the probability of the portfolio solution falling below the optimal solution is quite high. As the protection levels increase, probability of underperforming decreases by several orders of magnitude.

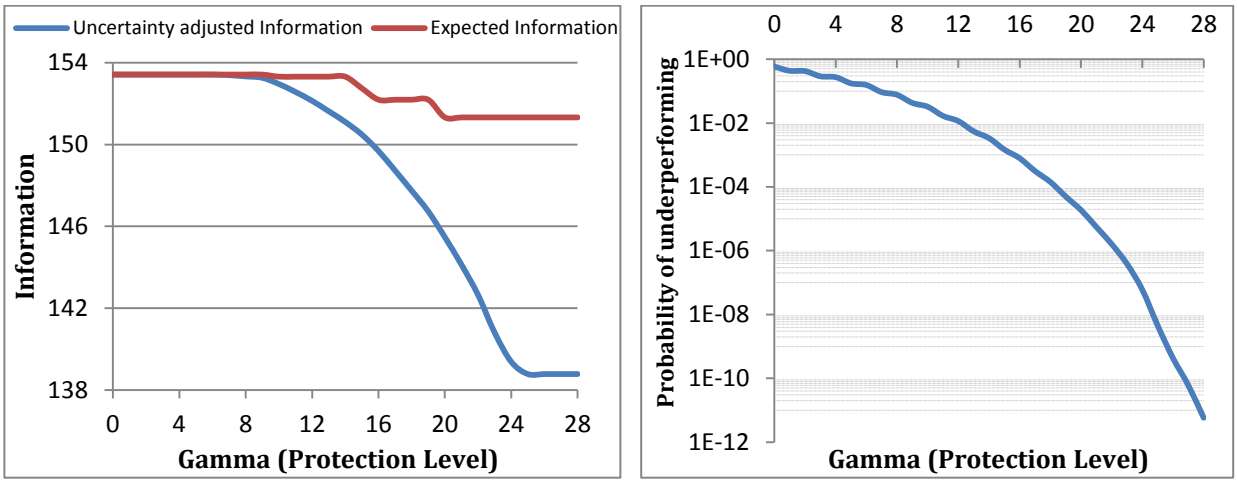


Figure 23: (a) Information vs. Protection level, and (b) Probability of Underperforming vs. Protection level

3.4.8 Conclusions and Future Work

In conclusion, we have successfully applied our proposed robust-optimization based scoring rules algorithm to the MAS simulation model. The algorithm provides a unique insight into the role of computational mechanism design, especially strictly proper scoring rules, in decision making. The applicability of the proposed mechanism goes beyond tactical data links and is amenable to any settings which involve exchange of information or services between buyers and sellers. Ensuring trust in the auctioneer, handling the correlation in data uncertainty and preventing bidder collusion represent some of the potential avenues for extending the scope of this research work.

3.5 Functional Dependency Network Analysis (FDNA) and Development Dependency Network Analysis (DDNA)

When complex systems and systems-of-systems (SoS) are involved, the behavior of the whole entity is not only due to that of the individual systems, but also to the interactions and interdependencies between the systems. Classical systems engineering approach is not always suitable to manage such feature, and new tools and methods are required, capable to identify, analyze and quantify properties of the system-of-systems as a whole. This research addresses the need to deal with complex dependencies between systems, in both developmental and functional relationships.

We propose a combination of two methods to analyze functional and developmental dependencies between systems in a system-of-systems, and to assess the impact of such dependencies on metrics that characterize global properties of a system-of-systems over its life span. The analysis can be used to drive decisions for system-of-systems design, architecture, and evolution, with respect to the identified metrics of interest. It also accounts for the presence of multiple stakeholders, and external factors that influence the operability and the development of a system-of-systems.

The methods support the analysis of trade-offs between competing features of a SoS and facilitates identification of better performing architectures. We show preliminary results of the application of the methods, and how the results can be interpreted to evaluate system-of-system properties on synthetic problems.

3.5.1 Introduction

The efforts in architecting a conglomeration of systems or a system-of-systems have met many difficulties due to the size and complexity of the underlying problem involved. In a system-of-systems, the constituent systems have, at least in part, operational and managerial independence (Maier, 1998. Sage, and Cuppan, 2001). Furthermore, the behavior of the whole entity is not only due to that of the individual systems, but also to the interactions and interdependencies between the systems (Mane, DeLaurentis, and Frazho, 2011. Hsu, 2009. Nai Fovino, and Masera, 2006). Classical systems engineering approach needs to be supported by innovative perspective and methods capable to handle the features that characterize a system-of-systems (Keating et al., 2006), and to analyze and quantify properties of the system as a whole, and during its development (Dahmann, and Smith, 2012).

In system-of-systems engineering, the first step required to perform the desired analysis involves determining and quantifying metrics that describe the features of a system-of-systems. Metrics at the individual systems level do not directly translate to the system-of-systems level. Many authors recognize the importance of this system-of-systems - level metrics, or *ilities* (Rhodes et al., 2009. De Weck et al., 2012), and acknowledge the need to include these metrics in the process of designing, architecting, and planning updates of systems-of-systems. There is

however a lack of methods that can effectively identify good designs with respect to preferred metrics, and drive decisions based on the trade-off between these qualities (Beesemyer et al., 2012).

To address these limitations, we propose a combination of updated versions of two previously developed methods (Guariniello, and DeLaurentis, 2013), to analyze functional and developmental dependencies between systems in a system-of-systems, and to assess the impact of such dependencies on *ilities*. We model the systems-of-systems as dependency networks, where nodes represent the component systems and the capabilities that the system-of-systems has to achieve. In functional dependency networks, the edges represent the operational dependencies between systems. In developmental dependency networks, the edges represent the developmental dependencies between systems. A functional dependency means that a certain system needs input (data, material, and energy) from another system in order to reach its full operability. A developmental dependency means that the development of a certain system is dependent from the full or partial development of another, but this dependency not necessarily affects its functioning.

The dependencies between systems are characterized by *strength* and *criticality.* Strength quantifies how much the behavior of a system depends on the behavior of another system. Criticality quantifies the negative impact that a system has on another, in critical conditions. These features give insight into the importance of the dependencies and we use them to quantify the impact of such dependencies on the overall behavior. The goal of the research is to quantify various metrics of interest using a combination of the functional and developmental dependency analysis methods. We propose to use results of this analysis to guide decision in system-of-systems engineering.

In functional dependency analysis, we compute the operability of each system as a function of the operability of the other systems in the network, based on the topology of the network and on the features of the dependencies. In developmental dependency analysis, similar considerations result in the evaluation of the impact of partial dependencies, stakeholders decisions, and development delays on the development of the entire system-of-systems. The representation of a system-of-systems as a network prevents the method from being domain dependent and allows for its application across various classes of problems. In some of the case studies (the *on-orbit servicing SoS*), we use only functional dependency analysis. In other problems (the *littoral warfare SoS*, and the *Mars exploration SoS*), outputs from the developmental dependency analysis result in a schedule for the development of the system-of-systems, and we feed the partial architecture achieved during development into the functional dependency analysis tool, in order to evaluate the partial operability and the partial capabilities achieved by the system-of-systems over time. This results lead to the evaluation of metrics, such as robustness and reliability of the whole system-of-systems, i.e. its capability to maintain an adequate level of operability during development, following degradation and partial failures. We can compare different architectures based on their flexibility, which is the capability to adapt to delays and external decisions. Hence, the combined application of the methods can be used to guide decision both in architecting the system-of-systems and in planning updates and modifications. The methods identify better architectures with respect to the desired features, and support trade-off between competing *ilities* of the system-of-systems.

3.5.2 FUNDAMENTALS OF FUNCTIONAL DEPENDENCY NETWORK ANALYSIS

Functional Dependency Network Analysis (FDNA) is a method to analyze the result of possible cascading effect of interdependencies between systems on the overall operability, in case of disruptions. The method was originally formulated by Garvey and Pinto (Garvey, and Pinto, 2009. Garvey, and Pinto, 2012), who applied it to capability portfolio analysis and risk assessment. We modified FDNA to make it suitable to analyze interdependencies in SoS, and successfully applied to aerospace system-of-systems (Guariniello, and DeLaurentis, 2013 AIAA. Guariniello, and DeLaurentis, 2013 IAC). In this section, we summarize the basic ideas and formulation of FDNA.

In FDNA, we model the architecture of system-of-systems as a directed network (Fig. 24). The nodes represent either the component systems or the capability to be acquired. Accordingly, the links represent the operational dependencies between the systems or between the capabilities. Each dependency is characterized by strength (Strength of Dependency, SOD) and criticality (Criticality of Dependency, COD), that affect the behavior of the whole system-of-systems in different ways. Strength of dependency accounts for how much the behavior of a system depends on by the behavior of a predecessor system, while criticality of dependency quantifies how the functionality of a system degrades when a predecessor system is experiencing a major failure. Those inputs can come from expert judgment and evaluation, or we may compute them by simulation and experiments.

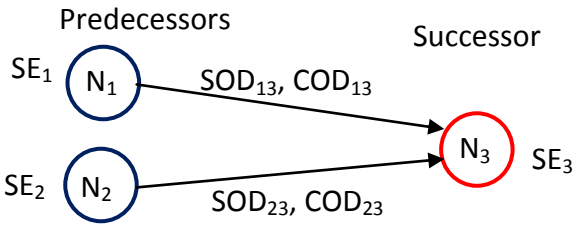


Figure 24: Synthetic FDNA network. N: node. SOD: strength of dependency. COD: criticality of dependency. SE: self-effectiveness.

This method is used to evaluate the effect of topology, and of possible degraded functioning of one or more systems on the operability of each system in the network. Differently from a Markov network approach, FDNA models the effect of disruptions on multiple dependent systems (rather than the probability to pass the disruption to one successor). Furthermore, FDNA models partial operability of the component systems, and can give better insight into the complex interactions between systems. Rather than being based on probabilities that a disruption propagates along the network, as in the Bayesian approach, FDNA assumes that a decrease in the operability of a system always affects all the dependent systems, with different impacts due to the features of the dependency. We can thus model more details of the interactions, that can result in a decrease of performance, instead than the total failure of a system. The analysis can be a deterministic evaluation of a single instance of the system-of-systems, or a stochastic quantification of the overall system-of-systems behavior. In the

deterministic analysis, given the *internal health status* (called Self-Effectiveness, SE) of each system, and the properties of the dependencies, FDNA quantifies the operability O_i of each system, based on equations (5.1) – (5.6). The operability of a node, ranging between 0 and 100, is defined as the "percentage" of effectiveness, that is the level at which the system is currently operating, or the level at which the desired capability is being currently achieved.

The operability of root nodes is equal to their self-effectiveness, since they are not dependent from other nodes:

$$O_i = SE_i \tag{5.1}$$

The operability of nodes that have at least one predecessor is computed as the minimum of two terms, one depending on the SODs, one depending on the CODs:

$$O_j = min(SOD\_O_j, COD\_O_j) \tag{5.2}$$

$$SOD\_O_j = \frac{1}{n}\sum_{i=1}^{n} SOD\_O_{ji} \tag{5.3}$$

$$SOD\_O_{ji} = SOD_{ij}O_i + (1 - SOD_{ij})SE_j \tag{5.4}$$

$$COD\_O_j = min(COD\_O_{j1}, COD\_O_{j2}, \dots, COD\_O_{jn}) \tag{5.5}$$

$$COD\_O_{ji} = O_i + COD_{ij} \tag{5.6}$$

In the stochastic version of FDNA, the self-effectiveness of each system follows a probability distribution. Consequently, also the operability of the nodes is probabilistic. In the previous studies, we proposed FDNA as a tool to identify the most critical nodes in the network, as well as the most important dependencies, in terms of impact on the operability when disruptions occur. In this study, we employ results from FDNA analysis to assess the impact of interdependency on possible metrics of interest used to quantify the goodness of a system-of-systems.

3.5.3 Fundamentals of Development Dependency Network Analysis

Development Dependency Network Analysis (DDNA) method, borrowing the concepts of SOD and COD from FDNA, is applied to development system-of-systems networks, where the links, like in PERT networks, represent development dependencies between systems. Differently from PERT, however, the dependencies are not absolute and account for partial independency of development of each system. In this section, we summarize the basic ideas and formulation of DDNA (complete description in Guariniello, and DeLaurentis, 2013 CSER). The outcome of DDNA is the beginning time and the completion time of the development of each system, as well as an assessment of the combined effect of multiple dependencies and possible delays in the development of predecessors. As in FDNA, this method evaluates the most critical nodes and dependencies with respect to development time and propagation of delays. We use results from the analysis to compare different architectures in terms of development time, capability to absorb delays, and flexibility.

The dependencies affect both the beginning time and the completion time of development of a system. Differently from PERT, development of a system can begin before a predecessor is complete, according to functions such as the parabolas in Fig. 25 (in this study, linear functions and other curves have been tested, to model different development dependencies. Inputs from experts will suggest the appropriate function to use to model the development dependency between the systems, given the specific problem).

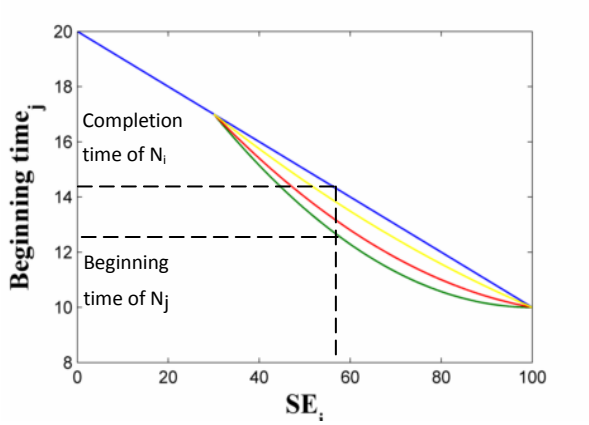
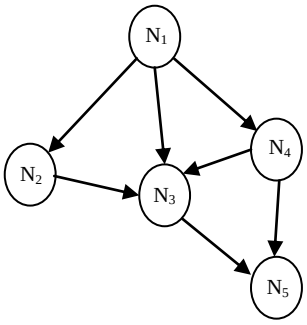


Figure 25: The dependency between node N_i and node N_j. If SEi is critical, the beginning time of N_j coincides with the completion time of N_i. Otherwise, the development of N_j can begin earlier: the straight line (PERT) relates SE_i to the completion time

Computation of the beginning and completion time for each node, results in a complete schedule for the development of the system-of-systems, showing the effect of partial development dependency on the development time. Fig. 26 shows a Gantt chart for a simple dependency of a system from two other systems, with development time computed with DDNA. The combined use of FDNA and DDNA allows to assess partial capabilities during the development of a system-of-systems.

(a)



(b)

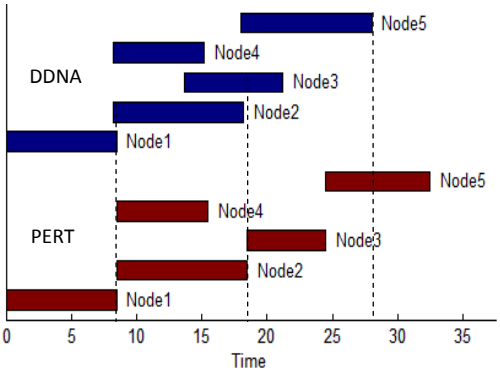


Figure 26: (a) Five-node development dependency network. (b) Gantt chart for the development of a five-node network. The dashed lines show the beginning of development of nodes 2 and 3 in PERT, and the completion of node 5 in DDNA

3.5.4 Illustrative Example and Results

In this section we show results of the application of FDNA and DDNA to the analysis of Systems-of-Systems. We also describe how various metrics of interest can be evaluated based on the results of dependency analysis.

3.5.4.1 Littoral Combat Warfare

A littoral combat warfare system-of-systems, comprised of ships, helicopters, UAV, and USV, and used to detect and engage enemy boats, mines, and submarines, is shown in Fig. 27a. We analyzed it with FDNA and DDNA, and in this section we present preliminary results about metrics that are function of the systems dependencies. We show the functional dependency network in Fig. 27b.

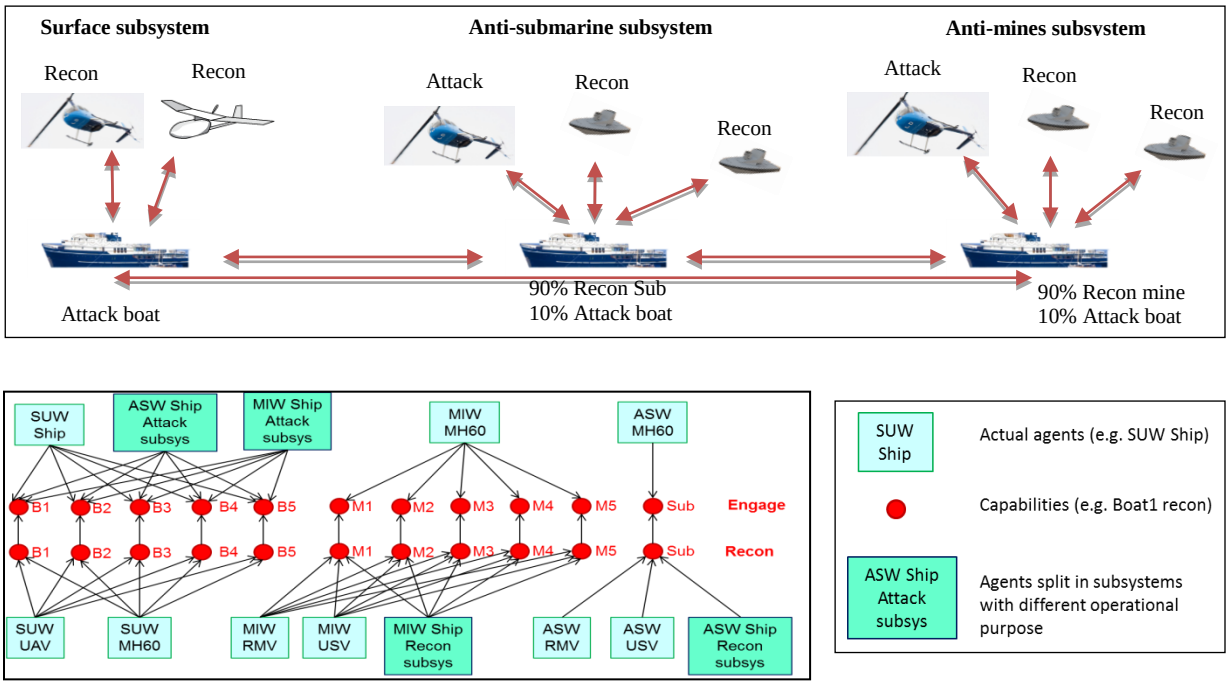


Figure 27: (a) Littoral combat warfare system-of-systems. Each friend agent perform different functions, as shown in the figure. (b) Functional dependency for the littoral combat warfare SoS.

We consider three architectures, characterized by different development networks. The development networks, whose features are summarized in Table 10, represent different approaches, where various stakeholders participate into the system-of-systems at different times. The development dependencies may be modeled based on technology readiness, cost consideration (more systems are developed and deployed based on funding), and efficacy of the deployed systems (more systems are developed based on the results achieved by the system-of-system under development, i.e. on the partial capabilities).

Table 10: Features of the three architectures considered for preliminary results.

Architecture	Development features	Operational features
A	Ships and Surface system developed first, followed by anti-marine system, and then by anti-mine	According to Fig. 26
B	Two ships developed first, then MIW MH60 and RMVs, then UAV and USV, finally anti-submarine	According to Fig. 26
C	Two ships developed first, then MIW MH60 and RMVs, then UAV and USV, finally anti-submarine	MIW MH60 and ASW MH60 can attack both mines and submarine

Partial capabilities and robustness

We compare different development architectures, based on FDNA analysis of the system-of-systems under development. As the system-of-systems is developed, and systems are deployed, the entire network gains partial capability to detect and engage the enemy. Fig. 28 shows a comparison of the architectures, with respect to the capability over time to engage enemy units. The "steps" in operability correspond to deployment of new systems.

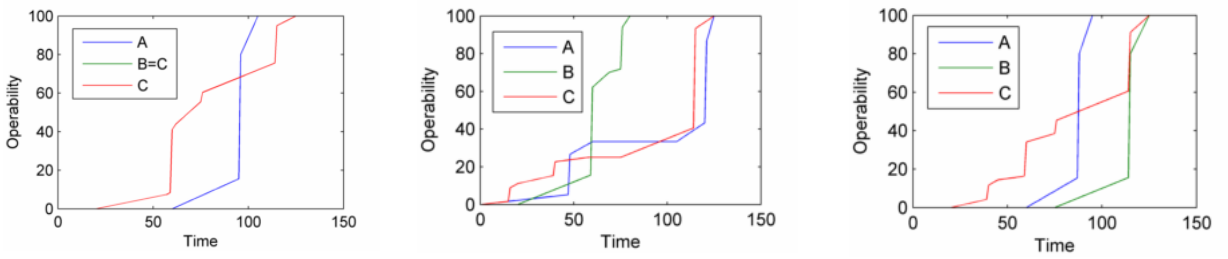
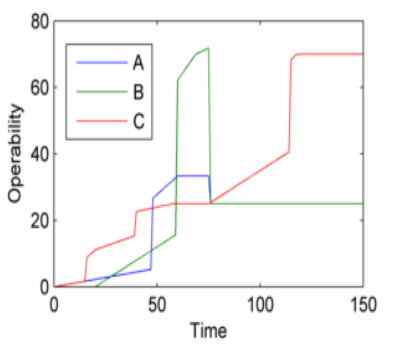


Figure 28: Time in weeks. (a) Capability to engage enemy boats. (b) Capability to engage enemy mines. (c) Capability to engage enemy submarines.

Architecture A reaches the capability to engage boats later than B and C, but it is capable to engage submarines faster than C. Architecture B reaches the capability to engage mines earlier than the other two. Architecture C, where the helicopters in the anti-mines and anti-submarines subsystems can help each other, can achieve partial capability of submarine engagement earlier. We can perform similar comparison in case of delays or loss of units. In Fig. 28, we show the results of an evaluation of the robustness of the architectures when operative loss of a unit occurs. The SoS does not reach full operability, but in case of loss of an MH60, architecture C is more robust to the failure (Fig. 29a). All the architectures are robust with respect to engaging boats in case of loss of a RMV (Fig. 29b).

(a)



(b)

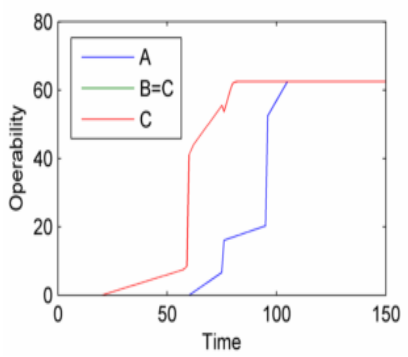


Figure 29: Robustness of the architectures. Capability to engage: (a) enemy mines when MH60 is lost; (b) enemy boats when RMV is lost

Resilience

When disruptions occur, the system-of-systems is partially able to recover the loss in operability, thanks to the interdependencies and to the complex architecture, than may allow systems to be re-tasked help each other and share part of their capability. Fig. 30 shows the results of the same loss as in Fig. 29a, if we suppose that in architectures A and B the RMV can switch their task from detection to engagement after the failure of the helicopter.

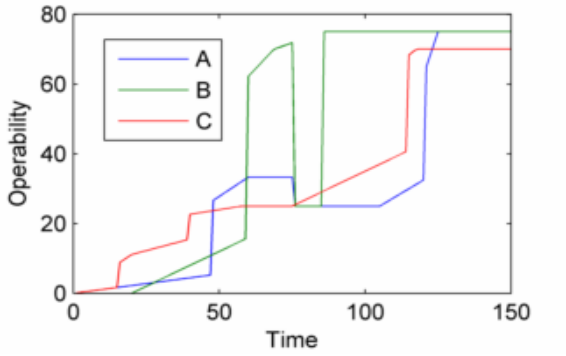


Figure 30: Operability of the anti-mines subsystem. Resilience of architectures A and B when an MH60 is lost. Comparison with Fig. 29a shows the increased resilience of the architectures, due to the capability to re-task systems in the littoral combat waters

Flexibility

During the development of the system-of-systems, stakeholders may modify their decisions, or the objectives of the complex system can change. The property that allows the system-of-systems to react to these changes is the flexibility. Differently from the resilience, which is a property that arises from the capability of the systems to adapt and be re-tasked, flexibility involves changes in the development itself. For this reason, if the development of the system

involved is already in an advanced phase, flexibility is limited. The example in Fig. 31 shows the flexibility of architectures A and B, in a scenario where a stakeholder decides to withdraw its participation in the development of part of the anti-mines subsystem. Other systems can be adapted to replace the missing one, only if their development is at most in its early stages. In the example, architecture A manages to achieve some partial capability that was missing in the case of re-tasking after development. However, both architectures A and B do not show the same recovery as in the previous example, due to the advanced development of the systems involved (in this case, no re-tasking is allowed for systems that are already deployed).

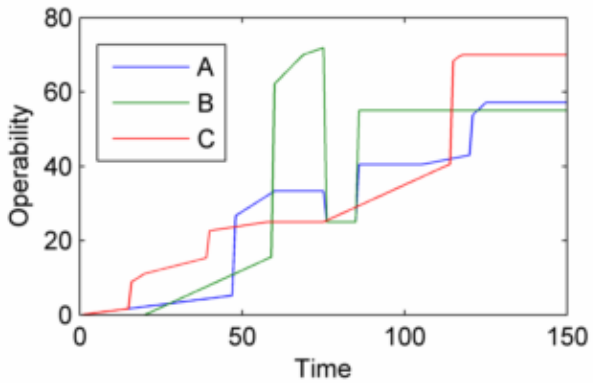


Figure 31: Operability of the anti-mines subsystem. Flexibility of architectures A and B when a stakeholder withdraws its participation.

Summary

Whereas we do not include cost analysis, and a complete investigation on the properties of the architectures, that should also account for probability of disruptions, delays, and stakeholder decisions, in table 11 we show a quick summary of the preliminary results presented in the previous sections.

Table 11: Features of the three architectures considered for preliminary results.

Architecture	Partial capabilities	Robustness	Resilience	Flexibility
A	Low for boats, medium for mines and submarine	Low	Medium-high	Medium
B	Medium-high for boats and mines, low for submarine	Low	High	Medium-low
C	High for boats and submarines, medium for mines	High	Not-used	Not used

3.5.4.2 On-orbit servicing

We analyze an on-orbit servicing satellites team that is a group of satellites able to perform inspection, refueling, maintenance. Such satellites, equipped with fuel and replacement modules, can perform more than one on-demand servicing operation, thus introducing

flexibility. This flexibility results in lower cost than purpose-launched satellites. The entire architecture is characterized as a two-level System-of-Systems (Fig. 32): the higher level is constituted by the servicing satellites, the modular satellites, and their operational interdependencies. The lower level describes the inner architecture of the modular satellites, i.e. the modules and their operational interdependencies. These satellites have their own objectives, and may be independent from the other satellites or may be part of a constellation, but we want to analyze the entire set of both modular and servicing satellites, that constitutes a SoS. FDNA analyzes the effects of interdependencies on operability over time, and identifies metrics to assess and measure the system value, like robustness, criticality, flexibility. Thanks to the System-of-Systems representation, the analysis is conducted at different levels: in the lower level, the functional dependencies between the component modules of a single satellite are accounted for. Degradation of the modules over time, as well as major disruptions, are then considered, and different architectures and patterns can be compared based on the cost and the level of partial operability still achievable after the disruptions. When the operability of a satellite is lower than a given threshold, maintenance can be requested. The global operability level is hence dependent not only on the operability of each satellite, but also on the architecture and the dependency features of the entire System-of-Systems (i.e., the possibility to obtain servicing). Analysis of the upper level is then performed, using the results from the lower level analysis, to evaluate and compare different architectures of the whole set.

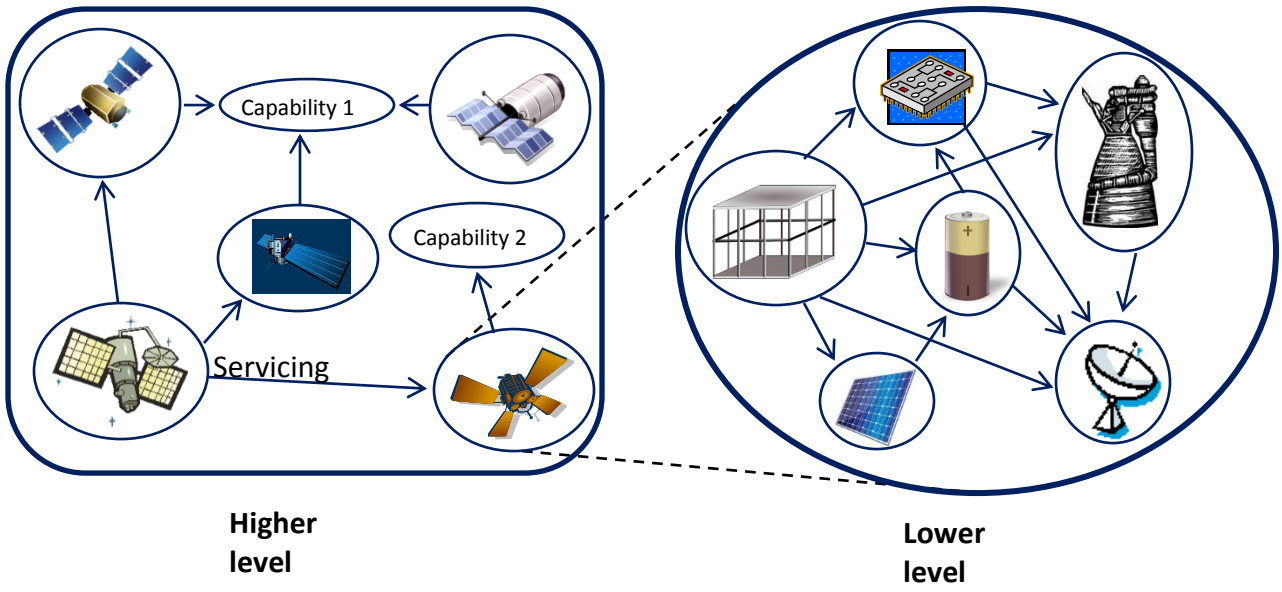


Figure 32: Two-level functional dependency representation of the on-orbit servicing System-of-Systems

Lower level

At the lower level, this study considers modular satellites, in which a structure contains modules that are supposed to be easily accessible to be replaced or maintained. Simplifying assumptions about the level of detail can be relaxed, and more detailed and realistic models can be used, in order to refine the results.

The satellites are divided into three groups: Communication Satellites, Observation Satellites, Experimental Satellites. Some of the modules are present in all the satellites, other modules are specific to a certain type of satellite. For a few of the systems (power, thrusters) alternative choices can be made. Table 12 shows the main properties of the satellites in the model used in this study, as well as type and number of modules that constitute the satellite. Table 13 lists the alternatives for the systems for which alternative choices are available.

Table 12. Properties and modules of the satellites. Systems for which alternative choices are available, are underlined

	Communication Satellites	Observation Satellites	Experimental Satellites
Orbit	Geostationary (GEO), Molnyia, Tundra	Low Earth Orbit (LEO), Medium Earth Orbit (MEO)	GEO, LEO, MEO
Structure	1	1	1
Power Controller	1	1	1
Power Source and Storage	X	X	X
Power Regulators	2-3	2-3	2-3
Communication / Main Software	1	1	1
Transponder and Gyros	1	1	1
Guidance, Navigation and Control (GNC) System	1	1	1
Thrusters	4-6-8	4-6-8	4-6-8
Payload	X	X	X

Table 13: Systems for which alternatives are modeled, and properties of the alternatives.

	Alternatives	Properties
Orbit	GEO	Geostationary orbit: circular, inclination = 0°, radius = 42164.140 km
	Molnyia	Semigeosynchronous orbit, inclination = 63.4°, semimajor axis = 26561.744 km
	Tundra	Geosynchronous orbit, inclination = 63.4°, semimajor axis = 42164.140 km
	LEO	Circular, inclination = any, with preference of polar orbits for observation satellites, radius = 6578 - 7378 km (height = 200-1000 km)
	MEO	Circular, inclination = any, with preference of polar orbits for observation satellites, radius = 8378 - 11378 km (height = 2000-5000 km)
Power Source and Storage	Fuel Cells	1-2 fuel cells. No recharge, no batteries
	Solar panels on the surface + batteries	Surface of the satellite covered with solar panels, 1-3 batteries
	Deployed solar arrays + batteries	1-2 solar arrays, 1-3 batteries
Payload	Communication antennas	Between 2 and 6, only in communication satellites

	Sensors and data handling system	Between 3 and 5, only in observation satellites
	Experiments and data handling system	Between 1 and 3, only in experimental satellites

Lower level modeling and preliminary results

Satellites with different characteristics can be modeled and analyzed with FDNA at the lower lovel, to gain better insight into the effect of interdependencies, redundancy, various architectures. The architecture of a satellite can be specified by the user, or a random generator can be used: satellites are modeled, with a given probability for the choice of the type, and of the component modules. In this study, a random generator has been used, with the probabilities suggested by current practice. The generator can be easily modified according to the user's needs. Fig. 33 shows a communication satellite, created by the random generator: it is in Geostationary orbit, gets power from panels on its surface, and has two batteries, three power regulators, six thrusters and two antennas. The figure also gives an idea of the complexity of the functional interdependency between the modules.

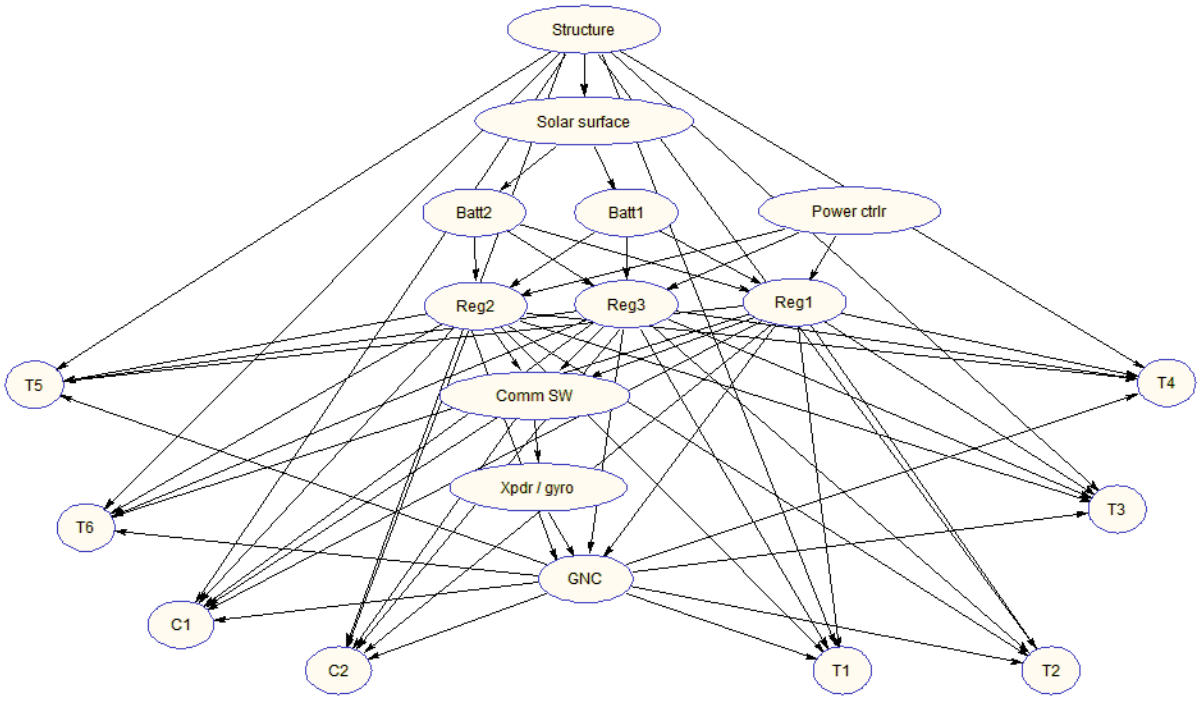


Figure 33: Functional Dependency Network of a communication Satellite. Ctrlr: controller. Reg: power regulator. Comm SW: communication software. Xpdr / gyro: transponder and gyroscopes. GNC: guidance, navigation and control. T: thruster. C: communicat

ir

f

itself, and from the correct operability of the recharge system. The dependency of an observation sensor will be critically dependent from the GNC system, while the criticality of the dependency from a power regulator will be low, if the satellite has three regulators. There are different ways to quantify such inputs: they can come from knowledge by experts, from simulation and Design of Experiments, from historical data analysis. In this conceptual study, values coming from experience have been used to characterize the interdependencies.

As for the self-effectiveness of the systems, a simple model of the evolution of this value for each module has been implemented. A timestep of one month is considered. Starting from the maximum level of operability, i.e. 100, each module experience a decrease in self-effectiveness due to three factors:

- Aging / wearing out / losses: modeled as a small, but always greater than zero, random loss in self-effectiveness.
- Degradation by minor failures: modeled with a beta distribution for the small loss in self-effectiveness.
- Major failures / accidents / catastrophic events: if the event occurs (there is a low probability), the loss in self-effectiveness is sampled from exponential distribution, and ranges between 70 and 100.

Fig. 34 shows some preliminary results from the analysis of an instance of the evolution of the same satellite modeled in fig. 33. The evolution of four modules is depicted in the graph: the power controller, solid electronic part inside the satellite, is mostly subject to some aging. The structure and the communication antenna, subject to weathering from the harsh space environment, show a steeper decrease in operability, and some "jump" due to minor disruptions (like meteorite hits, for example). The power regulator no. 3 experienced a major failure in this instance, and could be a candidate for on-orbit maintenance.

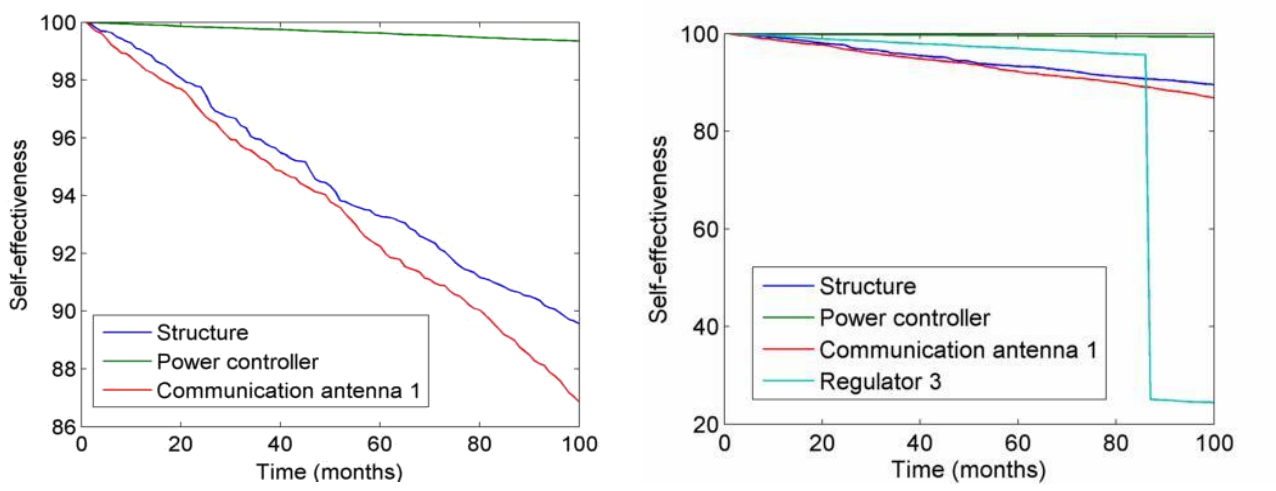


Figure 34: Self-effectiveness of modules over time. Left: self-effectiveness of Structure, Power controller, and Communication antenna. Right: the same modules as in the left figure, plus Regulator 3, which experiences a major failure.

The evolution of the self-effectiveness over time is used as input to the FDNA tool, to quantify the operability of each module over time. Fig. 35 reports the results of FDNA analysis for the same communication satellites as in the previous examples. First of all, we can notice that the structure operability is not affected by the drop in the operability of the regulator, as expected. Then, one of the positive impacts of interdependencies is quantified: even if the self-effectiveness of the regulator decreased to 25, its operability that depends on the batteries and

the power controller can be kept to a higher level. The drop in operability of the regulator has however an impact on the operability of the communication antenna and the gyros, even if this is not a critical dependency, thus the decrease is small. In the right part of Fig. 35, the same satellite has been analyzed, without full redundancy of 3 regulators. In this case, the dependency of the gyros from the regulator is more critical, resulting in a bigger loss of operability.

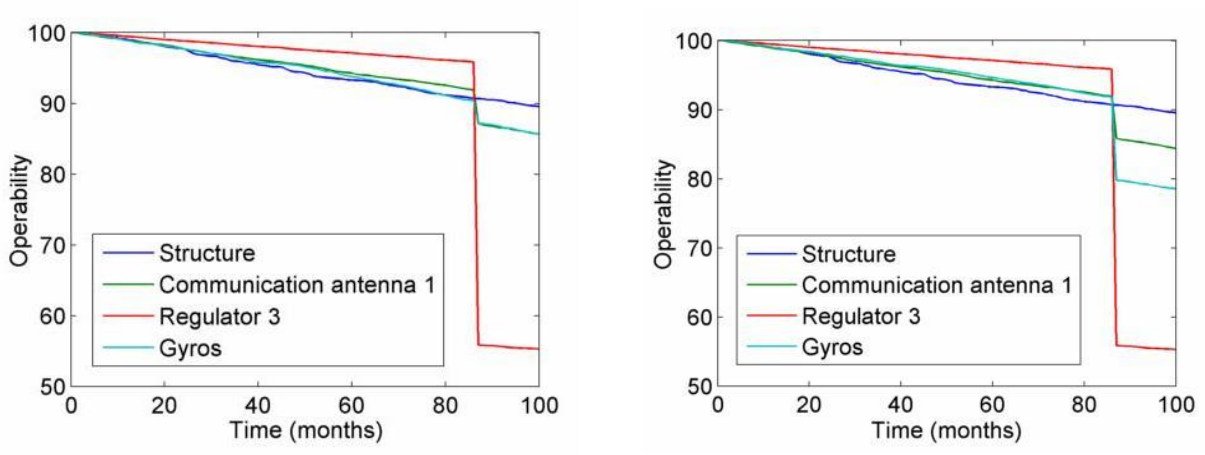


Figure 35: Operability of modules over time. Results come from Functional Dependency Network Analysis. Left: lower criticality of the regulator. Right: higher criticality of the regulator.

Higher level results

Results from the lower level are combined in the higher level, where the architecture of possible constellation of satellites is modeled: the overall operability, or other measures of interest, are quantified based on the operability of each satellite. For example, for a mission involving a tandem of observation satellites, we could be interested into the operability of each sensor on board of each satellite, or we could give more importance to the operability of a particular couple of sensors in the tandem. Nodes of interest are modeled as in the lower level, but at the higher level they are more likely to represent capabilities to achieve, rather than actual systems.

As shown in fig. 32, at the higher level also the architecture of on-orbit servicing is represented. For example, different analyses could involve the possibility for a satellite to perform servicing only to satellites in the same orbit. Also, a servicing satellite could carry spare parts for a sensor, a part needed only by observation satellites in our model.

As a preliminary example, two observation have been randomly generated. Both the satellites are in polar orbits (inclination of 87.26° and 91.94° respectively), the first one in MEO, with an orbit radius of 8493 km, the second one in LEO, with an orbit radius of 6850 km. The first satellite has two solar arrays, two batteries, two power regulators, eight thrusters, and four sensors. The second satellite has a fuel cell as power source, three regulators, four thrusters, and five sensors. An instance of evolution of self-effectiveness over time was simulated,

resulting in a major failure of thruster no. 5 of the first satellite after 57 months, and a major failure of the fuel cell in the second satellite after 47 months, followed by a major failure of sensor no. 3 after 95 months. While the failures in the first satellite had almost no impact, the failure in the fuel cell of the second satellite was critical to the operability of the sensors, though some operability of the fuel cell was still kept. We considered the satellites to be working in tandem, with the overall operability modeled, in a functional dependency network, as a function of the operability of each sensor and data handling systems of each satellite, and Fig. 36 shows the evolution over time of the operability at the higher level. Results show how, in spite of the major failure in a thruster, the first satellite is able to keep its sensors working with a high level of operability. The second satellite instead, relying on a major loss in the operability of its fuel cell, experiences a large decrease in the operability of its sensors (to about 55). The overall operability of the satellite tandem, affected by the degraded sensors of the second satellite, also decreases after 47 months. The critical failure in sensor 3 of the second satellite heavily affects the overall operability after 95 months. Servicing the second satellite by replacing the fuel cell, would keep the overall operability to a level above 93, for 95 months.

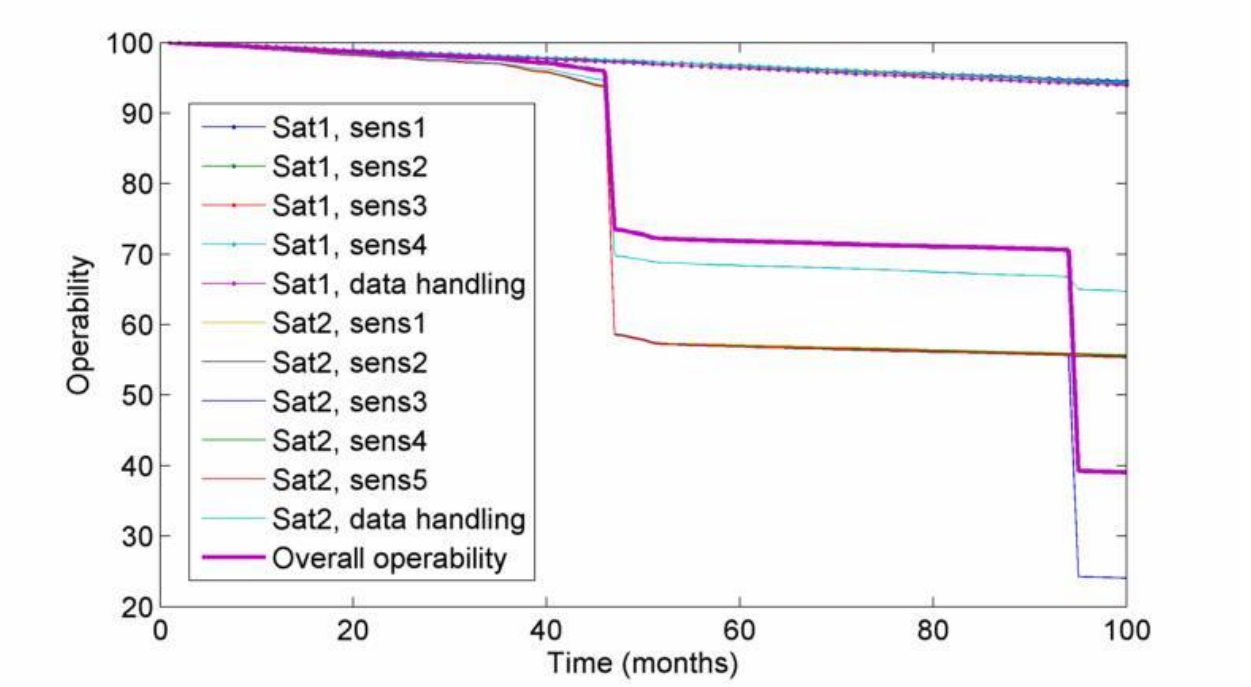


Figure 36: Higher level operability of a tandem of observation satellites in polar LEO. Critical failures in one of the two satellites affects the functioning of the overall System-of-Systems.

Simplifying assumptions

To achieve the preliminary results, and to perform the analysis of the case scenario presented in the following section, a few simplifying assumptions have been made for this study:

- servicing satellites are not subject to failures, so they are not decomposed into subsystems

- the cost analysis is simplified, accounting only for the Δv required for the maneuvers to reach the satellite to be serviced. More complex cost analysis can be performed
- in this paper, the proposed methods is used only for analysis, and not as a tool to guide decision in architecture and design of the on-orbit servicing SoS
- the presence of different stakeholders is not accounted for: the servicing satellites that are able to perform the required servicing, always perform it
- servicing is always effective.

Case Scenario Analysis and Results

A case scenario, including 30 operational satellites, was generated through a random generator. The main features of the 30 satellites are listed in table 14. The number of modules for each satellite ranges between 17 and 25.

Table 14: Operational satellites in the case scenario.

Sat. number	Type	Orbit	Number of payloads (sensors, antennas, experiments)
1	Observation	MEO	3
2	Observation	MEO	3
3	Observation	MEO	3
4	Communication	Molnyia	2
5	Experimental	LEO	1
6	Observation	LEO	4
7	Communication	GEO	2
8	Communication	GEO	2
9	Communication	GEO	6
10	Communication	GEO	2
11	Experimental	MEO	2
12	Experimental	LEO	1
13	Observation	LEO	4
14	Observation	MEO	3
15	Communication	Tundra	4
16	Experimental	LEO	3
17	Observation	LEO	4
18	Communication	Tundra	2
19	Experimental	MEO	1
20	Observation	LEO	4
21	Experimental	LEO	3
22	Experimental	LEO	2
23	Communication	GEO	3
24	Observation	LEO	3
25	Observation	MEO	5
26	Communication	GEO	6
27	Experimental	LEO	3
28	Observation	MEO	4
29	Observation	MEO	4
30	Observation	LEO	4

Based on their orbits and objectives, the satellites have been arbitrarily gathered in ten missions / constellations, described in table 15, that constitute part of the functional dependency network at the higher level.

Table 15: Missions / Constellations at the higher level in the case scenario

Mission / Constellation number	Type	Orbit	Component satellites
1	Observation	MEO	1
2	Communication	Molnyia	4
3	Communication	Tundra	15, 18
4	Communication	GEO	7, 8, 9, 10, 23, 26
5	Experimental	LEO	5, 12, 16, 21, 22
6	Experimental	LEO	27
7	Experimental	MEO	11, 19
8	Observation	MEO	2, 3, 14, 25
9	Observation	MEO	28, 29
10	Observation	LEO	6, 13, 17, 20, 24, 30

Analysis of a scenario with a single satellite

In this section, we describe and discuss results of the analysis of the operability and servicing of a single operational satellite (No. 1). The satellite is composed of 21 modules. At the higher level, the overall capability is functionally dependent on the three sensors, and the data handling system. This overall capability constitutes the measure of merit of the entire system, and is used to request servicing when its operability decreases below a threshold set at 70.

Since the evolution of the self-effectiveness of each module is probabilistic, a Monte Carlo simulation has been performed, analyzing 1000 instances of the satellite evolution over 100 months. Fig. 37 shows the expected value for the overall capability at each timestep, and the "best" and "worst" cases, in a scenario where on-orbit servicing is not available. The "best" and "worst" cases are defined as the instances with respectively the highest and the lowest average of the overall capability over time.

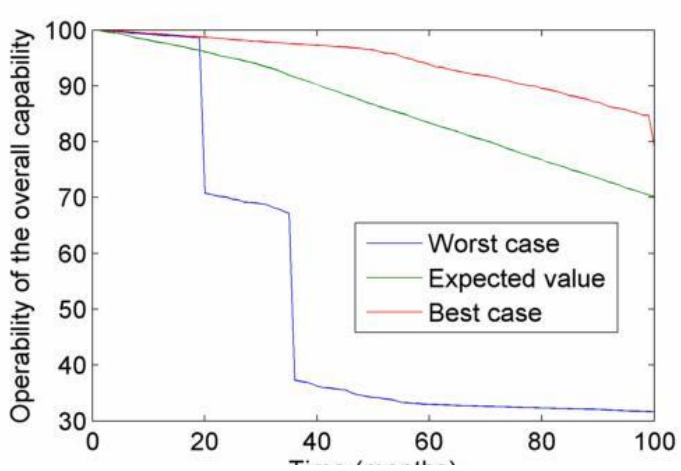


Figure 37: Overall capability of mission No. 1 (satellite No. 1) when on-orbit servicing is not available.

Without servicing, the expected value for the overall capabilities keeps always above 70. It must be noted, however, that in 369 instances out of 1000, this value dropped under the threshold of 70, that means that the satellite would require on-orbit servicing.

In fig. 38, the same results from Fig. 37 are compared with the same scenario, when on-orbit servicing is supposed to be always available for the satellite constituting the mission No. 1. The analysis shows that the expected value in this case keeps above 80. In what was the worst case in the scenario without servicing, the satellite is now serviced, and the operability of the whole mission is thus kept above the desired threshold. Using the same definition as we did before for "worst" case, we notice that when servicing is available, a different instance becomes the "worst" case.

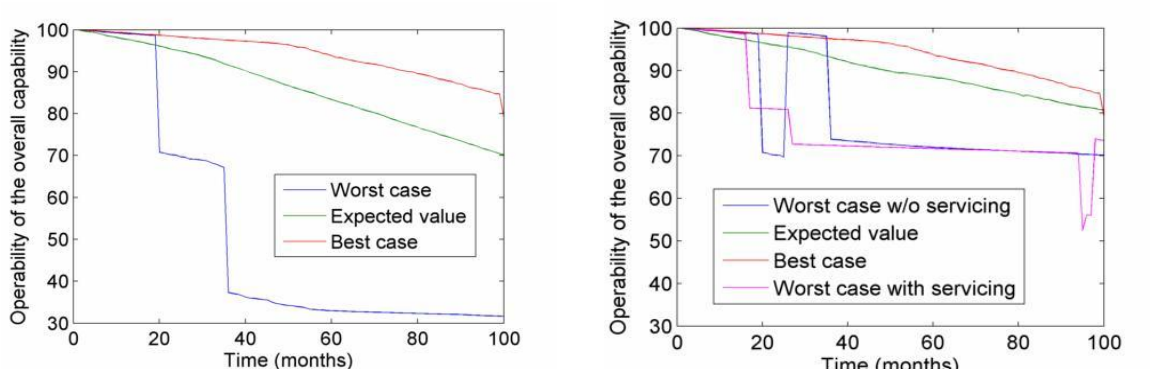


Figure 38: Left: mission No. 1 without servicing. Right: mission No. 1 with servicing.

Analysis of a scenario with all ten missions, without servicing

In this section, all ten missions listed in table 16 are analyzed. 1000 instances of the evolution over time of the operability of each module of the satellites described in tables 15 and 16 have been run. Fig. 39 shows the expected value of the overall operability of each of the ten missions considered in this scenario.

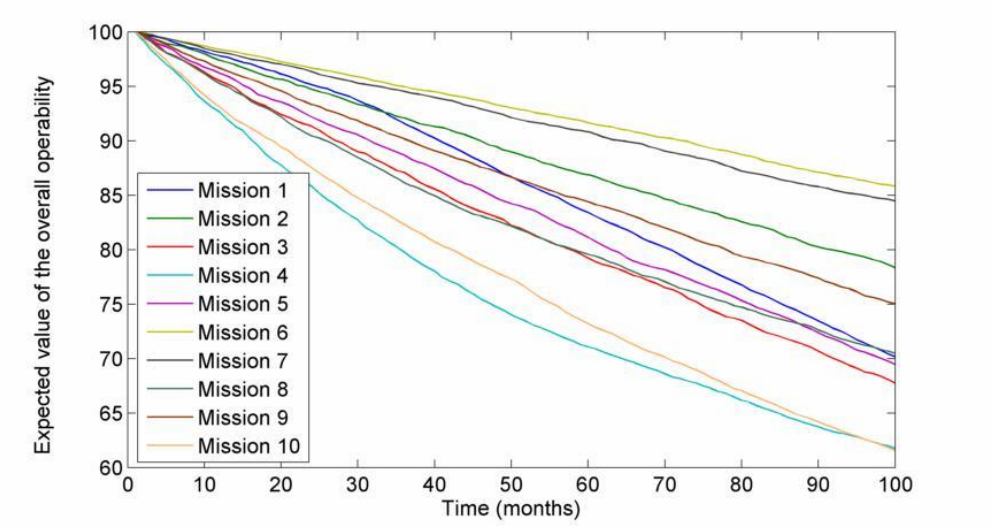


Figure 39: Expected value of the overall operability of the ten mission in the case scenario, when servicing is not available.

Fig. 40 shows the overall operability in the worst case for each mission. Since servicing is not available, some of the instances are subject to large decrease in the operability of interest.

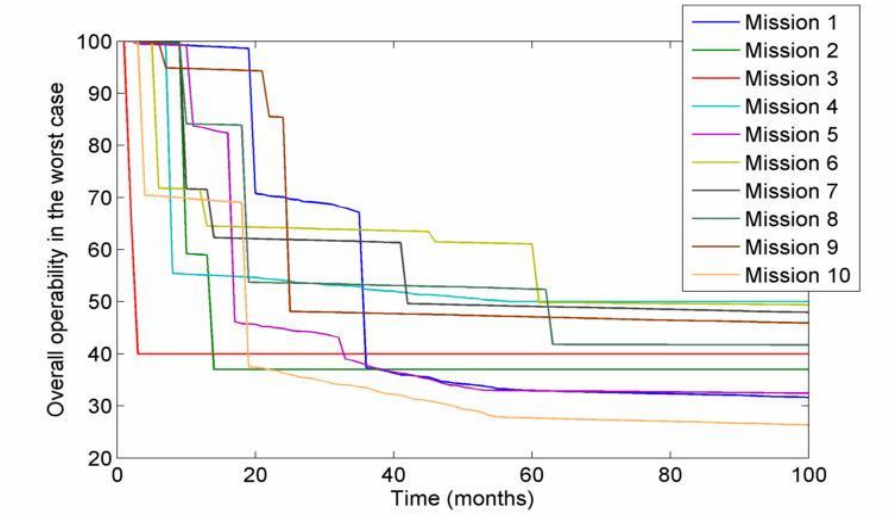


Figure 40: Evolution over time of the operability of the overall capability for each mission, in the worst case.

Table 16 lists the percentage of instances of each mission that would require at least one servicing operation. Since in this model most of the satellites are considered critical to achieve the desired overall capability, missions and constellations involving a larger number of satellites are more susceptible to need servicing.

Table 16: Percentage of instances requiring servicing.

Mission	1	2	3	4	5	6	7	8	9	10
Percentage of instances requiring servicing	36.9%	27%	56.4%	91.7%	58.3%	15.7%	22%	55.9%	39.8%	79.4%

Analysis of a scenario with all ten missions, with servicing

The same scenario analyzed in the previous section has been simulated again, with different architectures for servicing: a variable number of servicing satellites, in different orbits, and with different spare modules onboard, have been tested. Given an architecture, for each instance and at each timestep, if a mission overall operability decreases under the threshold of 70, the satellite causing the major loss can request servicing. If a satellite with the appropriate spare part is available, servicing is executed. The required Δv for the maneuver is computed, and after the operation, the servicing satellite will have the same orbit as the serviced satellite, for computation in the following timesteps. The spare part used for the servicing will not be available anymore in following timesteps of the same instance. The total number of requests, and the number of answered requests, are saved, allowing for the computation of the percentage of requests which were satisfied.

Table 17 lists results for some of the architectures tested. Given the long period (100 months), and the high threshold for the operability, easily reachable just by aging, the number of requests of servicing is quite high. The analysis show the efficacy of the on-orbit servicing, resulting in an increase both in the expected value of the overall capability, and in the overall capability of the worst case. The Δv is reported in the table as the average over the instances of a given architecture, but it can also be analyzed in trade-off with the gain in capability. The average percentage of requests satisfied does not depend in the case from the location of the servicing satellites (that instead influences the Δv). A version of the analysis allowing a satellite to be serviced only by satellites having the same orbit type has been tested, with similar results. In that case, however, the percentage of satisfied requests decreases, because the required spare parts could not be available. On the other side, this causes a reduction in required Δv, but also a lower increase in the expected value of the overall capability. As aforementioned, this analysis leads the way to more complex evaluations, and can guide decision-making in design and architecture of on-orbit servicing System-of-Systems.

Table 17: Results of the analysis of different architectures for on orbit-servicing in the modeled scenario (30 operational satellites, 10 missions). For each architecture, 1000 instances of 100 timesteps each have been run.

Servicing Architecture (number of satellites, orbit, [spare parts])	Average number of request (100 months)	Average percentage of satisfied requests	Average Δv	Average increase in expected value of the overall capability	Average increase in overall capability of the worst case
1 servicing satellite, LEO, [1 Sensor, 1 Battery, 2 Fuel Cells]	181.33	1%	8.21 km/s	0.826	9.14
1 servicing satellite, MEO, [2 Solar Arrays, 1 Power Regulator, 2 Antennas]	168.44	1.55%	9.31 km/s	1.334	0.43 (not serviced in most cases)
1 servicing satellite, GEO, [1 Solar Array, 2 Batteries, 1 Antenna, 1 Sensor]	158.46	1.87%	11.16 km/s	1.520	3.42
3 servicing satellites, LEO, [3 Fuel Cells, 1 Electronics, 1 Gyro, 3 Batteries, 1 Power Regulator, 3 Antennas, 1 Sensor]	124.41	5.59%	24.99 km/s	2.355	12.68
3 servicing satellites, MEO, [1 Solar Array, 2 Power Regulators, 3 Sensors, 1 Antenna, 2 Fuel Cells, 2 Batteries, 1 Electronics]	118.30	6.07%	26.35 km/s	2.294	13.40
3 servicing satellites, GEO, [3 Solar Arrays, 2 Batteries, 3 Fuel Cells, 2 Antennas, 3 Sensors, 2 Electronics]	124.29	5.33%	20.38 km/s	2.393	15.656
3 servicing satellites, LEO, [2 Batteries, 1 Sensor, 1 Antenna, 1 Fuel Cell], MEO, [1 Solar Array, 1 Antenna, 1 Sensor, 1 Electronics], GEO, [1 Antenna, 1 Solar Array, 1 Power Regulator, 1 Battery, 1 Fuel cell]	115.39	6.78%	27.44 km/s	2.544	13.39
6 servicing satellites, 2 LEO, [2 Sensors, 2 Fuel Cells, 1 Electronics, 1 Antenna, 1 Power Regulator, 1 Solar	107.55	8.53%	30.63 km/s	2.68	16.087

Array], 2 MEO, [2 Batteries, 2 Fuel Cells, 2 Electronics, 2 Power Regulators, 1 Sensor], 2 GEO, [2 Antennas, 3 Sensors, 2 Solar Arrays, 1 Power Regulator, 1 Battery, 1 Fuel Cell]					

3.5.4.3 Mars Exploration

The combined use of FDNA and DDNA allows to quantify the partial capabilities that can be achieved during the development of the System-of-Systems. When other considerations are added, such as delays and critical impact of nodes, architectures can be compared based on the trade-off between the evolution of their features over time and partial capabilities.

Consider the combined FDNA-DDNA space System-of-Systems network in Fig. 41. The light edges are functional dependencies, while the bold edges are development dependencies.

We are interested into three capabilities: Mars exploration, observation, and colonization. In this simplified model, the number of systems is small, but the problem can be easily scoped up. Considerations about the independent development by different stakeholders, affecting DDNA values, and about failures and flexibility, affecting FDNA, can be added according to the user's need.

Fig. 42 shows the evolution over time of the operability of the three required capabilities, due to the development of the systems in the SoS. In this case, there are neither failures, nor delays. As expected, the observation, that requires fewer systems and less complexity, reaches its full operability in a short amount of time. Colonization gradually reaches level of partial operability, following the development of the required systems.

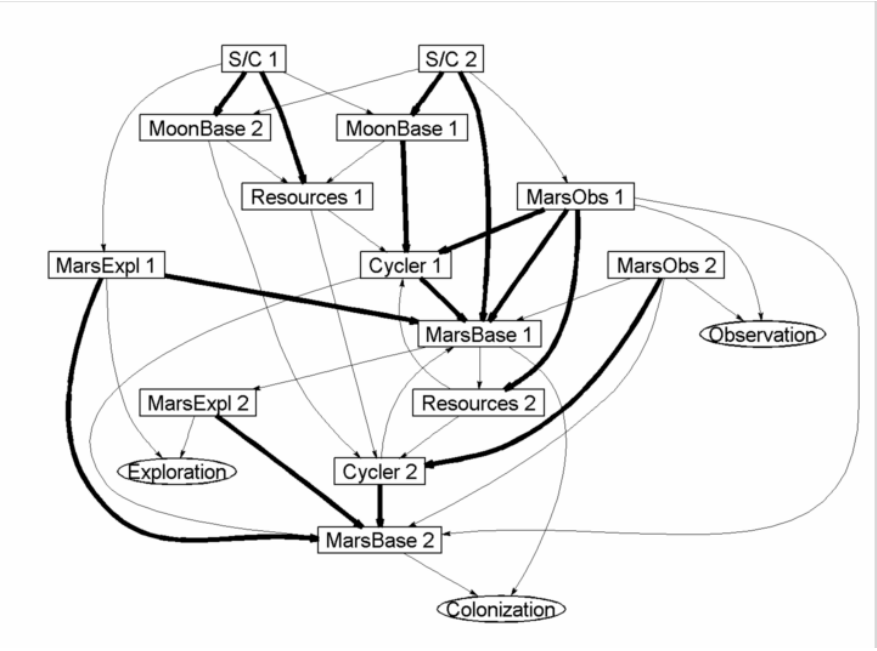


Figure 41: A space mission combined FDNA-DDNA network. Light edges represent functional dependencies, bold edges represent development dependencies.

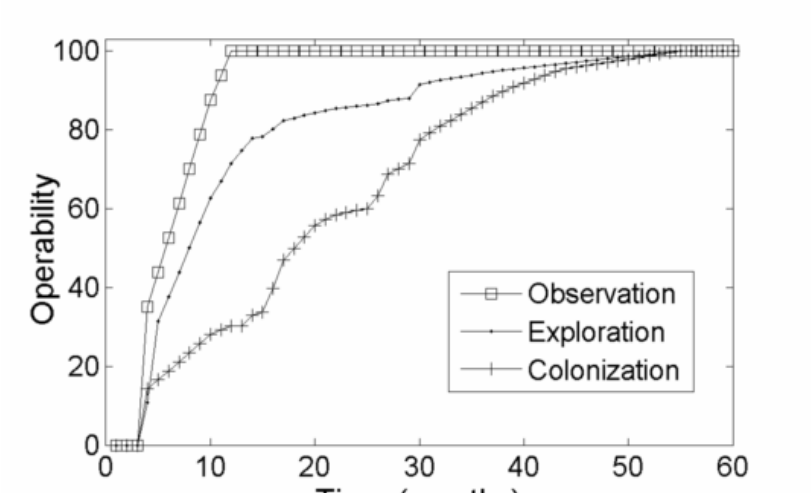


Figure 42: Time evolution of the operability of the desired capabilities for the space System-of-Systems in Fig. 41.

3.5.5 Conclusion and Future Work

This research work proposes the combined use of two dependency analysis tools to address the analysis of metrics that describe the overall features of a system-of-systems, throughout its development and lifetime.

In this study, we showed how FDNA and DDNA can be used to evaluate some of these properties in a complex system-of-systems, and to compare different developmental and functional architectures. Our research offers innovative perspective, and includes analysis of the impact of interdependencies on the features of a system-of-systems. The methods are domain independent, and applicable to various classes of problems. The use of the methods described in this paper is meant to serve as preliminary step towards a complete analysis and quantification of the metrics of interest for design, architecture, and development of systems-of-systems. These metrics will support decisions in system-of-systems engineering. Results from the method will also allow for trade-off between the metrics, according to the specific problem, to the available resources, and to the objectives.

Future improvement of this research will include cost analysis, a probabilistic model for the evolution of the system-of-systems, and formalization of the value of the metrics of interest. We will use an agent based model test bed to validate the inputs required by the methods to analyze specific problems.

3.6 Robust Portfolio Optimization

Our research in this section extends prior work in the *robust portfolio optimization* toolset and leverages a Conditional Value-at-Risk (CVaR) perspective to managing risks that can incorporate agent based simulation data in the decision-making process. We demonstrate the method using a LCS inspired Naval Warfare Scenario (NWS) as an illustrative case study. We also include an added framework for dealing with alternative measures of linear uncertainty.

3.6.1 INTRODUCTION

Modern engineering systems have evolved to now encompass large collections of interoperating systems, or a 'System-of-Systems' (SoS), that work cohesively to provide some overarching desired set of capabilities. The constituent systems in the SoS have a hierarchical structure, fall under independent operational and developmental jurisdictions, and have complex interactions due to the many interconnectivities that exist across technical and programmatic dimensions. Typical engineering efforts focus on locally incremental developments and do not explicitly consider their effects within the larger context of the original SoS architecture. Evolving these SoS constructs, by introducing new systems, retiring legacy systems, or implementing various acquisition policies, is fraught with cascading risks that manifest due to system interconnectivities. The result is often inflated costs, delayed schedules, and compromised performance, as evidenced by various program failures. The inherent difficulties in architecting SoS that typically span across multiple domains, especially when considering the ubiquity of uncertainties, presents the need for effective analytic tools in minimizing risks, mitigating unnecessary costs, and maximizing SoS level capabilities. These difficulties are further exacerbated by the large number of decision variables involved in developing an SoS architecture; this makes meaningful analysis of an SoS a task that goes well beyond the immediate mental faculties of the SoS practitioner.

This section focuses on one aspect of the workbench: a portfolio based approach that identifies optimal 'portfolios of interconnected systems based on practitioner's preferences of SoS level capability, cost and acceptable risks. The portfolio formulation extends prior research that uses robust portfolio optimization techniques to develop SoS architectures [4]. Our current work adopts additional innovations from financial engineering using a 'Conditional Value-at-Risk' perspective as a means of protecting the portfolio from simulated/observable worst case losses in performance or cost. Our framework is aimed at leveraging performance outputs of agent-based simulation of the operations of an SoS architecture as part of the portfolio formulation.

3.6.2 BACKGROUND

Research in financial engineering and operations research have yielded computational tools that assist portfolio managers in making better informed investment decisions. Central to portfolio allocation, is the idea of maximizing expected profit while mitigating the risks attributed to inherent volatility in the observed returns of the underlying financial asset. Seminal work by Markowitz in 1952 introduced a method of optimally allocating investments that maximizes expected profits given an investor's tolerance of risk. The resulting optimization problem is a quadratic programming (QP) problem which is amenable to highly efficient methods of solution. In a parallel vein, more recent advances in the field of optimization have recognized the impact of data uncertainty, where errors in estimated parameters of an

optimization problem can result in highly suboptimal solutions. The impact of uncertainty in data, as typically encountered in real world problems, has resulted in much theoretical work in developing robust counterparts of LP, QP and other types of convex optimization problems [Fabozzi, Kolm, Pachamanova & Focardi (2007), Tutuncu & Cornuejols (2007)] We have, in prior work, adapted the robust mean-variance approach of Tutuncu [9] to balancing the expected rewards of selecting 'portfolios' of interdependent systems against development time risk. However, the mean-variance approach accounts for both losses and gains through parameterization of risk as observed variance. Additionally, variances are assumed to typically follow a normal distribution. (or close to normal) – a notion that does not extend easily to operational contexts of risks where highly complex interdependencies can result in complex joint distribution behavior for interacting agents.

3.6.2.1 Conditional Value-at-Risk Approach to Risk Management

A more recent measure of risk, developed by financial engineers at J.P. Morgan, is the Value-at-Risk (VaR) measure that defines percentiles of loss and represents predicted maximum loss with a specified probability level over a defined time horizon [Tutuncu & Cornuejols 2004, Ursayev 2000]. A direct evolution of the VaR measure is the Conditional Value-at-Risk (CVaR) that represents a weighted average between the value at risk and the losses exceeding the value at risk measure; this is important as protections against VaR alone do not limit exposures to the maximum losses that can be incurred should worst case scenarios be realized. The CVaR formulation to managing portfolio risk is very attractive since it does not require explicit construction of complicated joint distributions in the formulation, results in a linear programming (LP) problem, and satisfies subadditivity of risks. (For a detailed derivation of the linear programming counterpart of CVaR, we invite the reader to reference [Uryasev 2000]). The formulation assumes a linear loss function associated with each asset, as is typically the case for holding financial assets. The resulting linear optimization problem can be written as the following:

$$\min_{x,z,\gamma}\left\{\gamma+\frac{1}{(1-\alpha)\mathrm{S}}\sum_{s=1}^{S}z_s\right\} \tag{6.1}$$

subject to:

$$z_s \geq \sum_i (b_i - y_{is})' x_i - \gamma \tag{6.2}$$

$$\sum_i \mu_i x_i \geq R \tag{6.3}$$

$$z_s, x_i \geq 0 \tag{6.4}$$

Eq. 6.1 is the objective function that seeks to minimize the CVaR and comprises of the value at risk term, ϒ, and weighted summations of the simulated loss scenarios (s=1….S), at the prescribed confidence level, α. Eq. (6.2) is the inequality constraint associated with the loss incurred for each simulated scenario where bi is the expected return and yis is the stochastically simulated return scenario (s) for asset (i); the number of scenarios (S) represents the total number of Monte Carlo simulations run. Eq. (6.3) enforces a minimum expected return requirement of total R from the chosen portfolio; a Pareto frontier is typically generated by solving the optimization problem of Eq. (6.1-6.4) using a range of values for R. The resulting frontier represents the optimal set of portfolios that best tradeoff expected return against CVaR. In the context of a SoS development framework, the frontier will illustrate the tradeoffs between performance and anticipated worst case scenario losses at the prescribed confidence level.

3.6.2.2 System-of-System Network Architecture and Optimization

In this research, we adopt a SoS hierarchical network description to guide the development of the portfolio optimization approach. While the interactions between constituent systems may exhibit complex dynamics due to various physical or operational effects, the archetypal system interactions are intuitively linear and combinatorial in nature. The interconnected nodes of the hierarchy are governed by connectivity rules of behavior, and, with each node having a discrete set of distinct *capabilities* and *requirements*. The nodes are subject to various behaviors of interaction that ultimately provide some SoS level capabilities. The individual nodes can be thought of as 'investment instruments' with associated costs (requirements) and potential payoffs (capabilities), having an overall SoS level performance (investment portfolio performance). This portfolio view allows tools from operations research and financial engineering to be used to tackle the combinatorial challenges of selecting an appropriate portfolio of systems, based on and SoS practitioner's preferences of tolerable operational risk, cost and desired SoS level performance.

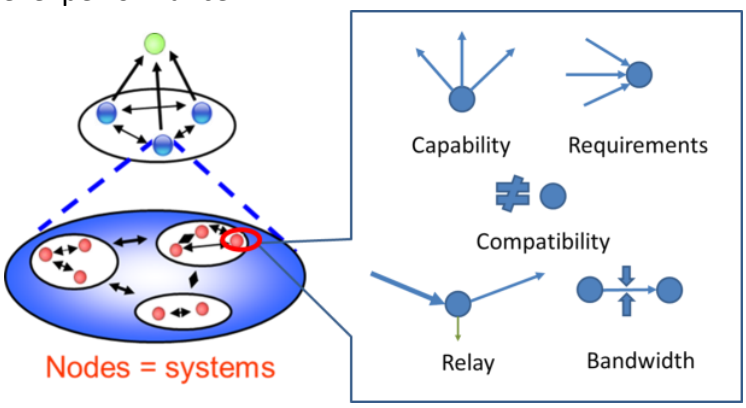


Figure 43: Generic SoS node behaviors

Fig. 43 shows the generic behaviors that individual nodes (systems) exhibit on the SoS network. The idea is to model basic, aggregate systems level interactions as simple, nodal behaviors that are applicable to a wide variety of inter-system connections. The motivation is to capture the salient features within a framework that can be translated to an effective mathematical model. While not exhaustive, the combinations of these nodal behaviors as modeling rules can cover a large set of real world inter-system interactions. Fig. 2 shows the five most intuitive nodal interactions where:

- Capability: Nodes have finite supply of capabilities that are limited by quantity and number of connections.
- Requirements: Nodes have requirements to enable inherent capabilities. Requirements are fulfilled by receiving connections from other nodes that possess a capability to fulfill said requirements.
- Relay: Nodes can have the ability to relay capabilities between adjacent nodes. This can include excess input of capabilities that are used to fulfill node requirements.
- Bandwidth: Total amount of capabilities or number of connections between nodes are bounded by connection bandwidth.
- Compatibility: Nodes can only connect to other compatible nodes.

The performance of a SoS is related to the ability of the connected network of individual systems to fulfill SoS level objectives. It is assumed that these core objectives can be at least, approximated quantitatively – a notion that leads us into the idea of agent based modeling. Agent based models have been extensively used as a flexible means of simulating collective behavior of interacting entities of 'agents'. In this case, the agents would correspond to the collections/clusters systems that are interconnected to give rise to some agent behaviors.

3.6.2 Conditional Value at Risk Optimization for SoS Architectures

The SoS portfolio optimization problem is posed as a mathematical programming problem that seeks to minimize the SoS performance index CVaR exposure as quantified from agent simulated operational losses in SoS level performance. The optimization problem also is subject to satisfying a range of physical and operational constraints. The resulting equations are given as the following:

$$\min_{x,z,\gamma}\left\{\gamma+\frac{1}{(1-\alpha)\mathrm{S}}\sum_{s=1}^{S}z_s\right\} \tag{6.5}$$

$$z_s \geq \sum_i (\mathrm{b}_i - y_{is})' x_i^B - \gamma \tag{6.6}$$

$$\sum_i b x_i^B \geq SoS_{CAP} \tag{6.7}$$

$$\sum_j x_{cij} \leq x_i^B S_{ci} \tag{6.8}$$

$$\sum_i x_{cij} \geq x_j^B S_{rj} \tag{6.9}$$

$$x_1 + \mathrm{L} + x_n = L \tag{6.10}$$

$$\sum_c x_{cij} - x_{ij} M \leq 0 \tag{6.11}$$

$$M \sum_c x_{cij} - x_{ij} \geq 0 \tag{6.12}$$

$$x_{ij} \leq \mathrm{Limit}_{ij} \tag{6.13}$$

$$\sum_i x_{cij} - \sum_j x_{cij} - x_j^B S_{rj} = 0 \tag{6.14}$$

$$x_{cij} \leq \mathrm{Limit}_{cij} \tag{6.15}$$

$$x_{cij} = 0 \quad c \in \mathrm{capability} \tag{6.16}$$

$$x_{cij} \in \mathrm{real, binary}\ , x_j^B \in \mathrm{binary} \tag{6.17}$$

Eq. (6.5) is the objective function that seeks to minimize the CVaR of selecting a collection of SoS level assets that directly contributed to the SoS performance index calculations. Eq. (6.6) incorporates agent based simulated outcomes of potential SoS level performance losses through the vector(y_{is}) which represent the simulated outcomes in performance for system (i) under scenario (s). Here, we define a scenario as being a situation where the system (i) is deployed alongside other systems in the agent model simulation and under different mission scenarios of operation. The number of scenarios is equal to the number of agent simulation runs required to reasonably approximate the outcomes of the SoS architectures. Eq. (6.7) ensures a minimum SoS level of performance as constrained by the constant (SoS_{cap}) – the optimization problem is solved using a range of values to generate the Pareto frontier that trades off SoS level performance for SoS value at risk. Eq. (6.8) ensures that supply of a capability (c) does not exceed the maximum limit of each node. Eq. (6.9) ensures that the requirements of each node are satisfied by incoming capabilities from connecting nodes. Eqs. (6.10-6.12) enforce combinatorial rules between systems and ensure compatibility. Eqs. (6.11-6.12) adopt a 'Big-M' approach to keeping track of the number of connections that each system

makes through the variable x_{ij}. Eq. (6.13) enforces limits on the number of connections that each node can make, as dictated by the specification of the node. Eq. (6.13) enforces that the total of some capability (q) that is supplied to a node, combined with its inherent capability (c) is not exceeded by demand for the capability from connected nodes.

3.6.2.1 Application to Naval Warfare Scenario (NWS)

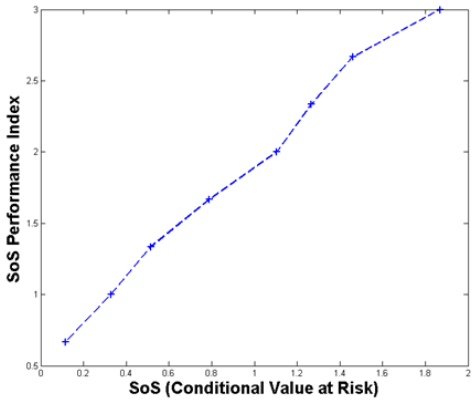
We demonstrate notional application of the presented method for the case of a Naval Warfare Scenario (NWS) that is based on the Littoral Combat Ship (LCS) concept of operations. The LCS is a naval platform that provides agile, cost effective solutions for naval operations in littoral waters. The platform serves to fulfill mission objectives through use of interchangeable ship packages that include: Mine Warfare (MIW), Anti-Submarine Warfare (ASW), Surface Warfare (SUW) and Irregular Warfare. From an acquisitions and systems engineering perspective, the LCS's modularity facilitates future development of the platform through a highly flexible *open architecture* policy and allows for competitive elements of contracting to be brought to bear in reducing acquisition costs.

Table 18: NWS candidate systems

		Weapon Range	Detection Range	Anti Mine	Comm. Capability	Power Capability	Power Req.	Comm. Req.	Max Connect.
ASW	Variable Depth	0	50	0	0	0	100	200	1
	Multi Fcn Tow	0	40	0	0	0	90	120	1
	Lightweight tow	0	30	0	0	0	75	100	1
MCM	RAMCS II	0	0	10	0	0	70	120	1
	ALMDS (MH-60)	0	0	20	0	0	90	150	1
SUW	N-LOS Missiles	25	0	0	0	0	0	250	1
	Griffin Missiles	3	0	0	0	0	0	100	5
Seaframe	Package 1	0	0	0	0	300	0	0	5
	Package 2	0	0	0	0	450	0	0	0
	Package 3	0	0	0	0	500	0	0	4
Comm.	Package 1	0	40	0	180	0	100	0	5
	Package 2	0	200	0	200	0	120	0	3
	Package 3	0	0	0	240	0	140	0	2
	Package 4	0	0	0	300	0	160	0	4
	Package 5	0	0	0	360	0	180	0	4
	Package 6	0	0	0	380	0	200	0	5

In our concept NWS scenario, Table 11 presents candidate system information for the Naval Warfare Scenario operational network. Each candidate system has a collection of capabilities and requirements as listed. The individual system capabilities, as listed in columns 1-5, can be used to either directly fulfill an overarching SoS requirement (listed in columns 1-3), or to fulfill individual support system requirements (columns 4-9). Columns 6-7 are systems requirement metrics across the candidate systems. Zero value entries in there columns indicate that the

respective listed system does not have that particular system requirement to be fulfilled. In this simplified scenario, it is assumed that a communications layer exists where all assets in Table 11 have an ability to 'communicate' with one another in the transfer of information, subject to a path-wise cost. The objective here is to select a collection of assets (system) from the available list in Table 11 to that minimizes the SoS level conditional value at risk; this is subject to input agent based model data on the interactive performance of candidate systems and the minimum expected performance of the SoS. Additional constraints include the fact that only one system can be selected for each package with the exception of the communications packages where a total of up to two may be deployed. We assume a simulated output of potential outcomes for interactions between feasible collections of candidate systems in the NWS; this collection of agent simulation outcomes represents the simulation vectors (y_{is}) of systems that interact within a missions (or multiple mission) scenario and under different feasible architectural considerations.



	Minimum SoS Performance Constr.	0.5	0.75	1.25	1.5	1.75	2.25	2.5	2.75
ASW	Variable Depth	X	-	-	-	X	-	-	X
	Multi Fcn Tow	-	-	-	X	-	-	X	-
	Lightweight tow	-	X	X	-	-	X	-	-
MCM	RAMCS II	X	-	X	X	X	-	-	-
	ALMDS (MH-60)	-	X	-	-	-	X	X	X
SUW	N-LOS Missiles	-	-	-	-	-	-	-	-
	Griffin Missiles	X	X	X	X	X	X	X	X
Seaframe	Package 1	-	-	-	-	-	-	-	-
	Package 2	-	-	-	-	-	-	-	-
	Package 3	X	X	X	X	X	X	X	X
Comm.	Package 1	-	-	X	X	X	X	X	X
	Package 2	-	-	-	-	-	-	-	-
	Package 3	X	X	-	-	X	X	X	-
	Package 4	X	X	-	-	-	-	-	-
	Package 5	-	-	X	-	-	-	-	X
	Package 6	-	-	-	X	-	-	-	-

Figure 44:. (a) CVaR efficiency frontier (b) Portfolio compositions for CVaR frontier

Fig. 44 (a) shows the efficiency frontier that results from solving the minimization problem of Eq. (6.5-6.17), using values ranging from 0.5 to 2.75 for minimum SoS performance required (SoS_{cap}). The problems were solved in the MATLAB [MATLAB 2011] environment using YALMIP [YALMIP 2004] interface with the Gurobi solver option. Each discrete point on the frontier of Fig 3(a) represents an optimal portfolio at a corresponding minimum SoS_{cap} value. As the required minimum SoS performance (SoS_{cap}) is increased, the value at risk increases as well; the frontier is similar in nature to the mean-variance frontier of Markowitz in that it trades of risk against performance. The difference here however is that the simulated risks (loss in SoS level performance) can account for highly complicated dependencies between assets and are not constrained to restrictive assumptions as in the case with the mean variance approach. The corresponding portfolio compositions (collections of systems selected) for each prescribed value of SoS_{cap} is given in the table of Fig. 44(b). As the minimum SoS level performance is increased, the composition of the portfolio changes, bearing the combinatorial rule in mind. The results shown in Fig 44(a) and (b) can provide the SoS practitioner with useful insights on the tradeoffs between SoS performance and potential loss due to complex cascading risks through comparison of the candidate optimal architectures. The optimal collections in Fig 44(b)

allow for the practitioner's additional insights to be brought to bear in deciding a final architecture that best fulfills strategic objectives.

3.6.3 Additional Portfolio Robust Measures

Our efforts in developing the robust portfolio optimization toolset have incorporated and demonstrated a range of algorithmic innovations from operations research and financial engineering, in dealing with decisions under data uncertainty. The robust linear approaches as introduced in earlier work for providing probabilistic guarantees on constraint violations for operational constraints, deals with uncertainties in the [A] matrix for a general set of inequality constraints [A]{x}≤{b}. Work in this research effort RT-44b have extended this to include risk management of portfolios using direct simulation based data – a very useful tool when joint distributions of risk measures becomes highly complex, and quantification of explicit losses becomes important.

We extend our consideration of uncertainty in linear operational constraints to account for uncertainties in both the entries of [A] and {b} simultaneously as well. This is accomplished through first parameterizing the uncertainties in [A] and {b} within the following *elliptical uncertainty* set:

$$\mathrm{U} = \left\{ \left[\mathbf{A^0; b^0} \right] + \sum_{j=1}^{k} u_j \left[\mathbf{A^j; b^j} \right], \|u\| \le \mathbf{1} \right\} \tag{6.18}$$

The conic optimization approach developed in literature (Nemirovski, 1998) assumes the above parameterization for uncertain linear coefficients and results in the following robust constraint:

$$(\mathrm{A}^0)^{\mathrm{T}}\mathrm{x}+\mathrm{b}^0 - \sqrt{\sum_{j=1}^{k}((\mathrm{A}^j)^{\mathrm{T}} x + \mathrm{b}^j)^2} \ge 0 \tag{6.19}$$

It must be noted that the uncertainties here are assumed to be *constraint-wise* uncertainties. The robust inequality in Eq. (6.19) can equivalently be rewritten into the following form:

$$\mathrm{z_j}=(\mathrm{A_j})^{\mathrm{T}}\mathrm{x}+\mathrm{b}^{\mathrm{j}},\mathrm{j}=0,\ldots,\mathrm{k} \tag{6.20}$$

$$(z_0, z_1 \ldots\ldots, z_k) \in C_q \tag{6.21}$$

where C_q is a second order cone. Although the resulting problem is nonlinear, it is nevertheless convex, amenable to highly efficient interior point algorithms and solvable in polynomial time for continuous variables. In the case of integer variables, the problem is NP-hard due to the integrality condition. The combination of state-of-the-art integer solvers with the underlying

interior-point algorithm to solve SOCPs makes the problem quite tractable for reasonably sized dimensions. The inclusion of the conic form of robust linear constraint, as shown in the equations above allows for strict feasibility constraints to be set, assuming uncertainty in both [A] and {b}; this can reflect, say, the uncertainties in power generated by systems [A] to meet an uncertain demand condition {b}.

3.6.5 Summary and Future Work

We have presented extensions to the robust portfolio optimization based approach in managing SoS architectural portfolio risks. The CVaR method describes the underlying SoS architecture as a generic network of interconnected nodes and leverages information from agent-based simulation data through a Conditional Value-at-Risk (CVaR) optimization framework. Use of explicit output information from a simulation environment provides valuable objective information on the performance and losses experienced by the SoS. The direct use of simulation performance data allows for explicit losses to be quantified and negates the need for complex joint probability distribution information prior to the portfolio optimization process.

A Naval Warfare Scenario (NWS) case study illustrates application of the method for operational risks; however, alternative measures of simulated risks can be used as well (e.g. financial, developmental schedule time). Additionally, the method is fully applicable to general SoS problems that can be described within the context of the generic network framework used in this research. Future work will explore alternative optimization frameworks to more generally address portfolio based performance and risk management in SoS development; this may include use of alternative methods of capturing measures of performance and outputs of key performance parameters from simulation data, and, optimization strategies that can best use such measures in building robustness and resilience for an SoS architecture.

4 Summary and Future Research

This report has detailed the results of development and application of a suite of MPTs that support decision-making in evolving SoS architectures. The efforts have extended prior work and refined relevant methods developed under RT-36 towards the goal of establishing an Analytic Workbench. These methods simultaneously enable the identification of positive impacts of interdependencies (SoS level capabilities and resilience) while mitigating negative one (e.g. developmental delay risks). The methods are demonstrated for extended test cases across multiple SoS scenarios and have shown promise in allowing SoS SE practitioners to perform measurable actions that translate to risk mitigation and SoS level capability development.

The research conducted in this RT-44b epoch has accomplished the following:

- Refined MPTs within the Analytic Workbench by extending theory for application on pertinent archetypal SoS practitioner questions that relate to cost, performance, schedule, and, on various 'illities' that drive SoS level performance.

- Implemented more realistic scenarios/assets for the concept NWS problem, to facilitate more instructive demonstration of the analytic workbench.

- Refined SoS level metrics and determine which metrics can be computed, and under which assumptions, for each candidate analysis method. (e.g. system importance measure in stand-in redundancy)

- Demonstrated, within the structure of an analytical workbench, the application of researched methods in evolving the NWS model, a FAA NextGen scenario and on orbit servicing. This involves application of each method to representative architectural challenges that SoS practitioners may face in developing the NWS problem.

- Established outreach efforts through journal and conference publications and INCOSE webinars. Sponsors have indicated the importance of these webinar sessions that are to showcase research in our RT- 44b efforts, in both facilitating feedback from industry and military experts in systems architectures.

4.1 Future Research Efforts

Research during our RT-44b phase has extended the theoretical and practical underpinnings of each method of the proposed Analytic Workbench. Additionally, the initial applications of our suite of methods on the demonstrative model of the NWS scenario, and, extended applications for on–orbit servicing and the FAA NextGen concepts have highlighted some promising analytical capabilities of the methods in dealing with a range of SoS architectural challenges. Our future work, under a SERC funded RT-108 effort will seek to specifically extend our workbench towards a more general implementation through pursuit of the following:

- Integrating standard input data specification for the workbench, to apply across methods; including relevant aspects of the DoDAF 2.0 definition framework as inputs to SoS analytic workbench methods

- Refining/extending individual analysis methods, including test and verification within the context of case studies and feedback from DoD SoSE collaborators

- Further design and definition of the workbench, including the software architecture

- Identifying other methods and tools which could be included in the workbench

- Develop a transition strategy for implementation of the work bench to both support user needs as well as a platform for developing and maturing added SoS analysis methods

Additionally, our work will seek to solicit potential SoS practitioner inputs from active collaborators towards refinement of the demonstrative version of our Analytic Workbench. Our goal would be to further enhance *value added* aspects of the proposed workbench by incorporating SoS practitioner feedback through example applications of the workbench in supporting SoS evolution of the LCS program.

5 BIBLIOGRAPHY

Ayyub, B., Systems Resilience for Multihazard Environments: Definition, Metrics, and Valuation for Decision-Making, Risk Analysis, doi: 10.1111/risa.12093, 2013.

Barker, K., Ramirez-Marquez, J. E., and Rocco, C. M., Resilience-Based Network Component Importance Measures, Reliability Engineering and System Safety, Vol. 117, pp. 89-97, 2013.

Beesemyer, J.C., Ross, A.M., and Rhodes, D.H., "An empirical investigation of system changes to frame links between design decisions and ilities," *Conference on Systems Engineering Research*, Saint Louis, Missouri, 19-22 March 2012.

Bertsekas, D., Tsitsikilis,J., Neuro-Dynamic Programming, Athena Scientific, Belmont, MA, 1996.

Bertsimas, D., & Sim, M. (2004). The price of robustness. Operations research, 52, 35–53. INFORMS

Burk, R., Parnell, G., Portfolio Decision Analysis: Lessons from Military Applications, In Portfolio Decision Analysis: Improved Methods for Resource Allocation , by Ahti Salo, Jeffrey Keisler and Alec Morton, pp.333-357, 2011.

Castet, J-F and Saleh, J.H., On the concept of survivability, with application to spacecraft and space-based networks, Reliability Engineering and System Safety, Vol. 99, pp. 123-138, 2012.

Chaize, M., Enhancing the Economics of Satellite Constellations via Staged Deployment and Orbital Reconfiguration, MIT, 2003.

Dahmann, J., Rebovich, G., Lowry, R., and etc, An Implementers' View of Systems Engineering for Systems of Systems, pp.1-6, 2010

Dahmann, J., and Smith, K., “Integrating systems engineering and test & evaluation in system of systems development,” IEEE International Systems Conference, Vancouver, Canada, pp.19-22 March 2012.

Davendralingam, N., Formulation and Demonstration of a Robust Mean Variance Optimization Approach for Concurrent Airline Network and Aircraft Design, Purdue University, 2011.

De Neufville, R., Real Options: Dealing with Uncertainty in Systems Planning and Design, Integrated Assessment Vol.4, No.1, pp.26-34, 2003.

De Weck, O.L., Ross, A.M., and Rhodes, D.H., “Investigating relationships and semantic sets amongst system lifecycle properties (ilities),” *Third International Engineering Systems Symposium*, Delft, The Netherlands, 18-20 June 2012.

De Weck, O., Neufville, R., Chaize, M., Staged Deployment of Communications Satellite Contellations in Low Earth Orbit, Journal of Aerospace Computing, Information, and Communication, Vol.1, pp.119-136, 2004.

Department of Defense, Defense Acquisition Guidebook, 2008.

Elsayed, E., Reliability Engineering, Addison Wesley Longman Inc., 1996.

Fabozzi, F. Kolm P.,Pachamanova D.,Focardi, S., Robust portfolio optimization and management (1st ed.). Hoboken, NJ: John Wiley & Sons, 2007.

Fitzgerald, M., Ross, A., Mitigating Contextual Uncertainties with Valuable Changeability Analysis in the Multi-Epoch Domain, Systems Conference (SysCon), 2012 IEEE International, pp.1-8, 2012.

Fitzgerald, M., Ross, A., Sustaining Lifecycle Value: Valuable Changeability Analysis with Era Simulation, Systems Conference (SymCon), 2012 IEEE International, pp1-7, 2012.

Garvey, P., and Pinto, A., “Introduction to Functional Dependency Network Analysis”, MIT, 2009.

Garvey, P., and Pinto, A., *Advanced Risk Analysis in Engineering Enterprise Systems*, CRC Press, 2012.

Guariniello, C., and DeLaurentis, D., “Dependency Analysis of System-of-Systems Operational and Development Networks”, *Conference on Systems Engineering Research, Procedia Computer Science,* Vol. 16, 2013, pp. 265-274.

Guariniello, C., and DeLaurentis, D., "Maintenance and recycling in space: functional dependency analysis of on-orbit servicing satellites team for modular spacecraft", *AIAA Space Conference*, San Diego, California, 10-12 September 2013.

Guariniello, C., and DeLaurentis, D., "Dependency Network Analysis: Fostering the Future of Space with New Tools and Techniques in Space Systems-of-Systems Design and Architecture", *IAF International Astronautical Congress*, Beijing, China, 23-27 September 2013.

Han, S.Y., Marais, K., and DeLaurentis, D., "Evaluating system of systems resilience using interdependency analysis," IEEE International Conference on Systems, Man, and Cybernetics, Seoul, South Korea, 14-17 October 2012.

Hannah, L., Powell, W., Stewart, J., One-Stage R&D Portfolio Optimization with an Application to Solid Oxide Fuel Cells, Energy System, pp.1-23, 2010.

Hsu, J., "Emergent Behavior of Systems-of-Systems", *INCOSE Mini-Conference*, 2009.

Jehiel, P., & Moldovanu, B. (2001). Efficient design with interdependent valuations. Econometrica, 69, 1237–1259. Wiley Online Library.

Keating, C., Rogers, R., Unal, R., Dryer, D., Sousa-Poza, A., Safford, R., Peterson, W., and Rabadi, G., "System of systems engineering," *Engineering Management Journal*, vol. 15, no. 3, 2003, pp. 36-45.

Keles, P., Hartman, J., Evaluating Portfolios of Multi-stage Investment Projects with Approximate Dynamic Programming, 2007.

Klein, M., Moreno, G. A., Parkes, D. C., Plakosh, D., Wallnau, K., & Seuken, S. (2008). Handling interdependent values in an auction mechanism for bandwidth allocation in tactical data networks. Proceedings of ACM SIGCOMM 2008 Workshop on Economics of Networked Systems (NetEcon 2008).

Maier, M., "Architecting Principles for SoS", *Systems Engineering*, Vol. 1, No. 4 1998, pp. 267-284.

Maier, M., Research Challenges for Systems-of-Systems. Reconnaissance Systems Division -- The Aerospace Corporation, 2005.

Mane, M., DeLaurentis, D.A., and Frazho, A., "A Markov perspective on development interdependencies in networks of systems," *Journal of Mechanical Design*, vol. 133, no. 10, October 2011.

MATLAB version 7.10.0. Natick, Massachusetts: The MathWorks Inc., 2010.

Mezzetti, C. (2004). Mechanism design with interdependent valuations: Efficiency. Econometrica, 72, 1617–1626. Wiley Online Library.

Mikaelian, T., An Integrated Real Options Framework for Model-based Identification and Valuation of Options under Uncertainty, MIT, 2009.

Miller, N. H., Pratt, J. W., Zeckhauser, R. J., & Johnson, S. (2007). Mechanism design with multidimensional, continuous types and interdependent valuations. Journal of Economic Theory, 136, 476–496. Elsevier.

Nai Fovino, I., and Masera, M., "Emergent Disservices in Interdependent Systems and System-of-Systems," *IEEE International Conference on Systems, Man, and Cybernetics,* Taipei, Taiwan, 8-11 October 2006.

OUSD(AT&L), Systems Engineering Guide for System of Systems, Washington, D.C.: Pentagon, August 2008.

Papakonstantinou, A., Rogers, A., Gerding, E. H., & Jennings, N. R. (2011). Mechanism design for the truthful elicitation of costly probabilistic estimates in distributed information systems. Artificial Intelligence, 175, 648–672. Elsevier.

Powell, W., Approximate Dynamic Programming: Solving the Curses of Dimensionality, Edition 2, 2010.

Powell, W., George, A., Simao, H., and etc, SMART: A Stochastic Multiscale Model for the Analysis of Energy Resources, Technology, and Policy, Informs Jounal on Computing, pp. 1 - 18, 2011.

Powell, W., Topaloglu, H., Fleet Management. Department of Operations Research and Financial Engineering, Princeton University, 2002.

Powell, W., Ayari, B., Berger, J., and etc, The Effect of Robust Decisions on the Cost of Uncertainty in Military Airlift Operations, ACM Transactions on Modeling and Computer Simulation Vol.9, no. No.4, pp.39-57, 2010.

Ramirez-Marquez, J. E. and Coit, D. W., Multi-state component criticality analysis for reliability improvement in multi-state systems, Reliability Engineering & System Safety, Vol. 92, No. 12, pp. 1608-1619, 2007.

Rausand, M. and Hoyland, A., System Reliability Theory: Models, Statistical Methods, and Applications, Second edition. New Jersey: Wiley – Interscience, 2004.

Rhodes, D.H., Ross, A.M., and Nightingale, D.J., "Architecting the system of systems enterprise: enabling constructs and methods from the field of engineering systems," *IEEE International Systems Conference*, Vancouver, Canada, 23-26 March 2009.

Rogers, A., Dash, R. K., Jennings, N., Reece, S., & Roberts, S. (2006). Computational mechanism design for information fusion within sensor networks. Information Fusion, 2006 9th International Conference on (pp. 1–7). IEEE.

Ronald, O., Navy Littoral Combat Ship (LCS) Program: Background and Issues for Congress, Congress Research Service, 2013.

Ross, A., Donna, R., Using Natural Value-Centric Time Scales for Conceptualizing System Timelines through Epoch-Era Analysis, Jun 2008.

Sage, A., and Cuppan, C., "On the Systems Engineering and Management of Systems of Systems and Federations of Systems". *Information, Knowledge, Systems Management,* Vol. 2, No. 4, 2001, pp. 325-345.

Simao, H., Day, J., and etc, An Approximate Dynamic Programming Algorithm for Large-Scale Fleet Management: A Case Application, Transportation Science, pp. 1-20, 2008.

Tierney, K. and Bruneau, M., Conceptualized and measuring resilience, TR News 250, pp. 14-17, 2007.

Tutuncu, R., Cornuejols,G., Optimization Methods in Finance (1st ed.), New York, Cambridge University Press, 2007.

Uryasev, S. "Conditional value-at-risk: Optimization algorithms and applications," Financial Engineering News, 2000, (14) 1-6.,

Van der Borst, M. and Schoonakker, H., An overview of PSA importance measures, Reliability Engineering and System Safety, Vol. 72, No. 3, pp. 241-245, 2001.

Webster, M., Santen, N., Parpas, P., An Approximate Dynamic Programming Framework for Modeling Global Climate Policy under Decision-Dependent Uncertainty, 2011.

YALMIP : A Toolbox for Modeling and Optimization in MATLAB. J. Löfberg. In Proceedings of the CACSD Conference, Taipei, Taiwan, 2004.