A Novel Approach to Mission-Level Engineering of Complex Systems of Systems; Addressing Integration and Interoperability Shortfalls by *Interrogating the Interstitials*

Ray Deiotte, ISSAC Corp; Robert K. Garrett, Jr, Missile Defense Agency

Abstract

This paper will establish a class of Systems of Systems (SoSs) defined as Mission-Level Systems of Systems and will explicitly explore and propose a solution to engineering the Integration and Interoperability (I&I) issues confronting Mission-Level SoSs. Mission-level engineering revolves around the concept of a mission context which manages uncertainties, dynamics and stochastic behaviors of SoS's. It has been posited that complex SoS's are driven not by the performance and behaviors of the constituent components, rather they are driven by the complex integration and interoperability, the interstitials, of the components to achieve Mission-level goals. Building upon this work, the authors, SoS practitioners, present a model-based SoS engineering approach that defines a SoS/mission architecture using both a physical space and multiple event space constructs explicitly accounting for both architectural and employment nuances of the SoS. A modeling approach, which explicitly defines the interstitials in physical space as operable nodes in event space, will be presented. An exploration of extensions to agent-based modeling is proposed herein along with the discussion of the Event Space modeling approach. This paper provides an approach to Mission-level engineering of Systems of Systems, addressing traditional integration and interoperability shortfalls by interrogating the interstitials.

1   Introduction

SoS architectures tend to focus exclusively on collections of physical entities (hardware, software, and networks) rather than the distinct differences in the ways the physical systems can be employed. (Weter, 2012) (Office of the ASN (RDA) Chief Systems Engineer, 2006) The ability to investigate physical SoS architectures and the impact they have on employment strategies (and vice versa) has lacked severely throughout commercial and governmental arenas. The development of SoS's has focused primarily on bottoms-up composition with the emphasis on the constituent components and how they may physically connect to one another. (Office of the ASN (RDA) Chief Systems Engineer, 2006) The GAO has studied the problems of complex SoS's, and the U.S. Congress has attempted to legislate focus, but no holistic approaches have been proposed that provide the capability to interrogate both the physical and employment architectures of the SoS. (GAO, 2013) (GAO, 2008) (GAO, 2013) (GAO, 2012) Additionally, emphasis has been put on integration to compose SoS's, little has been done to address interoperability problems. While confusing, integration and interoperability are not the same, and there is not yet consensus within the SE community as to a common definition of interoperability (Morris, Levine, Meyers, Place, & Plakosh, 2004). Thus, there have been no actionable means to develop a mission capable SoS which deals directly with interoperability (Jamshidi, 2009), (Weter, 2012) So too does M&S of SoS's struggle, M&S for SoS's are focused primarily on Live, Virtual and Constructive (LVC)

Report Documentation Page		*Form Approved* *OMB No. 0704-0188*

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE **17 DEC 2013**	2. REPORT TYPE **N/A**	3. DATES COVERED **17 dec 2013 - 17 dec 2013**

4. TITLE AND SUBTITLE **A Novel Approach to Mission-Level Engineering of Complex Systems of Systems: Addressing Integration and Interoperability Shortfalls by Interrogating the Interstitials**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) **Ray Deiotte, ISSAC Corp Robert K. Garrett, Jr, Missile Defense Agency**	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Missile Defense Agency 730 Irwin Ave, Schriever AFB Colorado Springs, CO**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release, distribution unlimited**

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT **SAR**	18. NUMBER OF PAGES **62**	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**			

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

compositions for the intended use of training. Other intended uses tend to drive toward adding resolution and complexity across distributed, discrete-event simulations or through agent-based techniques. While much money has been spent on these methods, few technical gains at the mission-level have been realized. (Weter, 2012), (Jamshidi, 2009)

Complex, dynamic, distributed Systems of Systems (SoS's) are being developed and employed, which are intended to satisfy the objectives of single and multiple simultaneous missions. The development of these complex SoS's has, to date, relied on compositions built up from existing component systems in order to accomplish both the component-level and SoS-level missions to avoid the investment in new/improved systems development. The integration and interoperability (I&I) of these systems into a complex SoS has been historically plagued with problems and continues to provide both technical and resource challenges. To address these challenges, efforts have been undertaken to improve the traditional systems engineering requirements process to better capture SoS needs in the component systems' requirements baseline. (Congress, 2009) (GAO, GAO-11-502 Missed Trade-off Opportunities, 2011) (GAO, GAO-13-103 Weapons Acquisition Reform, 2012)

However, progress to improve SoS performance by this route has been limited. Perhaps another approach should be considered. In their paper, Garrett, et al. (Garrett, Baron, Moreland, & Anderson, 2012) posit that SoS's are driven not by the performance and behaviors of the constituent components, but rather by their interstitials, the I&I of the components (Garrett, Baron, Moreland, & Anderson, 2012). In their paper, Garrett, et al. proposes a way to explore the interstitials. In this paper we further investigate the I&I of complex SoS's and propose a methodology to access and effectively interrogate the interstitials.

This paper, and the techniques discussed herein, exploits both United States Department of Defense (DoD) and industry engineering best practices. Thus, we believe these techniques are extensible to many complex SoS's across the entire domain of Systems Engineering. The key notion of the best practices that we exploit is that of a *Mission* which guides the SoS. This mission constraint then focuses the employment of the SoS and influences SoS architecture and design. A mission is defined as the set of goals and objectives that the SoS is expected to perform against. Subsequently, a mission environment is defined as all of the entities and interactions; conditions, circumstances, and influences involved in the prosecution of the SoS against the mission. Finally, the mission environment is populated with mission threads. These mission threads are the description of the end-to-end set of activities and component systems employed to accomplish specific subsets of the mission goals and objectives. These mission-based concepts will be used herein to guide the SoS engineering process and the entire construct will be termed Mission-Level System of Systems Engineering, MLSoSE.

We found a few candidates across multiple domains like economics, biology, DoD and process engineering and some techniques like agent-based modeling that had potential, but fell short in addressing the complexities of partially self-governed Systems of Systems. In this paper, we define a method, develop a technique and demonstrate the methodology that accomplishes this which can be extended to any complex SoS in any domain.

1.1 The Problem

1.1.1 SoS – Definition and complexity

The system engineering literature is still working toward a consistent definition for system of systems (Jamshidi, 2009). Variance in the SoS definitions appear to the authors to concern key notions of federation, autonomy, and abstraction. In this paper we will present a Mission-level SoS definition that approaches the SoS as a top-down composition, with a focus on achieving strategic mission goals. In addition the concepts of autonomy and abstraction will be discussed as they pertain to SoS definition and mission success. Mission-level engineering is the domain of I&I over a complex SoS. Here the SoS "should be distinguished from large but monolithic systems by the independence of their components, their evolutionary nature, emergent behaviors, and a geographic extent that limits the interaction of their components to information exchange," (Maier, 1998) so that "each constituent system keeps its own management, goals, and resources while coordinating within the SoS and adapting to meet [mission] goals." (Director of Systems Engineering, 2010). The SoS then achieves mission performance goals by "integrating [appropriate] independently useful systems into [the SoS to deliver these] unique capabilities." (Director of Systems and Software Engineering, 2008). In addition to integration to achieve syntactic composition of the SoS, mission-level engineering" (Director of Systems and Software Engineering, 2008). Mission level engineering requires the implementation of governance, with the establishment of mission context, via an architecture, to achieve a unique, semantically composed SoS (achieving the goals for interoperability). This mission context needs to manage uncertainty, dynamics, and the resultant stochastic behaviors. While much has been written and institutionalized in systems engineering (SE) best practice, and while the literature base on Systems Engineering and integration (SE&I) has recently grown, the domain of mission-level engineering across complex SoS's has few established tools, techniques and processes.

The authors propose that the domain of mission engineering of complex SoS's is comprised of (Jamshidi, 2009):

1. Being a composition of independent constituent systems that maintain their own internal management. The component systems bring forth their engineering artifacts; models and simulations; verification, validation and accreditation for their intended use; and their operational governance.
2. Being bound together via a Command, Control, Communications and Computers (C4) based architecture established to meet mission-level goals,
3. Achieving its mission goals through the application of top-down governance defined by the Mission Environment,
4. Establishing Mission Threads to facilitate the development of the Mission-level SoS architecture and establishing the subsequent domain over which the SoS will be tested,
5. Providing a quantitative description of the mission-level SoS, proposing that the SoS behavior is dominated by the interstitials (not the detailed behavior of the individual constituent systems) and that the interstitials have inherent uncertainty due to the nature of the mission environment and C4 based architecture, and are thus stochastic in nature

According to this definition, a federation or confederation of autonomous systems is not necessarily a SoS, but it can portray many SoS characteristics and behaviors. In fact, there are many times where a

3

federation of heterogeneous entities do comprise a SoS, but the construction is deliberate and organized from the Top-Down rather than ad hoc, from the bottom up. With the domain of complex SoS engineering established, a given SoS can then be applied within several Mission Environments (MEs) and within any ME tested against many Mission Threads (MTs).

We have reviewed much of the literature and have found that there is a shortcoming in the lexicon with regard to the notions of abstraction, fidelity, precision, accuracy and resolution. The concept of accuracy refers to how close a representation approximates the real system. The mean of accurate measurements typically have a low percent difference while the standard deviation (the distance from each other) has a high percentage difference. Precision is the other side of the coin from accuracy: all of the measurements may not be close to the real system, but they are close together. In other words, the percentage difference of each measurement from the referent is high, but the standard deviation of all of the measurements is low. When we refer to fidelity we attempt to indicate that the level of accuracy *AND* the level of precision is high, meaning that all of our measurements are close to the referent, *AND* they are close to each other. The concept of resolution is best equated to the level of detail that can be observed in the representation similar to a multi-power microscope. In this manner, abstraction and resolution can be used interchangeably. Abstraction, too, carries a connotation of fidelity. However, abstraction focuses on ensuring that the precision of the model is similar to the precision of the referent and intends to keep the accuracy within an order of magnitude. Obviously as we increase the resolution or decrease the level of abstraction (as the two terms are inversely related) the necessary accuracy should increase while the level of precision stays in agreement with the referent.

Similarly, complexity can be difficult to quantify or even identify, but there is an inherent understanding of things that are complex. We know that automobiles may be complex, manufacturing processes may be complex, and even that large groups of people or animals may be complex, but they are all complex for very different reasons. For the purposes of MLSoSE, we define complexity according to the number of entities in the SoS and the amount of possible interactions those entities have with one another within the ME. Figure 1 illustrates the construct that we will use throughout the paper to represent the complexity of the ME. The lower left portion of Figure 1 illustrates the notion of few entities which interact very little with one another. In this domain we have M&S (and real systems) that do an amazing job at capturing the low-level physical dynamics of the systems. However, at the most extreme points (high number of entities, large amounts of interaction) the current set of solutions struggle to approximate those complex points. In the same fashion, the uncertainty that we know and can quantify sufficiently is large. Each shaded circle in Figure 1 represents a Mission Thread and its domain; the dotted circles about the Mission Threads represent the inherent uncertainty about the mission thread. The distance between the mission thread and the MT uncertainty circle represents a relative amount of uncertainty: the greater the distance between circles the greater the uncertainty. In other words, at the lower left corner of Figure 1, we typically have a small uncertainty space while at the upper right corner the uncertainty may be large and difficult to quantify. These are necessary concepts that will permeate this paper. Critical to the definition of the ME is the notion of uncertainty, which includes both the aleatoric uncertainties and the epistemic uncertainties that may lead to emergent behaviors. The

4

uncertainty is manifested within the unique character of each MT as a result of the impact of the mission environment on the employed MT.

1.1.2 Interstitials

The authors contend that it is possible that our rich heritage on successful SE and SE&I tools, techniques and process or procedures may not be extensible to the mission-level engineering of complex SoS.   In the component system SE&I, the approach is one about a classic systems engineering "Vee" with an emphasis on the establishment of requirements, determinism, test and Verification, Validation, and Accreditation (VV&A).   This approach may be too rigid and overly simplistic given the SoS propensity towards stochastic behaviors, un-testable regimes, and profound challenges for VV&A. The focus on the complex SoS should be on managing uncertainty, and achieving interoperability/semantic composition through the engineering of flexible and evolving architectures (Selberg & Austin, 2008).   These SoS architectures predominately consist of C4 that connect the constituent systems. In addition, a premise was presented by Garret, et al. (Garrett, Baron, Moreland, & Anderson, 2012), that "SoS behavior is driven by the Interstitials where the term "interstitials" is used to define the domain of interfaces, interoperability, and integration between constituent systems in a SoS."   To that end the authors propose that a mission-level SoS should be architected to accommodate the self-governing constituent systems and not necessarily the other way around. Their paper presents a set and graph-based approach to represent the 'physical space' architecture. In the graph of physical space, component systems are represented by nodes and the interactions are represented by edges. The interstitials are then a path of edges and defined by the cross terms of the physical space adjacency matrix.   An additional construct is being proposed in this paper where the physical space is transformed to an "Event Space" to facilitate modeling and subsequent simulation.   In this event space, the edges of physical space become nodes in a Bayesian Network represented by a probability distribution function (PDF). The key tenet of Event Space is that each event space node is explicitly an interstitial in physical space.

1.1.3 Top-Down Constraints – Mission Environment/Mission Thread

The DoD has invested significant effort in order to define an approach for testing component systems within a complex SoS under clearly defined mission context. (JTEM, 2009) This DoD mission context appears extensible to any mission domain, defense or civilian, public or private, and will be used as a basis throughout this paper. In addition, the DoD approach appears robust and directly applicable to the establishment of the mission/SoS architecture, requirements, and the subsequent engineering of the SoS I&I. Thus, Mission engineering needs to focus not only on those materiel, physical entities of the SoS, but also on the non-materiel attributes of doctrine, organization, training, materiel, leadership and education, personnel, and facilities (DOTMLPF) (DoD, 2012).   The notion of Joint refers to a multi-organizational, multi-cultural, enterprise construct brought together to "jointly" achieve the mission objectives.

There are two key concepts in the mission context: the Joint Mission Environment (JME) and the Joint Mission Thread (JMT). The JME is the set of physical entities – conditions, circumstances and influences to meet a specific mission objective.   The JMT is the path within the JME set that provides the operational and technical description of the end-to-end set of activities and systems (CJCS, 2010). For

5

the remainder of this document, we will use the terms Mission Thread (MT) and Mission Environment (ME) as constructs that are derived from the JME and JMT. The MT can be mathematically defined as an SoS operating within the ME, "A, and a subgraph, M, where M $\subseteq$A, a mission thread for a specific scenario, Ti can be represented as the composite vector, or tensor Ti, where Ti = {M1, M2, M_i}, and M_i is the active path in A at time, *i*." (Garrett, Baron, Moreland, & Anderson, 2012). This definition can be explicitly aligned with the construction of an event space mentioned previously. The authors believe that these constructs of mission, ME and MTs represent a readily extensible engineering best practice for mission-level engineering over a complex SoS, especially with respect to the domain of SoS I&I.

1.1.4   Uncertainty

The compositions with any given ME and/or MTs (the notion of joint is viewed as implicit throughout the balance of paper) introduce some amount of uncertainty. Examples include such things as initial conditions, natural environment, availability of resources, stimuli characteristics and behaviors, etc. This uncertainty is either in the form of things which we know about but change every time we employee the SoS (statistical uncertainty), or in the form of things we could know in principle, but don't know in practice (emergent behaviors, systematic uncertainty). The aleatoric and epistemic uncertainty (respectively) are necessary evils, and the best we can do is to quantify and manage aleatoric uncertainty, and drive out epistemic uncertainty so that we may, overall, reduce risk in the design, development, deployment, employment and retirement of SoS's and their constituent components. Ultimately, uncertainty should drive the level of abstraction of the representation (real or otherwise) of the SoS's detail otherwise unwarranted focus can lead to faulty conclusions or significant expense of both time and engineering resources when uncertainty is great.

Because of the economic, geographic and socio-political constraints which impact any complex SoS, many – if not all – of the aspects of the integrated SoS must be explored and subsequently tested within the realm of models and simulations (M&S). However, as a model is only a representation (i.e., an approximation) of the real world, thus there is inherent uncertainty involved in any M&S solution. Examples of conditions that introduce uncertainty in the M&S domain are the range those conditions can take, multiple causal chains, timing, and even the inherent uncertainty in component-level computations (accuracy, precision, units, etc.) and those in distributed computations (Pilch, Trucano, & Helton, 2006).

1.2   Common Definition of Complex Systems of Systems

Given the fluid definition for SoSs over their diverse application domains, we propose that the Mission-Level SoS is: (Jamshidi, 2009):

- Composed of independent constituent systems that maintain their own internal management
    - The component systems bring forth their engineering artifacts, models and simulations
    - Verification, validation, accreditation and certification for their intended use
    - Independent component operational governance
- Composed into a top-down C4 architecture
- Able to achieve its mission goals through the application of top-down governance defined by the Mission Environment

- Able to employ Mission Threads to facilitate the development of the Mission-level SoS architecture and establish the subsequent testing domain
- Dominated by the I&I (interstitials) and *not* the detailed behavior of the individual constituent systems – these interstitials have inherent uncertainty due to the nature of the ME and C4 based architecture, thus are stochastic in nature

From here forward the authors imply a Mission-level SoS whenever the term SoS is used. Inherently, a SoS intends to be greater than the simple sum of its constituent components, but to what end? Is the SoS intended to be slightly better, tremendously better, is it decomposable into single or pair-wise component functions; can the SoS be further abstracted up into another SoS? In short, to be considered a SoS, it must be composed of independent components that can work within or without the SoS context and when they are composed together, with the goal that the sum of the constituent components' performance is more than the individual components could perform, if employed, additively. The key to the SoS definition is that the components may not have been designed with a prior knowledge of the intended use of the SoS as a consideration manifested in their requirements basis.

1.3   Allusion to a Solution

Because of the natural and man-made limitations, and the realities of budgetary constraints to implementing and testing complex SoS's, much, if not all, integration and employment testing must be done through M&S. Traditional M&S approaches leverage a physical perspective of the referent system, preferring to recreate (through approximation or even the use of tactical hardware and software) the referent in the model. For traditional systems architected from the onset from the components up as a single system (e.g., airplane, car, etc.), this view is sufficient to accurately represent the system. However, when the SoS is comprised of independent, autonomous systems that were created for their own intended use, traditional modeling of every physical component becomes rapidly untenable.

We have deemed the modeling technique of representing each physical entity as defining a Physical Space. The Physical Space representation takes the form of a DoDAF OV-1 (DoDAF, 2012)or simple graph (components on nodes, interconnections on edges) and emphasizes the components on the nodes. Figure 2 illustrates a simple system of two sensors (node *A* and node *D*), a controller (node *C*), and an actor/sensor (node *B*). This representation and example will be used for the many of the examples in this paper. In a normal physical space representation, each of these elements would be explicitly modeled to a certain level of abstraction – discrete event models, continuum models, etc. Each of these models is typically rooted in the physics that defines the primary functions of each node: for a sensor the model may represent wave forms, interaction with external stimuli, weather, etc. However, this deterministic approach (deterministic in that we can understand every possible cause and effect in the system) is difficult to explore the untestable regimes of complex SoS's. As the complexity of the environment increases, the amount of the domain in which traditional deterministic representations can interrogate decreases. The number of factors that must be interrogated increases with each additional entity or interaction and to test each combination of these factors becomes untenable. The sheer volume of the factor space that must be explored drives the cost of what interrogation that can be

7

done exponentially upward. Thus, the use of physical space representations to investigate SoS's is difficult, if not impossible.

Additionally, the edges or interconnections between nodes are at best modeled at the information or message level and at worst implicitly and instantaneously. Many of the problems with using a physical space representation to address complex SoS's stem from the inability (or unwillingness) to quantify uncertainty in the models and ME and to allow an environment in which emergent behavior (caused in part by the interstitials) can arise and be explored. However, legacy M&S tools and other physical space representations are necessary for exploring detailed aspects of the SoS – they are simply unable to explore the entire domain of the ME in a timely, cost effective manner.

In order to accurately probe the behaviors and sensitivities of the ME, we searched for a methodology and toolset which would provide the capabilities necessary to explore the vast domain encompassed by a complex SoS. This led to the establishment of a set of goals for the methodology and toolset:

- Rapid Execution (tens of thousands of iterations over the SoS in minutes),
- Stochastic in nature (able to explore the domain and range of SoS variables), and
- Focused on the drivers of SoS behavior, i.e., the interstitials

We found a few candidates across multiple domains like economics, biology, DoD and process engineering and some techniques like Agent Based Modeling that had potential, but fell short in one or more of the areas listed above. Nowhere could we find a method, technique or tool that provided all three capabilities. In this paper, we define a method, develop a technique and demonstrate a tool that accomplishes all three capabilities above which can be extended to any complex SoS in any domain.

The authors intend to present a model-based Mission-Level SoS Engineering approach that defines a SoS/mission architecture using both a physical space and multiple event space constructs. The basis for the modeling approach is to define each event as an interstitial in physical space and then explore the relationships between events using causal inference and probabilities of occurrence. Exploration of the event spaces will be done using a hybrid-agent approach and a novel method to combine event spaces into more complex spaces due to multiple stimuli. This methodology provides a novel approach to mission-level engineering of complex SoS's addressing traditional I&I shortfalls by interrogating the interstitials.

2   Background

2.1   SoS

Complex SoS's have been realized as a way to address mission-level and multi-mission problems in many domains. However, defining, classifying, designing, integrating, employing, certifying and accrediting complex SoS's is a non-trivial problem. Much of this is due to a lack of uniqueness at the mission level and the seemingly incessant desire to build SoS's from the bottom up. The notion of architecting a complex SoS's with the mission-level objectives in mind is crucial to ensure that the SoS is capable of

meeting the governance restrictions of the mission environment and ultimately addressing Mission-Level goals and objectives.

2.1.1 What is the SoS

Here we define a hierarchy pyramid as a basis to build a multiple-mission employment of complex SoS architectures from component-level entities. This pyramid construct builds on that of Morris, et. al. and extends it to the realm of the Mission space. Extending the notion of disparate personal and technical pyramids, we combine them into the mission construct as they are not distinct nor mutually exclusive. (Morris, Levine, Meyers, Place, & Plakosh, 2004) At the lowest level we define the system, component and sub-component.   These are the entities that are typically the focus of a component system acquisition plan or strategy. The next level is that of Trees, which introduces the idea of stringing a few component systems together to achieve some larger goal. Above Trees is a Forest (to which the Trees are a subset) which contains all possible Trees in a Mission Environment (ME). The superset of the Forest is the Mission Thread (MT). The Mission level contains multiple MTs and all of the information necessary to describe a Mission: its architecture, employment, stimuli and operating conditions. Finally, the larger set of Missions, or multi-missions, is the Campaign level which encompasses all possible Missions for a given domain. This hierarchy can be seen in Figure 3. Notably, the goals at each level in the pyramid are different and require different levels of information and knowledge to satisfy the goals. To that end, a SoS can be composed at any level of the pyramid. It should also be stated that the different layers of the pyramid do not indicate different abstraction levels; rather they are larger, more complex representations of the SoS.   Unlike traditional hierarchy pyramids which focus on greater resolution or lower abstraction as one moves from the top to the bottom, we define the pyramid to be held at a constant level of abstraction from top to bottom. This allows us to examine complexity within the pyramid with the most complex concepts at the top of the pyramid and least complex at the bottom.   By taking this perspective, we can better understand the composition of the SoS and the proper level of abstraction for the intended use.

Starting with this nominal hierarchy pyramid, we claim that the systems domain, i.e., the traditional single system, system engineering domain, extends from the System/Component/Sub-Component level to partially inside the Forest(s) level. This implies system intended use ranging from component testing and integration to partial integration for achievement of a common goal with another system. Thus, the Mission-level SoS domain extends from the Trees to the Campaign level. Effective SoS's should be able to provide information between levels of the Mission Hierarchy if not span multiple levels themselves in order to adequately investigate SoS behavior and performance. That being said, one may craft a SoS at any level; however, if the expectation to develop a Mission-Level SoS from the bottom up exists, the foresight and Mission constraints must be in place to bound and guide the SoS development. Yet, experience has proven that crafting a SoS from the bottom-up (i.e., composing disparate components into a federation) is fraught with extensibility pitfalls and rarely results in meeting the breadth of Mission-level expectations/requirements.

2.1.2 Mission Construct Definition

In order to move away from the construction of SoS's from the bottom-up, the definition of the constraints and methodologies to develop the SoS must change. Accurately articulating the Mission-

9

Level domain in which the SoS should operate is necessary to properly bound the SoS functionality both in its architecture and employment. Additionally, the goals and objectives of the Mission-Level are very different from those of the other levels. A focus on integration, interoperability and the prosecution of many simultaneous missions lies at the heart of the Mission-Level Construct. The Mission-Level definition should guide the development of SoS architectures by focusing on how the users will employ the SoS that are not necessarily consistent with the original intended uses of the constituent components.

2.1.3   Architecture and Employment – Complementing Engineering Processes

At the Mission level, the architecture of the SoS is a significant component of the development of the SoS. However, the employment of that architecture may be equally or even more significant than the architecture itself. With each mission and each mixture of missions, the mission governance, e.g., the tactics, techniques and procedures (TTPs) and the concepts of operations (CONOPs) employed change as does the capability of the SoS. Thus, being able to involve Mission-level governance in the development of the SoS is critical to the success of Mission-Level SoS's. In addition, the ability to interrogate the governance themselves with regard to Mission-Level architectures and stimuli could prove very useful for the promulgation of the SoS.

2.1.4   Mission-Level SoS Construction Must be Top-Down

The traditional approach of constructing a SoS tends to aggregate components into a loose federation (at best). This approach typically ignores the complexities of I&I as well as the Mission-Level goals and objectives. While this is not true in all cases, most bottom-up approaches only address a subset of the Mission-Level goals. We propose that in order to meet the entirety of the Mission-Level goals and objectives the SoS must be crafted from the top-down (Jamshidi, 2009) (GAO, 2008) (GAO, 2012) (GAO, GAO-13-103 Weapons Acquisition Reform, 2012). A top-down approach explicitly accounts for these Mission-Level governance as well as the I&I issues and employment of the SoS.

2.1.5   Syntactic and Semantic Composability

 In order to define and address the I&I issues, even from the top down, we must address the notions of composition of the SoS. Extending Surowiecki, we begin by defining three levels of entity (component system) interactions: autonomy, collaboration and cooperation (ACC) (Surowiecki, 2005).The concept of autonomy can be defined as independent entities that can perform their mission(s) with little to no additional information or interaction with external systems (barring, of course, object of interest stimuli). Collaboration, on the other hand, may be defined as one or more autonomous entities passing and using information with other entities in order to accomplish the autonomous entity's mission(s). Cooperation; however, focuses on the sharing of information and eliciting behaviors in order to accomplish SoS or Mission-Level goals and objectives. An explicit decomposition of the ACC paradigm can be seen in Figure 16, Figure 17, and Figure 18. A key to the ACC construct is that it could/should be dynamic. That is the SoS in concert with the entity systems react to the mission environment using local and enterprise performance information to dynamically move between the levels of ACC to meet the mission goals. If we are now to look at the domains of mission hierarchy presented in Figure 3, integration and interoperability, and the concepts of autonomy, collaboration and cooperation, we posit that integration lies in the purview of autonomy and into collaboration. Integration then tends to be

10

focused on connections between elements, pair-wise interaction and single mission threads at the component level. Interoperability, then is ranges from the collaboration level through cooperation and focuses on many-to-many, meaningful interactions, multiple mission threads at the SoS level, Mission-Level goals and objectives and the Mission Environment as a whole. This construct is depicted in Figure 4. The authors recognize that this may be viewed as a departure from the literature that has tended to focus on network communications, but we feel it is a critical component to the MLSoSE process. (Jamshidi, 2009)

If we are to assume that each component of the SoS comes to the composition as a complete autonomous entity, then we say that bringing two of these components together is integration or syntactic composition. If the autonomous components are able to connect, pass messages (correct protocol, format, etc.) and some (perhaps all) of the data is populated, we typically claim successful integration. Interoperation on the other hand can only be achieved through semantic composition or meaningful interactions and behaviors across mission-level stimuli (including uncertainty) to affect Mission-Level goals. This means determining the quality of the interactions of multiple systems simultaneously within the constraints of individual performance and top-down governance. Obviously, achieving cooperation through interoperability and semantic composition can be quite a daunting task.

2.1.5.1 *Integration and Interoperability – The Interstitials*
All of the connections between autonomous elements, both physical and behavioral (syntactic and semantic) can be thought of as the interstitials – the space between objects or entities. These interstitials carry a vast amount of information ranging from strictly measurement information to decisions and other behavioral context modifiers. In a classic system, the interstitials are typically well defined and can be managed easily. But in the domain of complex SoS's there are many interstitials between elements – most of which are not well documented or known at all. Because of the complex nature of many SoS's and the need to provide a cooperative, interoperable composition, Garrett, et al. posited that it is the interstitials that drive SoS performance and behaviors, not the individual components. Following this notion, the bulk of the problems with SoS architecture and employment can be derived directly from the interstitials.

2.1.6 Metrics
Questions of how to measure a system are often (and should be) at the forefront of traditional systems engineering. The traditional systems engineering approaches dictate that the measure of the system's performance or behavior is dictated by key performance parameters (KPPs) or key performance measures (KPMs), which are identified early in the SE lifecycle – traditionally in the requirements elicitation and analysis phases.

However, this same approach has proven to be at best minimally extensible to the SoS and Mission-Level domains. (GAO, 2008) (GAO, 2013) Factors such as complexity, emergent behaviors, uncertainty and the interstitials make the definition of quantitative metrics to measure the SoS above the component level difficult. Upon review of the SoS body of knowledge, there appear to be few well-established, reliable metrics and measures for gauging the behavior and performance of Systems of Systems.

11

2.2   Optimization and Sensitivity Analysis

The concept of optimization is critical to the development and employment of a SoS. At its essence optimization focuses on a set of parameters or metrics and simultaneously attempts to maximize (or minimize depending on cost functions and error budgets) them to address a set of goals or objectives. The ultimate goal of optimization is to determine the conditions and construction of the SoS that maximizes behavior and response to stimuli while reducing uncertainty and identifying potentially catastrophic conditions (Wolkowicz, 2006).   Measuring where the SoS is sensitive (positively or negatively) provides the opportunity for optimization around those areas as well as to find a global optimum state for the system. SoS optimization can be tricky; however, there are both quantitative and qualitative modifications (architecture vs. employment) that can be made to optimize the SoS. Overcoming this significant hurdle is necessary to effectively deploy a SoS.

2.3   Uncertainty

Beyond strictly the architecture and employment difficulties of complex systems of systems, uncertainty in the Mission Environment causes problems with SoS's ranging from operations to fielding to risk management. The types, sources and management of uncertainty is crucial to the composition and ultimately achieving some level of Verification, Validation, Accreditation, and Certification, VVA&C, of the SoS.

2.3.1   Types of Uncertainty

There are two primary types of uncertainty: things we know about but can't reliably predict and things we don't know at all.   These are the known and unknown unknowns, or aleatoric and epistemic uncertainty. Things we know about but change every time we run an experiment (statistical uncertainty) is defined as aleatoric uncertainty.   Epistemic uncertainty is related to the things we could know in principle, but don't know in practice (things we haven't discovered yet, systematic uncertainty). (Pilch, Trucano, & Helton, 2006) (National Academy of Sciences, 2012)

2.3.2   Sources of Uncertainty

Within the SoS there are many sources of uncertainty stemming from the Mission Environment. Environmental constraints, physical properties, system properties, architecture, initial conditions, and employment are all sources of uncertainty in the SoS.   Environmental constraints can introduce uncertainty that is both aleatoric and epistemic through weather, time of year, time of day, humidity, etc. Physical properties of systems and the Mission Environment like copper and fiber transmission speeds, radar wave-form generation, specular reflectance and object characteristics can also cause both aleatoric and epistemic uncertainty. Both types of uncertainty can stem from system properties that diverge from the intended use of the system under the employ of the SoS. The uncertainty introduced by the architecture and employment of the SoS ranges from timing of messages to how sensor search plans are conducted to the types and combinations of TTPs employed. In order to better quantify risk, these uncertainties must be at least accounted for and the epistemic unknowns should be driven out. (Pilch, Trucano, & Helton, 2006)

Specifically related to M&S SoS's, uncertainty can arise from inconsistencies in the level of abstraction used for the models of the referent SoS. If a similar level of abstraction for each of the models and for

12

the architecture as a whole is not implemented, then the representation of the SoS can be skewed because of the informational and representational differences in the models. Consistent levels of abstraction for an intended use is critical for proper M&S composition and SoS representation. Because M&S must represent entities and interactions that are given to us for free in the real world, uncertainty can be introduced through the environment, stimuli, and the level of abstraction to which the real entities are modeled. Additionally, as we extend M&S to a distributed, global context where M&S components are expected to interoperate over a wide geographic expanse, our ability to address uncertainty introduced by the simulations ("sim-isms") decreases dramatically. (Chibo, Hua, Ruili, & Shudao, 2012) (Roy & Balch, 2011)

There has been a motivation to model complex systems and SoS's by using the most detailed representations of entities as possible, sometimes going as far as to use the real-world systems in an M&S composition. This is done in an attempt to manage uncertainty, however naively. The intent is correct: manage uncertainty in the representations so that we can understand capabilities and limitations. However, the introduction of live systems into an M&S construct can be fraught with problems. The farther we move away from the real-world entities, the more uncertainty we introduce. This correlation between abstraction and uncertainty is one that is typically handled implicitly, but to better handle uncertainty as a whole, we must begin to quantify the relationship between abstraction and uncertainty. The authors advocate using an iterative approach to employing multiple levels of M&S. In that the top-level M&S guides the lower-level deterministic tools which in turn inform the top-level, stochastic M&S.

2.3.3   Uncertainty Management

The process of uncertainty management is difficult to say the least. In order to manage the uncertainty in a complex SoS one must identify, capture, quantify and determine an approach to deal with the uncertainties identified. At best, one may manage some of the aleatoric uncertainty in a SoS, but the bulk of the aleatoric uncertainty will be merely identified. Over time, some of the epistemic uncertainty may be transitioned to the realm of aleatoric, but that assumes adequate investigation of complex problem spaces with numerous factors defining them. According to Jiminez, et al. as the number of factors increases above five, the amount of the space traditional techniques can interrogate is below 20% (Jiminez & Landgrebe, 1997). Because both architecture and employment can introduce uncertainty into a representation, the number of factors is significant and many of their interactions are poorly understood. Thus, a new approach for interrogating the solution space of complex systems and determining the sensitivity of the SoS to each factor must be identified and put into use in order to effectively begin to manage uncertainty. The use of optimization techniques can aid the management of uncertainty as minima and maxima, points that may cause the SoS difficulties, will be identified through optimization and sensitivity analyses. Additionally, if uncertainty can be managed so too can the risk of a SoS be managed.

2.4   SoS Interrogation Methods

So, one may ask: "If the SoS is so complex and so little of the factor space may be explored, how then do we interrogate a SoS effectively to give confidence in its behavior and performance?" Empirical means

are unable to test the entirety of the problem and solution space therefore traditional approaches employ some level of M&S to interrogate the SoS. The use of M&S here extends to the domains of Live, Virtual and Constructive (LVC) (Henninger, et al., 2008). Most testing domains focus on some combination of these three elements and each has their benefits and shortcomings. A purely Live composition (SoS) may allow exercising of the real system in a small construct of the operational environment. A Live and Virtual composition may allow a wider swath of the ME to be investigated, but there are both practical and accuracy issues in doing so. A purely constructive composition allows interrogation of a wide breadth of the ME, but introduces uncertainty due to the abstraction of the models. Each type of composition is necessary because each interrogates only portions of the SoS. The common method of modeling and simulating the SoS stems from deterministic, physics-based models rooted in the physical representation of system components and rules in the real-world (L,V,C based on replication not representation).

Based on the questions being asked during the interrogation, the M&S method changes because the inherent content of questions and the notion of good enough (i.e., uncertainty) play a significant role. For events like live and virtual tests, the most realistic representation of the system under test is necessary; for training events, a faithful yet abstracted model may be used. However, all of these models have in common that they are rooted in the physical abstraction of reality and are based on deterministic techniques. Based on the sheer number of factors in a complex SoS, it may be inferred that some situations may only be stimulated by using non-deterministic means.

As an observation, we have seen that attempting to interrogate the entirety of the problem space of the SoS using only deterministic approaches can become prohibitively expensive in terms of both fiscal and temporal resources. Eventually, attempting to interrogate the entire mission space deterministically is impractical, if not impossible, and must be augmented with other methods.

2.4.1 Uniqueness

Compounding the difficulties of interrogating the SoS is the fact that the SoS has no single unique outcome. Given initial conditions, the outcome of the SoS may never be the same because of the uncertainties in the SoS. Rather, the SoS produces a range of solutions given stimuli – some of which may be more plausible than others. The spectrum of solutions gives rise to the need to add an additional technique to the interrogation toolbox: stochastic modeling. This means bringing tools into the interrogation domain that allow the use of random and non-deterministic means to investigate the SoS domain space.

2.4.2 Determinism

Part of the difficulties of accurately representing a complex SoS is that the real SoS is inherently not deterministic: there is enough variability in the SoS that given identical initial conditions, the end result, state or behavior of the SoS will not be the same (National Academy of Sciences, 2012).Because SoS's aren't inherently deterministic, most engineers struggle with their representation and interrogation. Deterministic systems and SoS's are typically easier to represent because their outcomes are intuitively a result of a cause-effect relationship, which nicely complements the software built for each L, V, or C representation. However, there may be portions of the SoS that obey some determinism in the form of

14

causal relationships, but these are typically at the macro level and manifest themselves only at the highest levels of abstraction. That being said, initial conditions are rarely, if ever, the same so we need to be able to treat the SoS as a non-deterministic entity.

2.4.3   Stochastic Methods

If complex SoS's are not deterministic, then they are random and/or stochastic in nature. So, what does it mean to be stochastic? In the case of complex SoS's it means that given identical initial conditions, a wide range of possible end states, behaviors and outcomes are possible. This is a direct result of the inherent uncertainty and random processes at work throughout the architecture, possible employment options and the ME. Thus, traditional interrogation efforts which focus primarily on deterministic approaches tend to be unable to investigate a wide range of possible, perhaps even likely, states and behaviors. New methods to interrogate this stochastic space are necessary and by no means do they impinge on the domain of the deterministic tools. Rather, they should complement and leverage the existing methods, exploring the factor and solution space to determine where there may be issues or degradation in performance of the SoS and providing a limited set of conditions to the existing tools to investigate directly.

2.5   Modeling with Graphs

A fairly well-known method of modeling is to use a graph representation of the physical SoS. This representation is the basis for many other types of modeling – DoDAF views (OV-1, OV-2), SysML Context Diagrams, UML Use Case Diagrams, etc. In fact, any model that depicts entities of interest connected to one another is based on a simple graph. In the language of graph theory, each entity is called a *node* and each connection between them is called an *edge*. This can be seen in Figure 6. A graph that has each node connected to all of the other nodes in the graph is a complete or fully-connected graph. Other variations and subsets of this complete graph exist and relay information about the system that it models. A partial graph is one in which not all of the nodes are connected, as illustrated in image (a) in Figure 7. Additionally, arrows can be added to the edges in the graph illustrating directionality of flow (information, current, etc.) between two nodes; this is called a *directed graph* (see (b) in Figure 7.). A directed graph inherently contains more information than a simple partially connected graph and can be used to infer relationships at a macro level. In order to attach more information to a graph, one may extend the concept of the directed graph into two separate constructs. The first is the concept of a directed graph that has no cycles or loops in it. This is called a directed acyclical graph and looks like (c) or a tree graph as in (d) in Figure 7. The second type is the concept of a *directed multi-graph*. A directed multi-graph is a graph that has multiple, directed edges between each node that each represents a unique interaction or flow of information between nodes as seen in (e) in Figure 7. Finally, the notion of *loops* can be added to each node to indicate an internal process within a node, depicted in (f) in Figure 7. (Bondy & Murty, 1976)

2.5.1   Graph Math – Adjacency and Incidence Matrices

Within the context of graphs are the notions of adjacency and incidence matrices. An adjacency matrix describes which nodes are connected to which other nodes. It is an *n-by-n* matrix where each row and column represents a different node. Directionality and multiple edges can be accounted for in an

adjacency matrix by tracing row to column and incrementing the intersection by one for each edge found between those nodes, in that direction. An incidence matrix, on the other hand, is comprised of the nodes on each row and the edges in the graph on each column. When an edge connects two nodes, the corresponding intersections are labeled. In the case of a directed graph, a -1 is used to indicate the tail or source of the edge and a 1 is used to represent the head of the edge. Loops are identified with a 2 to indicate starting and stopping at the same node (other methods assign zero to loops, but we will use the 2 notation throughout). Figure 8 illustrates the adjacency and incidence matrix for a simple directed multi-graph. (Bondy & Murty, 1976)

2.5.2   Physical Space – Link to Mission Environment

The graph model is a way to capture the physical entities in a SoS and their interactions at some level. This "Physical Space" representation is directly linked to the ME in that it directly identifies the physical entities in the architecture and contains some of the information about the interactions between the nodes. So, we can capture the location, orientation, connectivity and possible modes of operation within the Physical Space, but we are unable to explicitly represent interactions between nodes, different stimuli and operating conditions that may exist.

2.5.3   Paths as Mission Threads

In order to expose some of the information alluded to above; we propose building on the concepts of (Garrett, Baron, Moreland, & Anderson, 2012)to use walks, or paths, through the physical space to define mission threads (MTs) (Bondy & Murty, 1976). If we can define the series of events that are necessary to accomplish a MT, then we can claim that mathematically, the set of paths is equal to the MT. This translation allows us to interrogate intra and inter dependencies within the physical space, thereby drawing in the interstitials.

2.5.3.1   Interstitials

If we look at the construct setup in section 2.5.1, we can see that the two constructs (adjacency and incidence matrix) contain information explicitly about the nodes and edges in physical space. Garrett, et al. claim that the off-diagonals in the adjacency matrix are the interstitials or the interactions between entities and can be interrogated as such. We agree and posit that, in addition, the incidence matrix also contains information about the interstitials. In the incidence matrix, we can identify interstitials by the multiple entries in a column. Any column that sums to zero is indicative of an interstitial interaction and must be explored in the problem domain.

2.5.4   Limitations

As mentioned above, traditional models tend to focus on putting the physical entities on the nodes of the graph and their interactions are represented by the edges. This paradigm is followed throughout the traditional modeling paradigm all the way to software implementation. This can be beneficial for certain intended uses; however, the edges on the graph are typically represented implicitly or only at a single level of abstraction – communications modeling.

3 Solution Space

3.1 Mission-Level Engineering

Given a set or type of stimuli the Mission is the collection of tasks, goals and objectives that are to be achieved to successfully address the stimuli. The Mission includes all of the physical assets necessary to meet the goals as well as all of the techniques and procedures necessary to effectively employ them. The mission includes all forests, treess, systems and components that may be employed. The context of the mission also includes the operational environment (Mission Environment) that contains the natural environment, external stimuli and other interactive entities (e.g., "Grey Hat" in the hacking communities are entities who are not necessarily benevolent or malicious). Mission-level engineering requires not only a SoS perspective, but also the ability to integrate the employment of the SoS for multiple stimuli, purposes and missions. Mission-level engineering demands direct interaction with the users of the SoS throughout the lifecycle. Engineering at this level is more complex than traditional Systems Engineering (SE) as it requires a different perspective (Jamshidi, 2009). SE traditionally starts and ends with the user of the system, but SoS engineering at the mission-level must use input from the user at every step of the process. Figure 10 illustrates the cyclical nature of Mission-level engineering and demonstrates that the traditional focus of SE is extended here to focus on integration, interoperability and employment of the SoS.

3.2 Introduction to the Event Space

The concept of the Event Space is simply an extension of traditional physical space representations in which we transform the edges in physical space (integration and interoperability) into nodes in event space so that we can operate on them effectively. Many Event Spaces may be derived from a single Physical Space, but each one contains more information than does Physical Space by itself. Event Space allows the direct inclusion of governance as well as I&I of the SoS and the causal nature of mission prosecution.

3.2.1 From Mission to Mission Threads and Event Spaces

If we define the Mission as all of the entities, interactions and behaviors of the SoS and its ME, we can capture implicitly all of the architectural, employment and stimuli nuances that a mission may encounter. However, exploring this space can be difficult without introducing some bounds and rigor to the analysis. In order to grasp the nuances of the Mission, we whittle down the ME by constraining the architecture to only those items that may be employed within the operational context. From this we can begin to form a set of events within the Physical Space which constitute a Mission Thread or Event Space.

3.2.2 Abstraction

The notion of abstraction is critical in the proper formation of Event Space. As opposed to traditional Physical Space representations, Event Space construction cannot be done from constituent component functionality or construction up to the SoS. We must start at the top of the SoS and clearly define the depth necessary to represent Physical entities and Events based on the questions that we are going to ask of the model. We propose a balanced, holistic, approach for determining adequate abstraction for

17

the SoS comprised of entity abstraction, functional abstraction and I&I abstraction. As depicted in Figure 11, the three paradigms of abstraction each represent a critical, non-exclusive component of abstraction that must be accounted for when developing a SoS conceptual model.

Defining the entities and how they are grouped is a critical aspect to gaining an adequate and consistent level of abstraction in the conceptual model. We categorize real-world entities into five major functional groups: 1) sensors, 2) actors, 3) controllers, 4) stimuli, and 5) environment. There are many ways to decompose entities in this manner; however, the critical aspect to any decomposition should be a common, consistent depth of abstraction held across the SoS. Typically, the system under test that we are asked to represent is primarily decomposed into actors, sensors and controllers only. Many have asked us why we chose this paradigm for demarcation in the model commenting that many of the elements in the system under test are aggregate systems, each containing a module or sub-system that provides the functionality of a sensor, actor or controller. The reason we take this approach is to provide a consistent boundary in the conceptual model. Many times we have seen models constructed through direct representation of real-world entities fail because of their inability to delineate between functions within the entity. With this in mind, we break real-world entities along their functional lines in order to provide a single, consistent level of abstraction for the conceptual model.

To approach functional or event abstraction we leverage the concept of an Observe, Orient, Decide, Act (OODA) loop first proposed by Col. John Boyd (Boyd, Destruction and Creation, 1976) (Boyd, A Discourse on Winning and Losing, 1984) (Boyd, Patterns of Conflict, 1986) to help guide the level of abstraction. OODA is an iterative, recursive process that allows the prosecution of stimuli with minimal resolution. Given the typical questions asked of a SoS ("What is the optimal deployment strategy of the SoS?", "What is the optimal employment strategy of the SoS?", "How many stimuli can the SoS interact with prior to "breaking"?"), we need the minimal amount of information from the model to address this. To answer these questions, we have extended the OODA formulation and decomposed each top-level function into a set of functions that can be used to group, extract and portray events in Event Space. These decompositions can be seen in Figure 12, Figure 13, Figure 14 and Figure 15, and will be used as a basis for abstraction of a conceptual model in section 4. Our interpretation of Boyd's work leads us to apply the OODA construct such that the Orient function is resident in each actor sensor and controller. As shown in Figure 13, the Orient function is much more than pure communications (i.e., notify functionality) and includes functions like discerning, correlation and fusion. This approach requires each actor sensor and controller to perform local Orientation functions prior to sending and after receipt of a notification from another system within the SoS. As complexity increases, this construct requires that both local and enterprise (MLSoS level) OODA functions must occur simultaneously. This concept is difficult to accurately capture and can lead to significant issues in the design, development, testing and employment of SoSs. As we begin to ask more detailed questions about sensitivities and optimization, a lower level of abstraction can be employed, but it should be driven by the objectives of model employment, not by the bottom-up construction of the system.

The third tenet of conceptual modeling that we propose is to employ the constructs of Autonomy, Collaboration, and Cooperation (ACC) (Surowiecki, 2005) on OODA and the entities as an I&I Abstraction. By defining interactions and interoperations on a common scale, we can better indicate

18

which components are critical in the formation and prosecution of the SoS. Figure 16, Figure 17, and Figure 18 depict a decomposition of all three ACC abstractions, which will be used throughout the remainder of this paper. Typically, entities that are functioning independently, or with only moderate interaction with the rest of the environment, are classified as autonomous. This classification can be applied to the system as a whole or to the function the system is performing during an OODA cycle. Entities which share information for the purposes of situational awareness are typically categorized as being involved in a collaborative relationship. And, those entities which share information for the purposes of directly acting on that information to achieve a goal they could not achieve independently are said to have a cooperative relationship. The ACC construct explicitly allows for dynamic inclusion of Autonomy, Collaboration and Cooperation across the SoS to optimize and/or maintain performance based on stimulus density and environmental feedback. As discussed earlier, this feeds directly into the syntactic and semantic composition definition of I&I.

3.2.3   Directed Acyclical Networks – Using Bayesian Networks

Physical space graph representations provide great insight into the physical architecture of the system, but fall short when attempting to identify all of the interactions that occur between each component of the SoS. Traditional representations of physical space (in an OV-1 or a standard network graph) the interactions between two nodes are identified only as a single edge (maybe directed, maybe not) with perhaps some information contained on that edge. One way to extend past the single edge limitations is to allow multiple directed edges between each node in the physical graph. This allows the modeler to define directionality of the edge as well as content for all interactions between two nodes. The resultant multi-graph can be interrogated for pair-wise investigation of interaction, but there are no causal representations between each node or within the graph as a whole.

The move to a directed, acyclical graph, DAG, allows us to maintain all of the interaction information of the multi-graph, as well as the causal nature of the graph that would be lost (or at least documented another way). The DAG is essential to portray causal relationships and to operate on the network using Bayesian methods. The DAG typically appears as a tree (or forest if multiple entry points) and each of the nodes in physical space are replicated multiple times to capture the tree structure of the SoS DAG. This limitation (multiple representations of the same physical entity) provides us a reason to pause and consider the necessity of the DAG. If we add another step to the creation of the Event Space, we can transition to a DAG not of physical nodes, but rather of the events that transpire between those nodes – we can extract a much simpler DAG with individual PDFs as the drivers of model performance. This DAG is, when married with nodal probability information, a Bayesian Network (BN) (Held, 2008). BNs have been used to model causal chains of events in the past and more recently, are used to perform inferential analysis and prediction given certain preconditions. BNs follow a paradigm introduced by Thomas Bayes in 1763 that leverages the causal dependencies related through conditional probabilities. As discussed in the next section, this provides a unique, novel method to capture SoS metrics for every event within the Event Space.

3.3 Metrics

In order to adequately quantify the performance and behavior of the SoS, one must introduce adequate metrics that capture all of the critical aspects of the composition and employment of the SoS. Traditional cost and schedule metrics are necessary and will be ignored for this discussion. We are interested in getting at Mission-level metrics that describe performance, behavior, integration and interoperability.

There are two metrics that we propose for component and SoS performance and behavior that are fairly generic which operate at the system and SoS level: probability of success (P_s) and breaking point (B_p). The probability of success, P_S, is the chance of the SoS's goals being achieved. The breaking point, B_P, is the number of stimuli acting on the SoS which cause the SoS to stop functioning or the P_S to fall below an acceptable threshold. The P_S can be represented as a string of independent probabilities for each function, process or event to give a bound to the performance of the system.

In a simplistic sense, P_S can be constructed as the probability of a component or an event executing properly. Within the SoS, these probabilities can be strung together many different ways to develop an overarching SoS P_S. Utilizing an adjacency matrix approach for capturing the interstitials as described in section 2, one can access the P_S of the interstitials and provide a more robust description of the SoS performance. However, simple multiplication of probabilities, indicating their independence, betrays the causality of events and functions within the SoS. In the next section, we will demonstrate how to utilize P_S while incorporating the interstitials as well as the causal nature of the SoS.

The B_P for a SoS can be as easily crafted as the P_S. If we stimulate the SoS with multiple stimuli, we can monitor the P_S of the system to determine when it falls below a threshold value. The B_P then corresponds to the number of stimuli it takes to make the system fail in the sense of unacceptable performance. There is a hard limit for B_P too, B_{PMAX}, which is where the system can no long function in the manner intended. Typically, this corresponds with a knee in the B_P curve at which point all P_S go to zero – this is a change in sign of the second derivative of the P_S against multiple stimuli. The breaking point indicates a bifurcation of the performance space that can lead to many different behaviors: branching, discontinuities or chaotic behavior (Garrett, Baron, Moreland, & Anderson, 2012). A sample representation of P_S is given in Figure 19.

System performance metrics have been the primary tool for system and SoS evaluation since the beginning of systems theory. However, we believe there is a set of complementary metrics to specifically and quantitatively address I&I. As we discussed in section 2, there are two primary classes of I&I with three subclasses: Syntactic and Semantic Composition which are broken into Autonomy, Collaboration and Cooperation. To that end, we propose the following Event Space specific metrics build on both P_S and B_P to measure the I&I of a SoS: Probability of Realization (P_r), Level of Integration (L_i) and Degree of Interoperability (D_i). These metrics are along the lines of those presented in Garrett, et. al., are event space specific, and ultimately build to a rigorous form of Integration and System Readiness Levels proposed for physical space by Sauser et. al. (Sauser, Ramirez-Marquez, Verma, & Gove, 2004)and ultimately, into an Enterprise Readiness Level that encompasses the entire Mission domain (Garrett, Baron, Moreland, & Anderson, 2012). The P_r measure intends to gauge the divergence

20

from the expected intended use, IU, of the SoS. The IU and associated goals (defined perhaps as an objective and threshold P_S and B_P) of the SoS are to be determined and documented at the time of design. During development, integration and testing, the SoS will be measured against the goals of the IU. The Stolarsky mean (Stolarsky, 1975) (Simic, 2009) measures the deviation from the acceptable values will be the P_r. The Degree of Interoperability represents the amount of cooperation in a given SoS with regard to the mission objectives. D_i should not be mistaken for all possible interoperability within the SoS, rather $D_i = 1$ corresponds to interoperability commensurate with the cooperation necessary to perform system functions. D_i represents a change in the fundamental architecting of SoS's and requires the objectives and mechanisms to meet those objectives be defined at the onset of the architecture and design process. On the other hand, L_i demonstrates the amount of integration a system is capable of, given the goals and intents of the SoS. L_i corresponds to an Integration Readiness Level, mapped to a percentage scale. L_i is driven by the traditional levels of the open systems interconnection (OSI) model, (ISO/IEC, 1994) requiring integration through level 7, ensuring that messages are correct at the protocol through the content layers.

3.4   Event Space as a Modeling Technique

When we attempt to represent physical systems in the same manner as we perceive them, we run into a problem akin to the old adage about problems cannot be solved by the same conscious that created them. The problem with physical space is that it focuses on the nodes of the SoS – the individual components – and leaves their interconnections to be modeled implicitly, each end of the interconnection modeled (perhaps) by one component, but the middle left to any number of possibilities. The thought is then that we need to be able to explicitly investigate the interactions of the components to determine how those interactions may drive the behaviors of the SoS.

In addition to being able to explicitly represent and investigate the interstitials, we also need a method that manages and explicitly accounts for causality of events between components. Attempting to leverage the utilities and exploit the failings of traditional Physical Space, we looked for a transform to take us from one space to another, much like that of real and reciprocal space in solid mechanics or time and frequency domain in Fourier transforms.

In order to operate on the edges in physical space, we utilize a directed acyclical graph (DAG) to capture the edge events from the physical space multi-graph into a pseudo-linear DAG (Held, 2008). We named this DAG the Event Space as it captured the events on the nodes of the DAG and transitioned the "contents" of the nodes in Physical Space to the edges in Event Space. Thus we can capture explicitly the interstitials and the causality of physical space in a form that is readily operable. Figure 20 is an example of a physical directed multigraph.

The transition from Figure 21 to Figure 22 notionally illustrates this process. The directed multigraph in Figure 23 represents an autonomous system composed of an organic actor, controller and sensor; this system is then integrated to a remote sensor net. The remote sensor net comprises a controller and two spatially separate sensors. The autonomous system is free to use the remote sensor data to augment its organic sensor as it engages 5 stimuli. The edges indicate the rich space of potential interactions. Figure 24 depicts four possible Event Space transpositions from the Physical Space. Each

21

space contains the content of the Physical directed multigraph as well as causal dependencies that result from the employment of the Physical space in the ME. Based on the construct of ACC and stimulus and employment variations, we can see in Figure 25 and Figure 27 how each of these makes a distinct impact on the structure of event space.

If we were to assign each of our nodes a P_S, we could easily see that the DAG (Event Space) explicitly resembles a BN (Held, 2008). Using the conditional probability rules of BNs and the P_S assigned to each of the nodes, we can explicitly compute the conditional probabilities for any point in the event chain, allowing the exploration of the entire SoS in a single space (Figure 19). Computing many P_S for the network as a whole allows us to determine the B_p as well, knowing that the B_p is the sign change in the second derivative of the P_S curves.

3.4.1 Probabilities

Each P_S used in the Event Space is a continuous PDF, allowing expressions of behavior of the event. However, we know that most behaviors don't follow a typically Normal distribution or even one of the more "exotic" distributions (Weibull, Logarithmic, Exponential, etc.). Often, because of the multiple factors that underlie each behavior, their PDF is "bumpy", that is they have defined substructure. In order to approximate these substructures, we can use linear superposition of Normal PDFs as in Figure 28 or by using estimations and inverse problems (Banks, Kenz, & Thompson, 2012). This approximation allows us to effectively employ a PDF for each event.

Now, we need to know what the PDFs for each event are. There are two methodologies to do this. First, we can start with first principles and create the PDFs from expert knowledge, basic math and physics. This method will give us a rough order of estimate for the event within the Event Space, but it is resource intensive and subject to their own types and levels of uncertainty. The second way we can develop these PDFs is to make informed estimations by employing existing data that relates to the events in the form of test data, assessment (M&S) data, training and exercise data or exploratory data. All bring a level of inherent uncertainty with them, but typically, the data is validated and/or certified for an Intended Use and can be leveraged to benchmark the data used in the Event Spaces with only small resource investments.

Because we are dealing with the stochastic nature of the SoS and constantly capturing and quantifying uncertainty, we need not know all of the PDFs with 100% certainty. What we can do; however, is create PDFs utilizing both techniques outlined above within given resource means, and run them in parallel to ascertain deviations and similarities in the two. Deciding which is correct may be more difficult and choosing a method to select the estimate generating model leaves room for uncertainty, but it can be readily quantified (Banks, Kenz, & Thompson, 2012).

Another complementary approach using minimal resources to finding PDFs and the Bayesian technique is to utilize a Frequentist (Mayo & Cox, 2006) approach where we look at actual data and determine a relative frequency for each event. By doing this we can circumvent some of the issues with defining a PDF for each event; however, we will use the relative frequency in much the same manner as the data pulled from the PDFs.

22

$$P(X) \approx \frac{n_x}{n_t} \tag{1}$$

or

$$P(X) \approx \frac{\left(n_{low} \leq n_x \leq n_{high}\right)}{n_t} \tag{2}$$

We will use it in the Bayesian formulation where we need to gauge dependence. Departing slightly from the Frequentist approach; however, we use a range of occurrence values to assess the impact on the overall SoS to ensure sensitivity.

3.4.2 Causality

Due to the nature of the employment of complex SoS, capturing the dependence between events in the space is critical to ensuring behaviors and their stimuli are accurately captured. By transitioning from Physical Space to Event Space, using OODA and ACC, we can begin to capture the dependence on the behaviors of the SoS. Utilizing the Bayesian construct to contain the Event Space naturally allows the explicit inclusion of causality into the computations of the metrics (Held, 2008). This is important as many traditional component and system metrics tend to be computed using independent probabilities which may cast an overly negative tone (or at least inaccurate) on the true behavior of the SoS.

Because of the causal nature of the Event Space and the Bayesian structure we have used to capture it, we can utilize Bayes' rule of conditional probability:

$$P(B|A) = \frac{P(B \cap A)}{P(A)} \tag{3}$$

to determine the outcome of the network at any node. In theory, this sounds nice and straightforward; however, in practice, determining the multi-variate probability density functions may be impossible or at least impractical. A set of conditional probability tables may be built for each node:

$$P(B|A) = \begin{bmatrix} A = T & A = F & B = T & B = F \\ 1 & 0 & 0.98 & 0.02 \\ 0 & 1 & 0.45 & 0.55 \end{bmatrix} \tag{4}$$

outlining all of the possible input states and resultant nodal probabilities. For a large, complex SoS with many events, this could be daunting. We have devised a simple, linear approximation to account for causality given a single parent (the serial case, Figure 29) (McCabe, 1968).

$$P(C|B) \approx \varphi\big(P(B)\big) + \big((1 - \varphi)P(C)\big) \tag{5}$$

This approximation utilizes a normalization factor, φ, which can be approximated or even optimized to devise the best employment and causal dependence between events and physical entities.

$$\begin{aligned} P(A, B, C) &= P(C|B)P(B|A)P(A) \\ &\approx \big(\varphi P(B) + (1 - \varphi)P(C)\big)P(B|A)P(A) \end{aligned} \tag{6}$$

23

In order to determine how multiple nodes stemming from a single parent may impact the outcome of a child node, we must borrow a formulation from the reliability community (START, 2010). We utilize a parallelization schema in addition to a normalization parameter to determine the impact on the child node.

$$
\begin{aligned}
P(F|C,D,E) \approx \Big(&\varphi\big(1 - (1 - P(C))^{w_c}(1 - P(D))^{w_d}(1 - P(E))^{w_e}\big) \\
&+ \big((1 - \varphi)P(F)\big)\Big) P(C|B)P(D|B)P(E|B)P(B|A)P(A)
\end{aligned}
\tag{7}
$$

By weighting each of the parallel nodes, w_i, we may also address the amount of influence each one exerts on the child. Similarly, if we have multiple, non-related or distantly related parents influencing a child (Figure 31) we can use the multi-variate, multi-normalization parameter formulation of (8) to determine the outcome of the child node.

$$
\begin{aligned}
P(D,B,C) = P(D|B,C) \\
\approx \Big(\varphi_B\big(P(B)\big) + \varphi_C\big(P(C)\big) \\
+ (1 - \varphi_B - \varphi_C)\big(P(D)\big)\Big) P(B|A)P(C)P(A)
\end{aligned}
\tag{8}
$$

where
$$
\sum \varphi_i = 1
\tag{9}
$$

3.4.3   Dealing With Multiple Stimuli – Convolution

Thus far we have dealt with capturing how an SoS may react to a single stimulus, but we know that real-world systems must interact with far more than one stimulus at a time. The way that we capture the SoS's ability to interact with multiple stimuli is to convolve multiple, single stimuli event spaces into a single complex Event Space, i.e., a mission thread, which provides the computation of the probability of each stimulus given the others. This means being able to create new edges between two event spaces to create complex networks.

We can create these additional edges by exploring the similarity of events (based on OODA, ACC and the entity tenets of abstraction) in each event space and determining if they exist on the same physical node in Physical Space. If they do, and the nodes are sufficiently "close" (temporally separated) to one another we can craft a new edge between them, Figure 32. This new edge may cause a degradation or improvement in behavior due to the nature of the interaction. Being able to convolve multiple event spaces into a single Event Space network, we can begin to explore employment options and "raid" sensitivities. This can be seen in Equation (10).

$$
G_3 = G_1 *^{\Delta}_{\delta} G_2
\tag{10}
$$

The asterix in equation 10 indicates a convolution of the two networks with Δ, and δ indicative of an offset between the two networks and the acceptable temporal influence duration, respectively.

As seen in Figure 33 there are direct links between the entity, functionality and I&I aspects of each node that require the two networks to be convolved. Simply stated, there is dependence between events

24

because each event occurs on the same physical node which may cause a behavioral or performance variation. Basically, convolution of two networks attempts to find a set of edges that connect the two networks, thereby allowing the computation of the impacts of multiple stimuli concurrently.

$$G_3 = [V_1 + V_2, E_1 + E_2 + E_*] \tag{11}$$

where,

$$E_* = \left\{ e_{ii}^*, e_{ij}^*, \dots, e_{jj}^* \right\} \tag{12}$$

In equations 11 and 12, the notion of E_* is that there are new edges formed in the graph which are not part of the original set of edges, E_1 or E_2. These new edges provide links between the convolved networks and begin to insert non-linear behaviors that are expected between two networks.

Determining the new edges is done by searching for common functions or types of events that fall within an influence boundary and then linking them with a new edge originating on the earlier node and terminating on the later node. This algorithm is described below:

$$\forall v_i \in V_1, v_j \in V_2 \tag{13}$$
$$e_{ij}^* = E(v_i, v_j) \tag{14}$$
iff
$$v_i \equiv v_j \tag{15}$$
and
$$v_j(t) + \Delta + \sum e_j(t) = \left(v_i(t) + \sum e_i(t) \right) \pm \delta \tag{16}$$

The ability to interrogate multiple stimuli interacting with a SoS provides great insight into the behavioral and performance ramifications of the dense stimulus environment.

3.5 Agents

Agent-based modeling has been a staple for modeling and interrogating complex groups of autonomous entities for nearly two decades. Agent-based models utilize simple, heuristics based representations of independent, autonomous entities to investigate how groups or supply chains may operate. Agent-based models are traditionally built from the bottom-up with no central authority or controller for how the system operates or is transitions from state to state.

3.5.1 Hyper-Agents

We propose an extension to traditional agent-based modeling techniques in the construction of Hyper-agents. Hyper-agents encompass many (if not all) event spaces derived from a physical space for a SoS. Through convolution, Hyper-agents have the ability to look at SoS-level interactions and all of the permutations therein while being stimulated by numerous external entities. Through artificial intelligence (AI) and decision logic, Hyper-agents can explore complex SoS architectures and employments. The Hyper-agent approach allows the interrogation of SoS-level metrics and behaviors to determine mission thread sensitivities that can be subsequently investigated by legacy, physics-based M&S tools. A simulation built around Hyper-agents can embed tens to hundreds of SoS representations into a field of stimulus agents to efficiently explore many mission-level permutations.

3.6   Optimization and Architectural Ramifications

Today, M&S is really quite good at investigating system-level component and system solutions based on well-formed, physics-based models. However, there is a need to explore new capabilities in the SoS and to explore both materiel and non-materiel approaches to address user needs. Multivariate optimization combined with the Hyper-agent construct can allow both architectural and employment optimization over the solution space. Optimization with evolutionary algorithms operating on the Event Spaces in Hyper-agents allows exploration of gaps, morphing architectures and employment strategies to meet the needs of a user. Within the realm of SoS Engineering and SE, the engineering and employment communities can both provide requirements and solutions that optimize cost and performance and minimize uncertainty in the behavior of the SoS.

4   Discussion – Uncertainty is the running thread throughout MLSoSE

4.1   There is No Single "Right" Answer

As we constructed this methodology to build and explore a conceptual model of complex systems of systems, we came to the conclusion that, because of the uncertainty in the system (from architecture to employment) there is no single "right" answer. Rather there are many answers that will suffice; bounded by both architecture and employment, e.g., independent versus conditional relationships. So we need to explore a range of solutions, look at trends in the solution space that indicate a level of goodness, and then determine what our error budgets in the SoS are. In other words, answer the question "what is good enough?"

Because of the probabilistic approach we have taken, the ultimate goal of the methodology is to demonstrate the sensitivity of the SoS to each factor in a multi-factor space in order to limit the problem domain that complementary, physics-based approaches must explore. Additionally, if we understand the trends of the SoS and the factors that the SoS is most sensitive to, we can better architect the physical system as well as the employment of the system once it is developed and deployed.

A conceptual model has been built and exercised over a Physical Space and several single stimuli Event Spaces presented previously in Figure 23 and Figure 24. The conceptual model identifies multiple sensors, actors and controllers all working to prosecute stimuli. The resultant event spaces in Figure 24 show variations in stimulus as well as employment and architecture of the SoS. These variations are highlighted in Figure 25 and Figure 27 and explicitly capture the nuances of employment, stimulus and architecture that physical space representations cannot. Applying the techniques described in section 3.4.1, we applied a PDF to each node in event space and a random correlation variable, φ, to each dependency. Drawing randomly from each PDF, we computed the approximations for causal probabilities for each node in Event Space and a traditional independent estimation of the network alongside the conditional computations which can be seen in Figure 33, Figure 34, and Figure 35. Additionally, we performed a two stimuli and three stimuli convolved Event Spaces to illustrate how convolution is performed, the results of which can be seen in Figure 36 and Figure 37, respectively. Figure 38 and Figure 33 illustrate the details of each convolution; indicating in Figure 38 where

26

controller (grey) and sensor (green) convolutions may occur and the abstraction relevance to convolution of two sensors in Figure 33.

From these computations, we note in Figure 33, Figure 34 and Figure 35 that there is a wide range of possible answers for the behavior of the SoS between the independent and dependent curves. We can also see that the level of dependence on prior events and the level of autonomy, collaboration and cooperation through the architecture (i.e., employment strategies) drive trends in the SoS far more than the individual nodal probabilities themselves. This is more apparent as we convolve multiple networks in a multi-stimulus environment as in Figure 36 and Figure 37. In the convolved results we can also see that there are benefits and consequences to increasing the complexity of the Event Space architectures. We believe that as we investigate more networks we will identify that there are certain breaking points along the lines of ACC abstraction in convolved networks. Thus, we can attempt to better architect the employment of the SoS and we can begin to appropriately bound the range of solutions for an SoS given a set of error budgets. From this, we can ultimately eliminate those variables, conditions and interactions that have no impact on the SoS and minimize those that do.

4.2   Lessons Learned - Structuring the Conceptual Model

Before we can get to the point of determining how the architecture and employment of the SoS impacts its overall behavior, we must first appropriately structure and develop the conceptual model through the Mission Environment and the Mission Threads in a holistic, top-down manner.

From our previous experiences, the authors have come to realize that many times detail (too much or too little) becomes detrimental in Mission-Level SoS Engineering. One must construct the conceptual model with common and consistent abstraction given the objectives (questions being asked of the model). The best way that we have found to do this is through the tenets of abstraction: entity, functional and I&I abstraction.

Once we have identified all of the entities, characterizing them as actors, sensors or controllers, and their possible interactions in the physical domain, we must then extract the walks that define the mission threads and event spaces. Because there is a seeming plethora of methods to extract this information, we decided on the most complete, yet generic "kill chain" that we could come up with: OODA. Following OODA allows us to recursively formulate, to whatever level of abstraction we desire, the appropriate events in event space or walks in physical space. Similarly, we can also classify events derived from the application of OODA to physical space according to the taxonomy of Autonomy, Collaboration and Cooperation (ACC). In doing this we can appropriately bin, not only what type of interaction we are dealing with, but we can bound where to find supporting data and what data we may need to describe the PDF of that event. The ACC taxonomy can be seen in Figure 16, Figure 17, and Figure 18. The impact of ACC can be seen in Figure 25 and Figure 27 as simple and slight changes in employment or stimulus can cause dramatic structural changes in Event Space. None of these variations are easily or explicitly identified in physical space representations.

Through convolution, based on abstraction, we can see that the ACC paradigm gives us insight into what may or may not drive the overall behavior of the SoS. Convolution of multiple event spaces must be

27

done for each type of abstraction. As seen in Figure 33 the likeness in entity, similarity in functionality (as well as dependence on resources) and equivalence in I&I dictates how the convolution will be performed. From this concept, we can begin to explore degrees of convolution and their impact on the overall SoS. Mathematically, we can represent these degrees in the correlation factor, φ, in the computations, varying it over a wide range to ascertain the impacts to behavior and to tune the conceptual model to observed data.

The goal of the conceptual model is to be able to execute it on a white board or as a game. There is no need to code the exercise of the conceptual model as the model should be complete, well-bounded and executable. The methodology that we have described above provides the ability to create a set of models that describes the space of the SoS and then exercise them in concert when multiple stimuli or scenarios are posed against the SoS.

4.3 Dynamic Nature of the Mission-Level SoS Composition

A key component in a MLSoS is the ability to dynamically evolve within the ACC domain in order to maintain performance based on environmental factors, e.g., increasing stimulus density. This can be accomplished by simultaneously managing across the local and enterprise entity states and functions after appropriate integration and interoperability engineering and testing have been accomplished. The vision of the balanced MLSoS is an initial tendency toward Autonomous behavior. Based on the environment, particularly stimulus density, the SoS will evolve through collaboration to cooperation as sensors, controllers, and ultimately communications saturate (both locally then centrally for the enterprise SoS). The distributed nature of the Orient function is a key in the SoS architecture to achieving dynamic re-balancing between autonomy, collaboration, and/or cooperation. Frequently, focus on the Orient function centers around robust information networks and standardization of message formats. While the attention to syntactic composition is necessary, the authors believe it is insufficient. Interoperability must be achieved by focusing not on the interface itself, but around the interface and through the SoS to insure semantic interoperability. This implies the engineering of interface standards must not only insure that message content has consistent meaning on both sides of the interface (beginning with the local orient functions) but ultimately that the semantics are maintained across the MLSoS.

4.4 Relationship between Architecture and Mission Environment and Mission Threads and Physical and Event Space

There are a number of different sources of uncertainty which stem from the architecture of the SoS to its ME. Because of the complexities of any given SoS and its ME, one must be able to draw an atomic line between the architecture, ME and resultant Mission Threads in both Physical and Event Space. Both the physical architecture and the employment of that architecture define the performance and behavioral space of the SoS. Therefore, each change to the physical environment must be transferred to the MTs and each change to the employment strategy must be mapped back to physical space when there are changes in laydown or performance parameters of any of the individual elements.

The ability to map back and forth between spaces provides a necessary, complementary technique to focus legacy M&S tools to only the points of interest within the SoS with real, reasonable boundary conditions for those investigations. Using the tools and techniques described herein, one can easily investigate a large combinatorial space and provide bounds to any M&S tools that may be necessary to investigate at a lower level of abstraction. Executing the map between physical and event space allows us to effectively interrogate a wide range of employment and architectural differences within a short period of time and manifest the results in a meaningful way to both engineers and users of the SoS. This makes for an extremely rich Conceptual Model that can inform all aspects of the Mission-Level SoS Engineering process as well as the lower level component SE processes.

4.5 Employing the Conceptual through Simulation

Because of the richness of the Conceptual Model and the heuristics described in this paper, we can effectively exercise the model over many permutations of the stimulus and response spaces to gain insight into performance and SoS behaviors. Because the Conceptual Model is built to represent many possible reactions and combinations of reactions to stimuli, there is a need to perform analysis not at the individual MT level, but at the aggregate SoS level. This type of analysis focuses on many different variables, each computed through the heuristics of the Event Space and the questions that are being asked of the model. Additionally, because we are focusing on SoS-level behaviors, we must not look for single valued answers from the Conceptual Model, rather an analysis approach that focuses on being able to infer "goodness" based on the trends of SoS performance and behavior. This allows the use of the Conceptual model for efforts to optimize on both the upper right *and* upper left sides of the classic System Engineering "Vee" or through the entirety of the MLSoSE lifecycle (Figure 10).

4.6 Getting at the I&I Metrics

From the P_S and B_P metrics computed for each Event Space and for each SoS, we can start to develop a method to determine the I&I metrics: Probability of Realization (P_r), Level of Integration (L_i) and Degree of Interoperability (D_i). As we investigate the sensitivities of the SoS we will find that for each layer of ACC, there will be tradeoffs to the overall success of the SoS. This starts by assessing L_i for autonomous and collaborative interactions and their impacts on the behavior of the SoS. As L_i corresponds to an Physical Space Integration Readiness Level, we can immediately assess the IRL of the SoS. Building up from IRL, we can also investigate the D_i and the collaborative and cooperative interactions and develop a Physical Space SRL to grade the SoS. Utilizing L_i, D_i, P_S and B_P we can ultimately assess the probability of realization, which corresponds to a Physical Space Enterprise Readiness Level (ERL), to grade the SoS as a whole.

4.7 Implications to SoS Acceptance; VVA&C –

4.7.1 Processes

Current system level Verification, Validation, Accreditation and Certification (VVA&C) processes address a very component centric, bottoms-up approach to each VVA&C activity. These processes assume that each individual entity is solely responsible for I&I and they should be covered in the VVA&C artifacts delivered to the SoS. From these artifacts, the attempts to build mission level SoS VVA&C processes claim that SoS VVA&C is a simple linear extraction and aggregation from the components and can be

29

accomplished with little to no testing or investigation at the SoS level (National Academy of Sciences, 2012) We believe that the construct of MLSoSE and the methods to achieve it can provide for a paradigm shift resulting in a means to achieve the recommendations of the National Academy (National Academy of Sciences, 2012)

4.7.2   Limitations

We contend that the extensibility of current component system approach is severely limited when it comes to SoS VVA&C. The assumption of linear extrapolation from component VVA&C artifacts is at best insufficient. Each component is built with a specific set of intended uses that may not necessarily align with the intended uses of the SoS. Additionally, the syntactic and semantic portions of the SoS I&I, and the SoS uncertainties are not traditionally fully encompassed by the component VVA&C activities; therefore, their artifacts won't/can't provide sufficient information to complete an SoS VV&C process. We posit that because of the inherent complexities of the SoS and its different intended uses, the VVA&C activities for the SoS are far more than simply the linear sum of its parts. Because of this, special attention must be paid to the intended use of the SoS (under the construct of Mission Environment and Mission Threads) and the composition of the SoS during the VVA&C activities.

4.7.3   VVA&C and the Pyramid

At each level of the pyramid, VVA&C is different and requires a unique (or at least modified) process to accomplish its goals.   At the system level, VVA&C can be well accomplished with traditional SE techniques and provides a robust methodology for delivering and fielding functional systems. At the Trees and Forest Levels; however, the VVA&C processes of the component level start to fall apart because they do not put as much attention on syntactic integration as necessary nor do they address semantic integration. At the Mission-Level, VVA&C techniques further fall apart due to the uncertainty inherent in the Mission Environment. Once at the Campaign level, dealing with multiple missions, the ability of VVA&C techniques that currently exist to actually capture any of the complexity and uncertainty is near zero. The bulk of this is due to the different intended uses and I&I issues.

4.7.3.1   Multi-level VVA&C

All of that being said, there are vertical components to VVA&C that must exist between levels in the pyramid: constraint comes from the top and support comes from the bottom. In order to provide effective SoS VVA&C each level of the pyramid must complement those above and below it while retaining their own uniqueness. The key to accomplishing VVA&C across multiple levels is alignment of intended uses and quantification of uncertainty. This essentially reduces to an accurate capture and description of the component, system and SoS capabilities and limitations with regard to the SoS intended use.

4.7.4   Acceptability

The notion of acceptability is at best a two-tiered function for the SoS. At one level, there is the engineering acceptability of the SoS (i.e., the SoS does what it should when it should do it and all of the requirements in the ME and MT have been met) and at the other is the user's acceptability of the SoS. While acceptability cannot be met without both tiers, special emphasis has to be put on the user's acceptability of the SoS. The SoS employment is expected to elicit particular responses or display a

30

certain set of behaviors under employment constraints. Many, if not all, of these user-based needs are ignored during engineering acceptance and must be a critical component of SoS VVA&C.

5 Conclusions and Future Work

5.1 Definition of Mission-Level SoS

We have defined a consistent definition of a SoS that articulates the nuances of the difference between a system and a SoS. We propose that the SoS is:

- Composed of independent constituent systems that maintain their own internal management
  - The component systems bring forth their engineering artifacts, models and simulations
  - Verification, validation, accreditation and certification for their intended use
  - Independent component operational governance
- Composed into a top-down C4 architecture
- Able to achieve its mission goals through the application of top-down governance defined by the Mission Environment
- Able to employ Mission Threads facilitate the development of the SoS architecture and establishes the subsequent testing domain
- Dominated by the integration and interoperability (interstitials) and *not* the detailed behavior of the individual constituent systems; these interstitials have inherent uncertainty due to the nature of the mission environment and C4 based architecture, thus are stochastic in nature
- Can dynamically re-balance the SoS composition both locally and at the enterprise between autonomy, collaboration and cooperation based on the mission environment to achieve mission goals.

The last bullet can be achieved with the advent of the Event Space construct. An event Space allows for a unique set of metrics. By in-situ monitoring of P_S and B_P metrics and tracking the evolution of Probability of Realization (P_r), Level of Integration (L_i) and Degree of Interoperability (D_i) dynamic balancing across the SoS can be accomplished. An example could be after employment of the SoS composition the dynamic rebalancing process is achieved by a flexible central SoS C4 architecture including a sensor network, C4ISR, monitoring the mission environment, and a series of entity systems (integrated actor/sensor/controller) monitoring the local ME. Most likely the entity systems will initialize in an autonomous state. Local system 'saturation' due to increasing stimuli loading could/should drive the local system to request SoS support from the central C4ISR moving through collaboration and ultimately to cooperation. As the stimuli density increases the SoS may saturate in terms of sensor-net and/or communications capability. This enterprise saturation could then drive the entity systems back toward autonomy.

This definition and set of conditions can be used to define any SoS within any domain ranging from commercial supply chain management to air traffic control to any DoD construct. This common definition should be employed to define the difference between systems and SoS's.

31

5.2   Implications for Real World Problems

Because the Intended Use(s) of the systems that comprise the SoS are not necessarily the same, or even aligned with the IU of the composition (the SoS for the Mission) there must be a methodology to capture those differences and architect an SoS with those deviations in mind. It is in this way that Mission Level SoS Engineering is distinctly different from traditional Systems Engineering. Thus the lifecycle engineering of a SoS must be treated differently than that of an ordinary system. Specifically, in the MLSoSE paradigm, requirements are derived from the mission context as conditions, behaviors and operating environments vice the SE requirements which consist of many *will* and *shall* statements that are intended to elicit specific functionality vice mission-level behaviors. The MLSoSE paradigm also provides a methodology and quantification of statistical bounds on requirements for the SoS. This leads directly to V&V and can help render a SoS flexible, rather than fragile.

While this discussion may seem esoteric in points, the implications for problems encountered in engineering today are profound. With the MLSoSE construct outlined in this paper, engineers now have the ability to interrogate and ultimately optimize not only the physical structure of the SoS but also the employment thereof. This technique is already being tested against problems during both the early requirements definition and design phases as well as for assessing behavioral and performance anomalies. The technique allows exploration of component and SoS architectural concepts as well as development, optimization and evolution of employment strategies, processes and procedures. As we discussed, uncertainty quantification is critical to increase confidence and reduce risk. Through the exploitation of this technique along with some of the pervasive UQ techniques in the literature today, can be used to accurately quantify uncertainty (in the aleatoric realm), identify sensitivities in the SoS and provide a mechanism to reduce risk.

5.3   Importance of UQ

Because of the nature of all SoS's, as outlined in the definition of an SoS, the role of uncertainty quantification is a key consideration on the critical path to mission success and core to risk management. In order to provide adequate confidence in the SoS, both during development and upon fielding, uncertainty and its sources must be sufficiently quantified, explored and mitigated. The ability to do this relies on the ability to identify those factors to which the SoS is sensitive and then explore the bounds and impacts of the sensitivity as well as driving out and exploring emergent behaviors.

While the technique described in this paper takes a significant step towards identifying and investigating uncertainty, the iterative approach with other tools at various levels of abstraction is necessary to adequately quantify the uncertainty in the system. Additionally, the ability to quantify uncertainty in the early portions of the SoS lifecycle provides a foundation on which to form new Verification, Validation, Accreditation and Certification (VVA&C) techniques that leverage the inherent confidence in the SoS rather than the single valued, perfect solution that is implemented currently.

5.4   Implications for SoS Architecting

By following the approach to Conceptual Modeling described herein, the SoS engineers can begin to develop effective, employable architectures that focus more on the necessary capabilities of the SoS

32

vice the capabilities of the constituent systems. Because the integration and interoperability issues impact not only the physical architecture but the employment of the SoS as well, this technique provides the ability to explore those I&I issues and their impacts on *both* the physical architecture and employment strategies of the SoS. The long term implications of building and exercising a conceptual model in this manner are significant changes in the way SoS's are thought of, developed, deployed and employed. Ultimately, exploitation of the event space may result in employment modifications and even changes to the physical architectures at both the component and SoS level.

The goal of this conceptual modeling approach is to craft a SoS which is flexible enough to accommodate the dynamic nature of the Mission sets. The SoS, in order to achieve Mission objectives, may shift between levels of the ACC as the SoS become saturated with stimuli. These shifts may be continuous or discontinuous and functionality within levels of the ACC hierarchy. Because of this, the execution of the OODA paradigm tends to morph between autonomy, collaboration and cooperation blending aspects of each tenet to prosecute the stimuli at hand and achieve Mission-level objectives.

5.5   The Abstraction Paradigm

The abstraction paradigm that we have developed for conceptual modeling is the key to MLSoSE. Recall Figure 11, there are three major tenets to abstraction that *must* be addressed in concert: Entity Abstraction, Functionality Abstraction, and I&I Abstraction. In addition, these tenets should be considered at both the local and enterprise levels in order to dynamically rebalance across the ACC paradigm based on the environmental conditions of the MLSoS. If only one of the tenets is addressed, then the Conceptual Model will be incomplete. Traditionally two tenets (Entity and Functionality) of abstraction are addressed in most Conceptual Models and the resultant products are: 1) marginally credible with no way to assess validity, 2) difficult to interrogate as the intent of the Conceptual Model has been lost, and 3) unable to present information that can be used to address any I&I questions necessary to achieve a semantically composed SoS. In order to create a usable, effective Conceptual Model all three tenets of abstraction must be applied consistently; only then will a common level of abstraction be achieved across the SoS that addresses the Intended Use(s) of the SoS.

5.6   Simulation Tools and Readiness Levels as Future Work

Currently we are in the process of finalizing the second version of software which develops, exercises and analyzes the conceptual model. This work is being applied to many different domains to answer questions regarding development, testing, assessment, operations planning and requirements derivation. Additionally, in the near future we will explore the ramifications of this technique on VVA&C and intend to develop a methodology for VVA&C based on the techniques described herein and their relationship to the Readiness Levels proposed by Sauser, et. al. at the Steven's Institute of Technology.

6   Works Cited

Banks, H., Kenz, Z., & Thompson, W. (2012). *A Review of Selected Techniques in Inverse Problem Nonparametric Probability Distribution Estimation.* Raleigh: N.C. State University.

Bondy, J., & Murty, U. (1976). *Graph Theory with Applications.* New York: North-Holland.

Boyd, C. J. (1976). *Destruction and Creation.*

Boyd, C. J. (1984). *A Discourse on Winning and Losing.*

Boyd, C. J. (1986). *Patterns of Conflict.*

Chibo, M., Hua, L., Ruili, W., & Shudao, Z. (2012). *Quantification of Uncertainties in Detonation Simulations.* Las Vegas: ASME Verification and Validation Symposium.

CJCS. (2010). *CJCSI 6212.01E.* Washington, D.C.: DoD.

Congress, U. (2009). *Weapon Systems Acquisition Reform Act of 2009.* Washington, D.C.: Congress.

Director of Systems and Software Engineering, D. A. (2008). *Systems Engineering Guide for Systems of Systems.* Washington, D.C.: DoD.

Director of Systems Engineering, O. o. (2010). *Systems Engineering Guide for Systems of Systems.* Defense Research and Engineering.

DoD. (2012). *Manual for the Operation of the Joint Capabilities Integration and Development System, JCIDS.* Washington, D.C.: DoD.

DoDAF. (2012). *DoDAF Version 2.0.*

GAO. (2008). *Defense Acquisitions: Significant Challenges Ahead in Developing and Demonstrating Future Combat System's Network and Software.* Washington, D.C.: GAO.

GAO. (2011). *GAO-11-502 Missed Trade-off Opportunities.* Washington, D.C.: GAO.

GAO. (2012). *GAO-13-103 Weapons Acquisition Reform.* Washington, D.C.: GAO.

GAO. (2012). *Next Generation Enterprise Network: Navy Implementing Revised Approach, but Improvement Needed in Mitigating Risks.* Washington, D.C.: GAO.

GAO. (2013). *Army Networks: Size and Scope of Modernization Investment Merit Increased Oversight.* Washington, D.C. : GAO.

GAO. (2013). *High Risk Series: An Update.* Washington, D.C.: GAO.

Garrett, R., Baron, N., Moreland, J., & Anderson, S. (2012). Managing the Interstitials, a System of Systems Framework Suited for the Ballistic Missile Defense System. *INCOSE*.

34

Held, J. M. (2008). *Systems of Systems: Principles, Performance and Modelling.* Sydney: The University of Sydney.

Henninger, A., Cutts, D., Loper, M., Lutz, R., Richbourg, R., Saunders, R., et al. (2008). *Live Virtual Constructive Architecture Roadmap (LVCAR) Final Report.* IDA.

ISO/IEC. (1994). ISO/IEC Standard 7498-1:1994.

Jamshidi, M. (2009). *System of Systems Engineering, Innovations for the 21st Century.* Hoboken: Wiley.

Jiminez, L., & Landgrebe, D. (1997). Supervised Classification in High Dimensional Space - Geometrical, Statistical and Asymptotical Properties of Multivariate Data. *IEEE*.

JTEM. (2009). *Analysis Handbook for Testing in a Joint Environment.* Washington, D.D.: JTEM.

Maier, M. W. (1998). *Architecting Principles for Systems of Systems.* INFOED.

Mayo, D. G., & Cox, D. R. (2006). Frequentist Statistics as a Theory of Inductive Inference. *2nd Lehmann Symposium - Optimality*.

McCabe, J. T. (1968). *Estimating Conditional Probability and Persistence.* Washington, D.C.: Air Weather Service, USAF.

Morris, E., Levine, L., Meyers, C., Place, P., & Plakosh, D. (2004). *System of Systems Interoperability (SOSI): Final Report.* Carnegie Mellon University.

National Academy of Sciences. (2012). *Assessing the Reliability of Complex Models: Mathematical and Statistical Foundations of Verification, Validation, and Uncertainty Quantification.* Washington, D.C.: National Academies Press.

Office of the ASN (RDA) Chief Systems Engineer. (2006). *Naval "Systems of Systems" Systems Engineering Guidebook, Vol. II.* Navy.

Pilch, M., Trucano, T., & Helton, J. (2006). *Ideas Underlying Quantification of Margins and Uncertainties (QMU): A White Paper.* Albequerque: Sandia.

Roy, C., & Balch, M. (2011). A Holistic Approach to Uncertainty Quantification in Modeling and Simulation. *USA/South America Symposium on Stochastic Modeling and Uncertainty Quantification in Complex Systems.* Rio de Janeiro Brazil: Virginia Tech.

Sauser, B., Ramirez-Marquez, J., Verma, D., & Gove, R. (2004). *From TRL to SRL: The Concept of System Readiness Levels.* Hoboken.

Selberg, S., & Austin, M. (2008). Toward an Evolutionary System of Systems Architecture. *Proceedings of the Eighteenth Annual International Symposium of The International Council on Systems Engineering (INCOSE).* The Netherlands: INCOSE.

Simic, S. (2009). An Extension of Stolarsky Means to the Multivariable Case. *International Journal of Mathematics and Mathematical Sciences*.

START. (2010). *Understanding Series and Parallel Systems Reliability.* Rome: RAC.

Stolarsky, K. (1975). Generalizations of the Logarithmic Mean. *Mathematics Magazine*, 87-92.

Surowiecki, J. (2005). *The Wisdom of Crowds.* New York: Doubleday.

Weter, D. (2012). *Joint Staff J7 Modeling and Simulation Program Update.*

Wolkowicz, H. (2006). *Optimization: Theory, Algorithms, Applications.* Waterloo: University of Waterloo.

7 Author Biographies

Mr. Raymond Deiotte earned a B.S. in Physics from the University of Colorado in 2004. He began his career at the Colorado Center for Astrodynamics Research where he provided software and algorithmic support for remote sensing research for multiple government and private agencies. He spent 3 years at the Boeing Company working Air Force and Missile Defense related research and development, specifically, System of Systems architecture and design for distributed training, test and operations. Mr. Deiotte moved to the ISSAC Corp in 2008, where he has been the lead engineer and scientist and holds the title of Director of Innovation. While supporting the Boeing Company with ISSAC Corp, Mr. Deiotte's work led to ISSAC Corp winning a Gold Performance Excellence Award. With the ISSAC Corp, supporting the Missile Defense Agency, Mr. Deiotte has provided expertise and novel solutions for the M&S Engineering Directorate ranging from artificial intelligence engines to support requirements elicitation to methods for Verification and Validation of complex Systems of Systems. Mr. Deiotte has extensive experience and expertise in Model Based Engineering of component systems, Systems of Systems and M&S.

Mr. Robert K. Garrett, Jr. earned a B.S. in Materials Engineering from Purdue University in 1981 and an M.S. in Materials Engineering from Purdue University in 1983. He worked with the Naval Surface Warfare Center for 27 years starting in 1983 at the White Oak Laboratory, and ending up at the Dahlgren Laboratory in 2000 working primarily in research and development. His expertise is in systems engineering, integration of diverse technologies into weapon systems, and the application of materials science and continuum mechanics to weapon systems development. In 2010 Mr. Garrett joined the Missile Defense Agency supporting the Modeling and Simulation Directorate. He has focused his attention on adapting, developing, and exploiting modeling, simulation, and analysis tools and processes for integrating a complex, adaptive System of Systems. Mr. Garrett was recognized by the International Council on Systems Engineering (INCOSE) for his published work, "Managing the Interstitials: A System of Systems Framework Suited for the Ballistic Missile Defense System". The work was selected by a committee of Associate Editors as the recipient of the 2011 Best Paper Award.
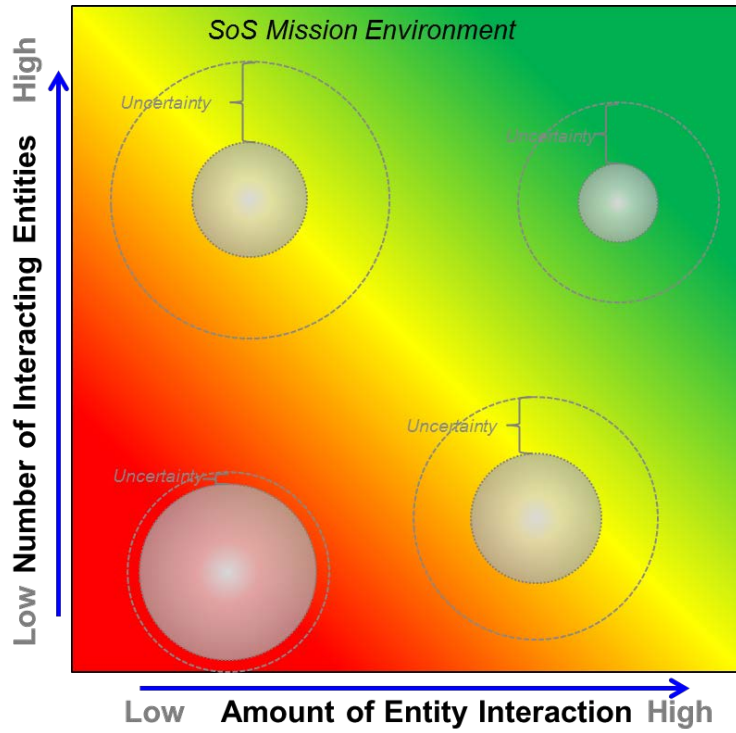
Figure 1: Describing Complexity
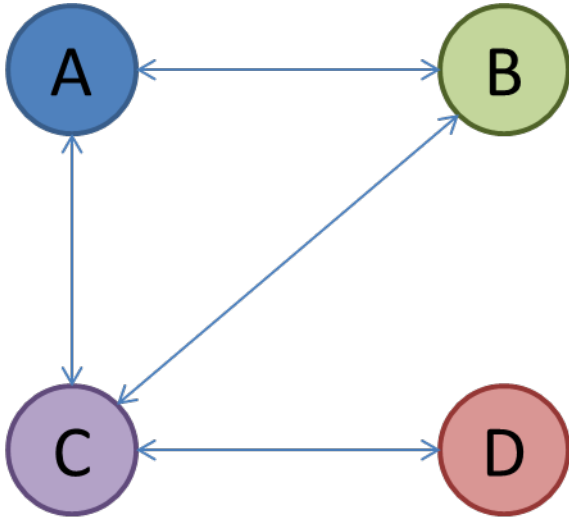
Figure 2: Physical Space Representation
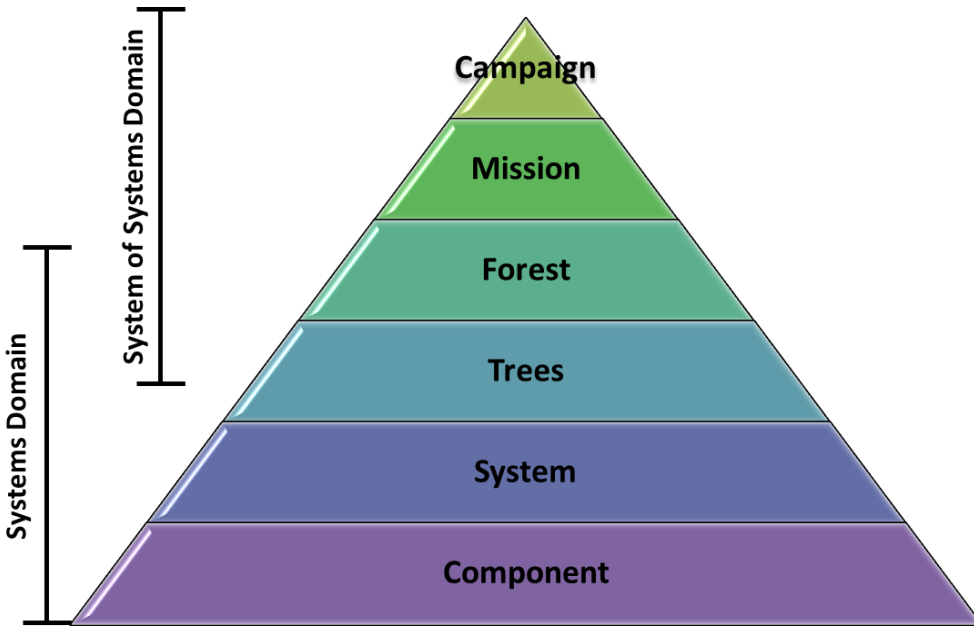
Figure 3: Mission Hierarchy

Interoperability (Semantic Composition)
• Mission Environment
• Meaningful Interactions
• Cooperation
• Many-to-Many
• Multiple Mission Threads

Integration (Syntactic Composition)
• Tends to be focused on single mission thread
• Tends toward system-to-system interactions (pair wise)
• Builds on constituent systems
• Connection-based
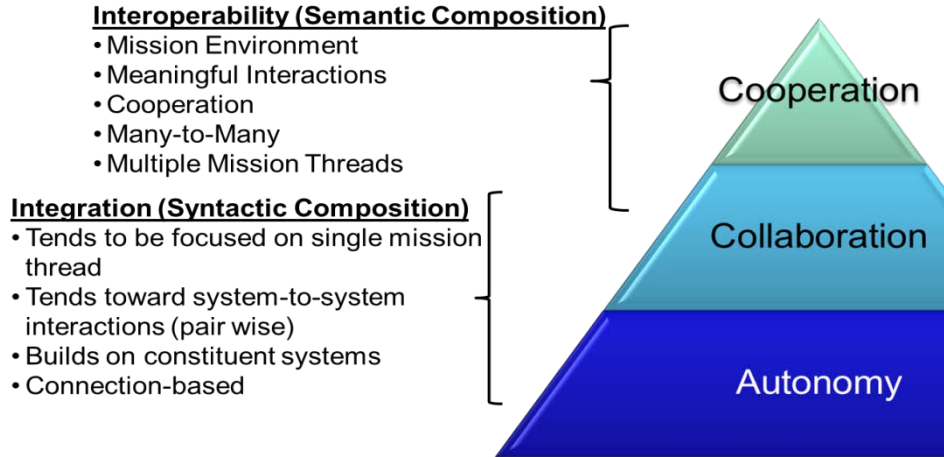
Figure 4: I&I Levels and Sub-classes
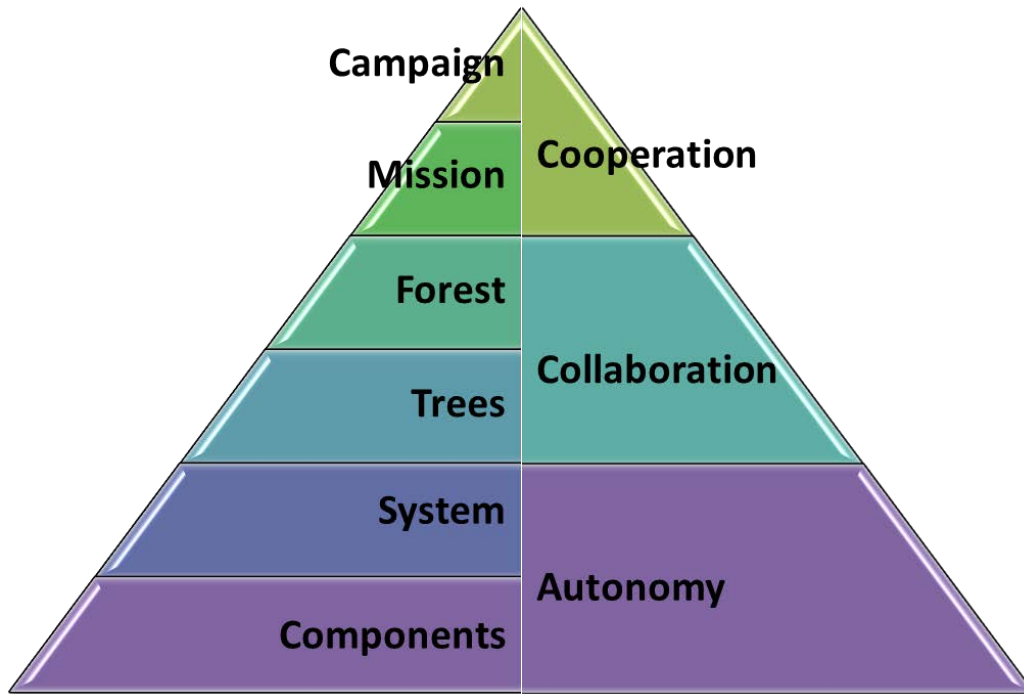
Figure 5: Crosswalking I&I with the Hierarchy
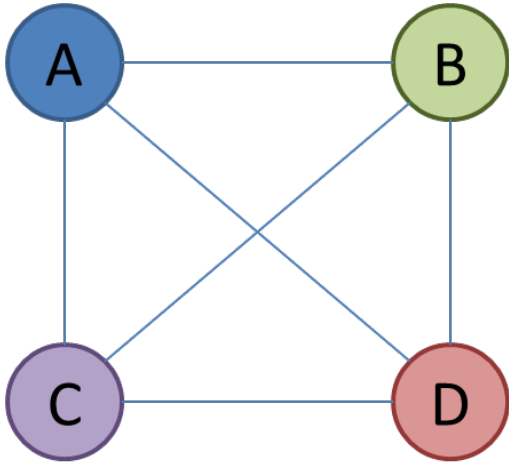
Figure 6: Simple Graph
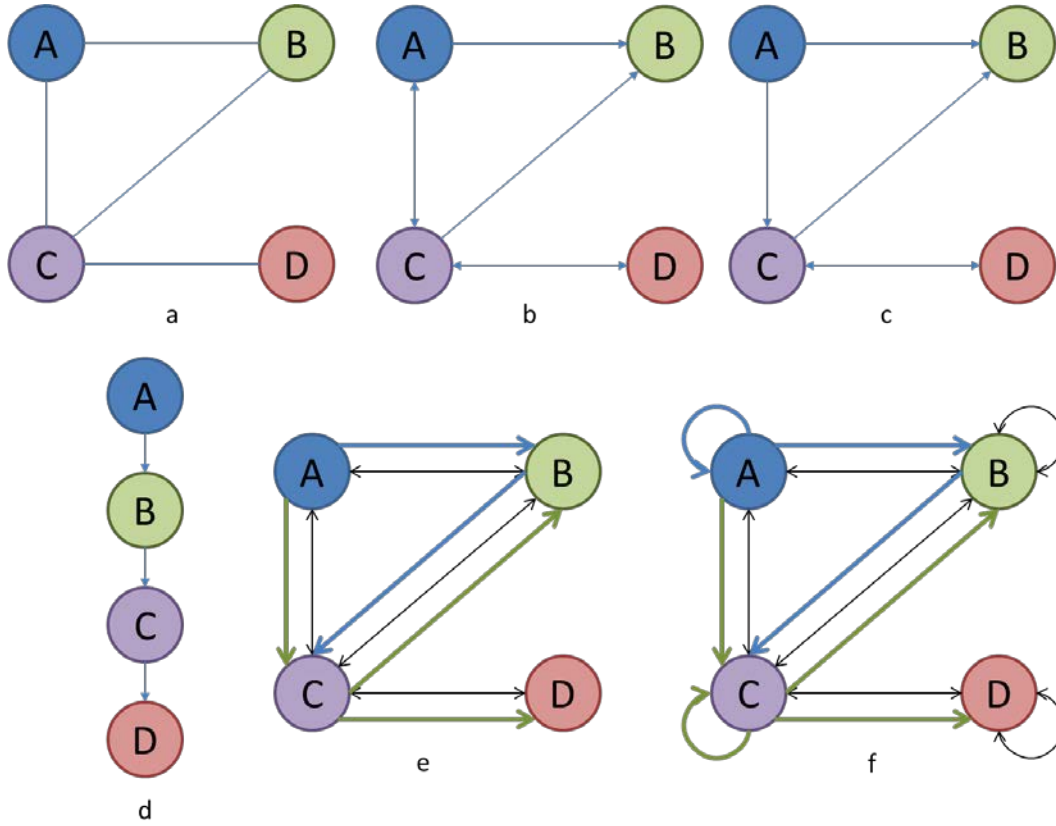
Figure 7: Simple, Directed, and Multi-Graphs
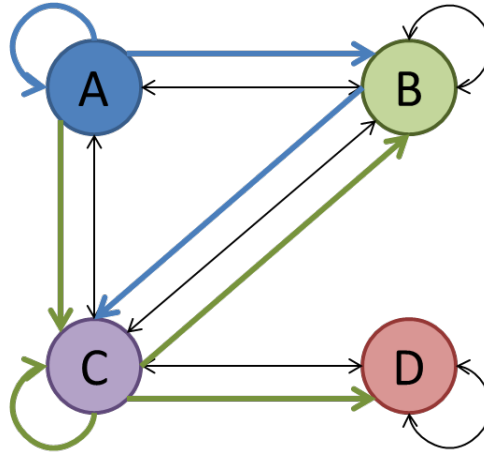
	A	B	C	D
A	1	2	2	0
B	1	2	2	0
C	1	2	1	2
D	0	0	1	2

	e1	e2	e3	e4	e5	e6	e7	e8	e9	e10	e11	e12	e13
A	2	-1	-1	1	1	0	0	0	0	0	0	0	0
B	0	1	0	0	-1	2	-1	0	1	1	0	0	0
C	0	0	1	-1	0	0	1	2	-1	-1	-1	1	0
D	0	0	0	0	0	0	0	0	0	0	1	-1	2

Figure 8: Notional Adjacency and Incidence Matrices
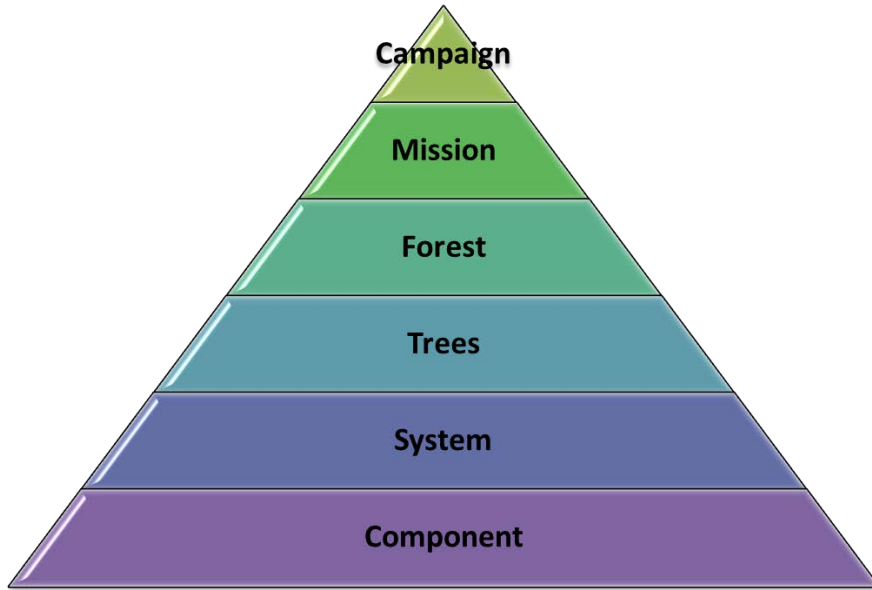
Figure 9: SoS Pyramid
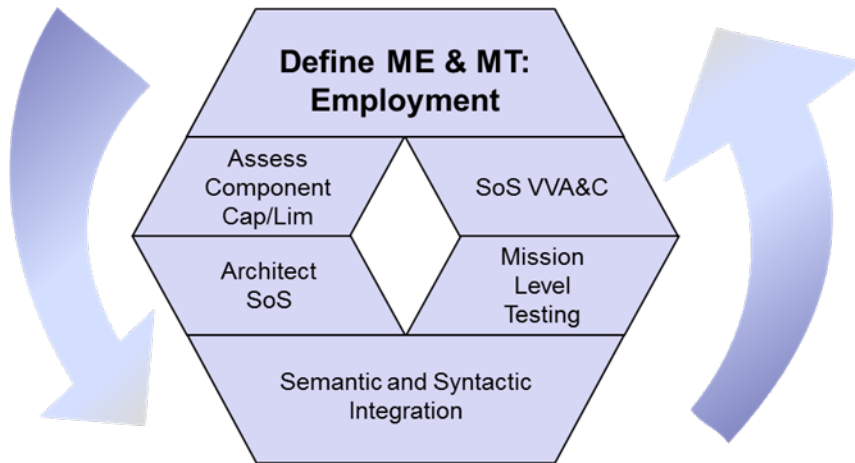
Figure 10: Mission-Level SoS Engineering
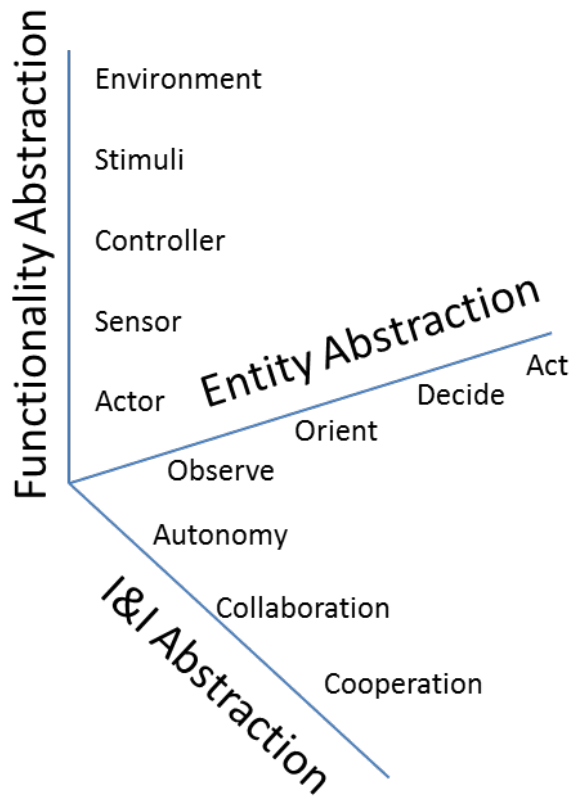
Figure 11: The Three Tenets of Abstraction
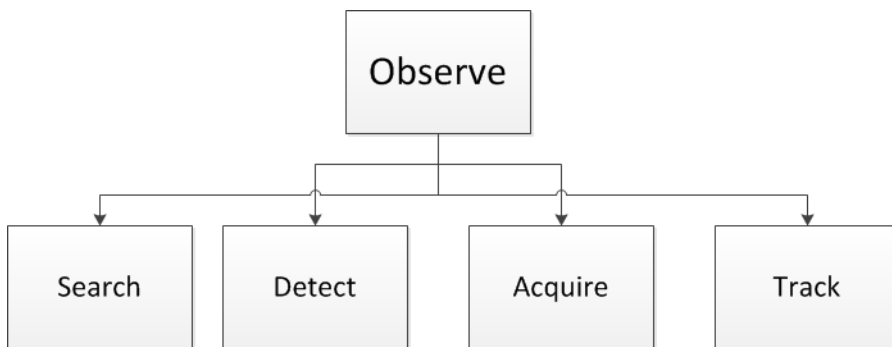
Figure 12: The Observe Functionality of OODA
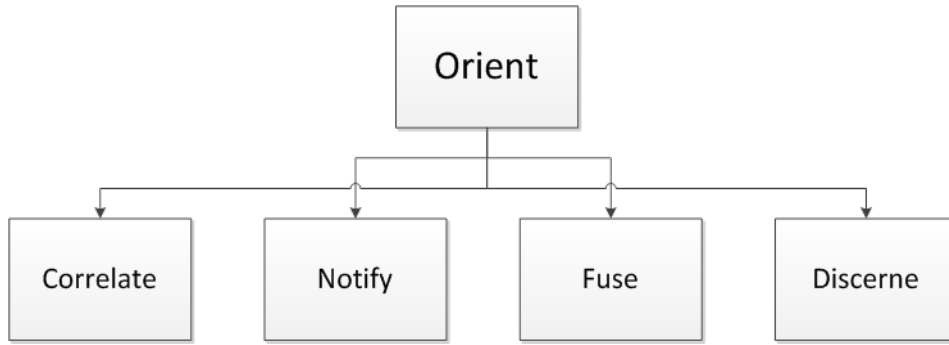
Figure 13: Orient Function of OODA
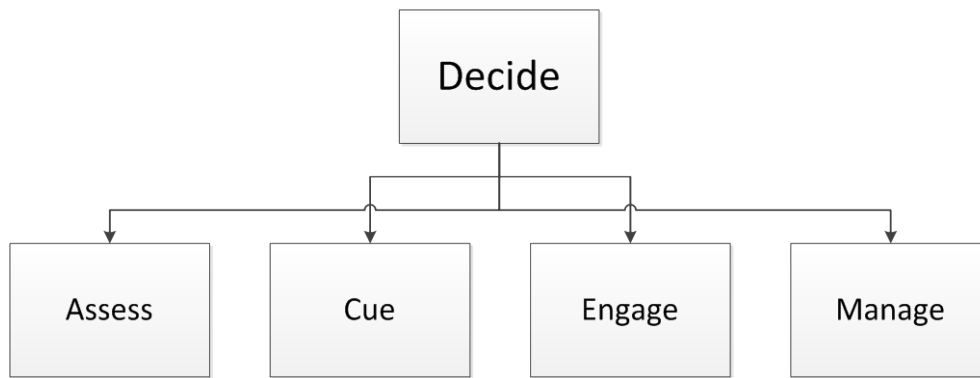
Figure 14: The Decide Function of OODA
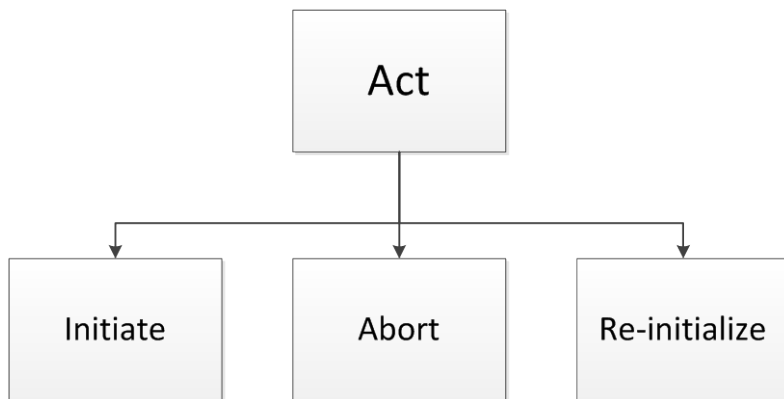
Figure 15: The Act Function of OODA
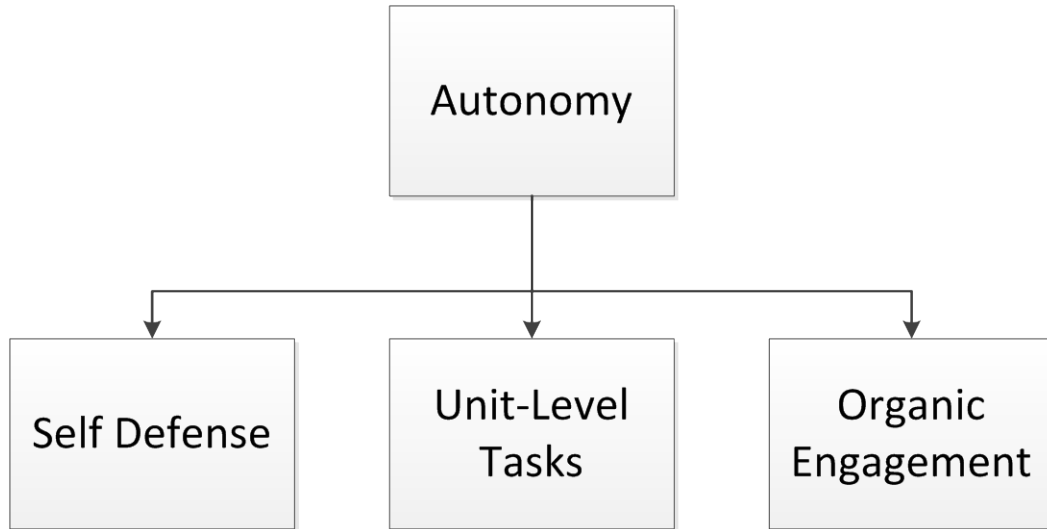
Figure 16: The Autonomy Taxonomy of ACC
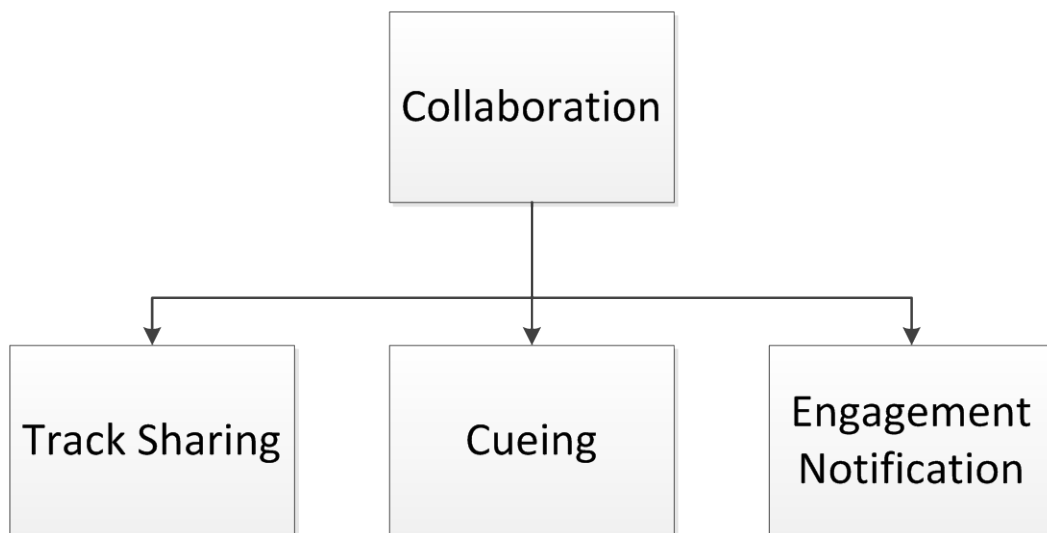
Figure 17: The Collaboration Taxonomy of ACC
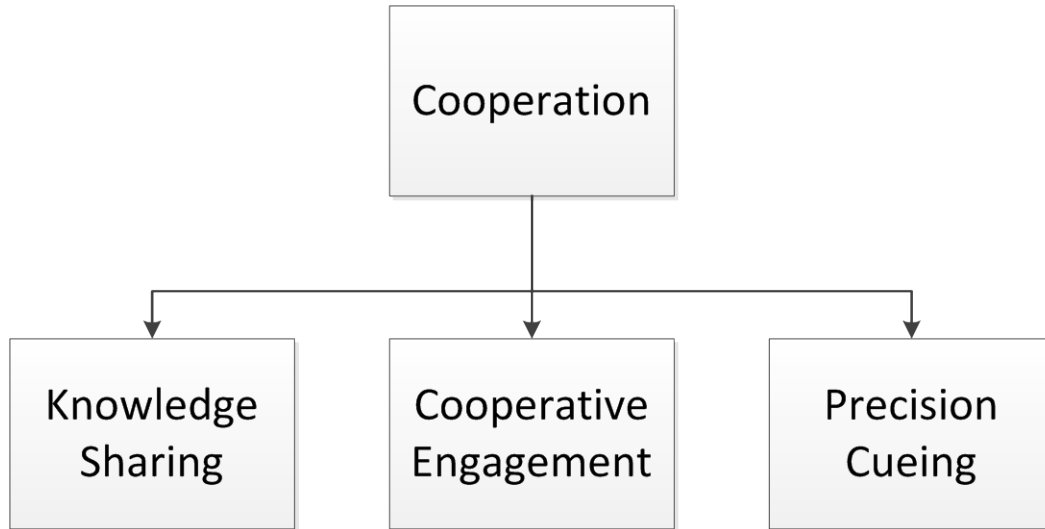
Figure 18: The Cooperation Taxonomy of ACC

$$P(V_{16}) = P_S$$
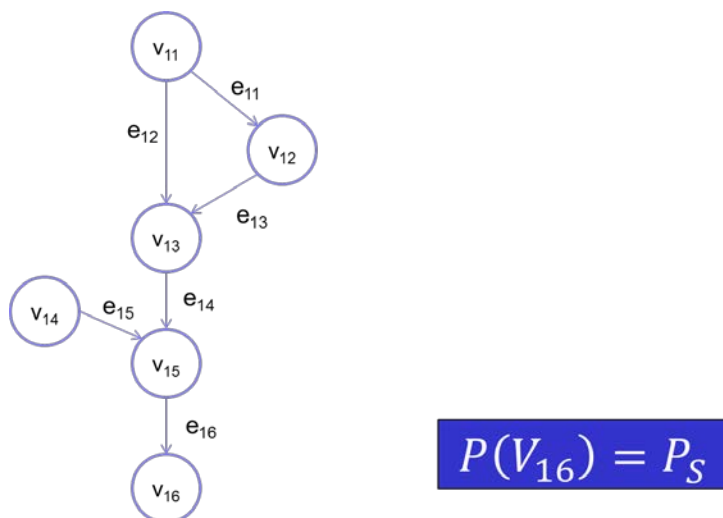
Figure 19: Metrics are a Natural Output of MBSoSE
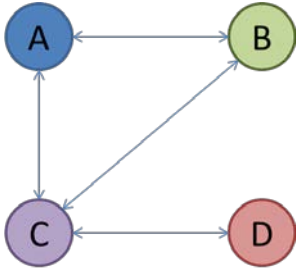
Figure 20: Simple Network
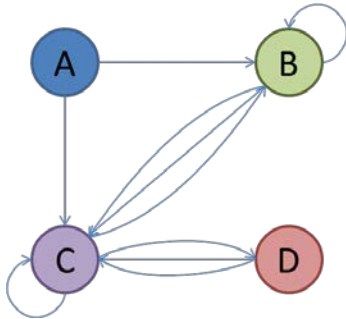
Figure 21: Expanded Multi-graph
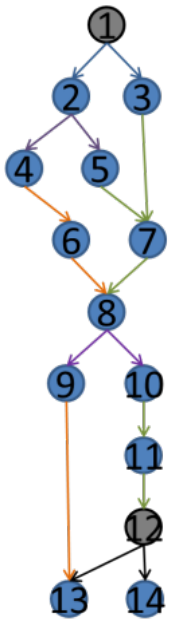
**Figure 22: Directed
Acyclical Graph**
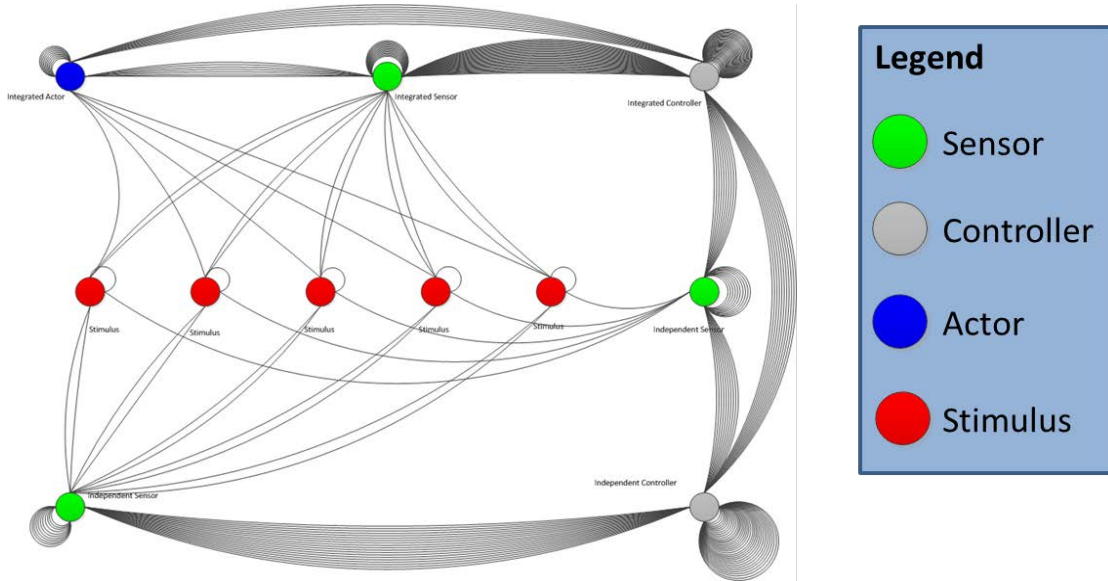
Figure 23: Sample Physical Space with Integrated and Non-Integrated Systems
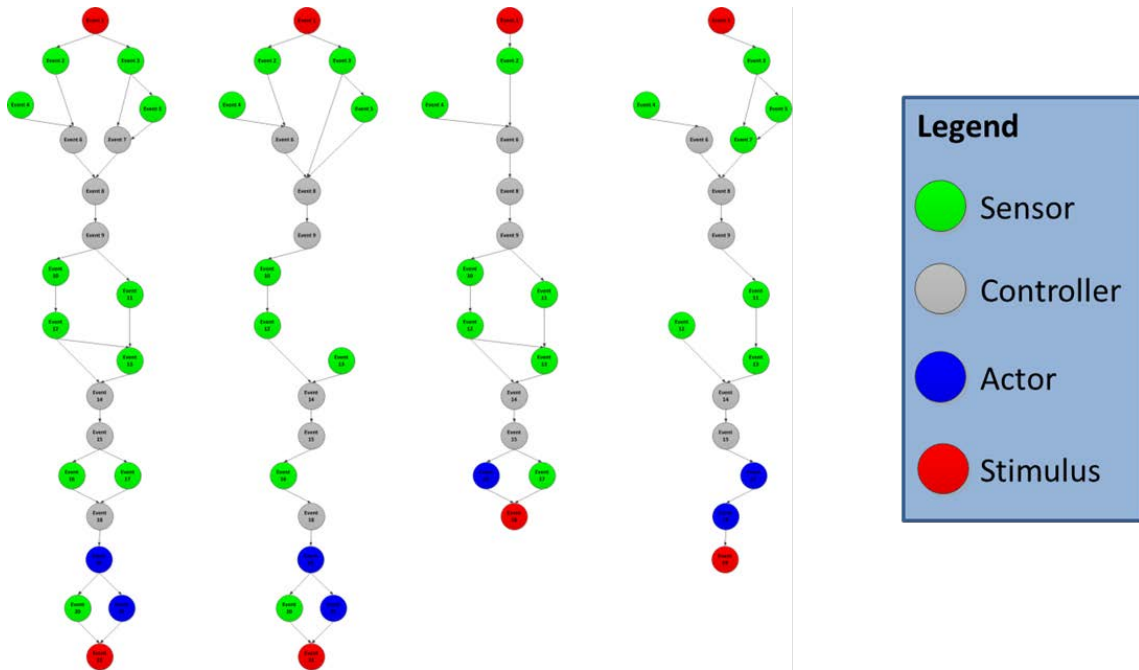
Figure 24: Sample Resultant Event Spaces
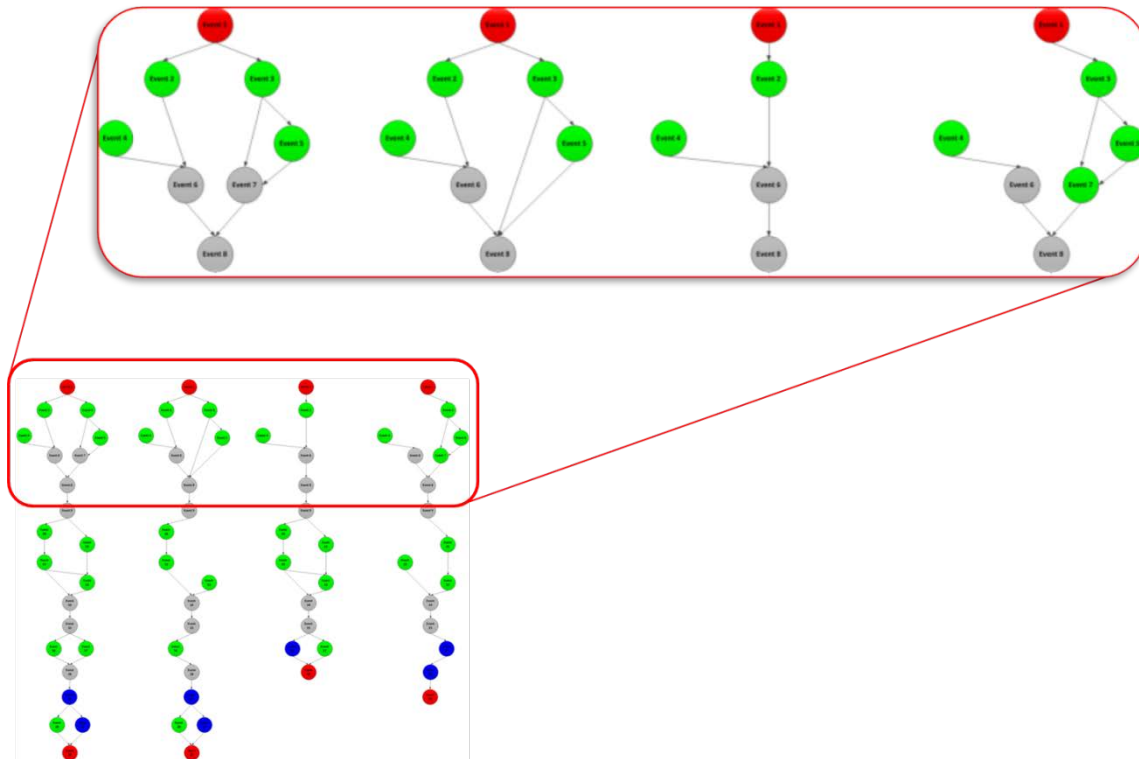
Figure 25: Sample Resultant Event Spaces
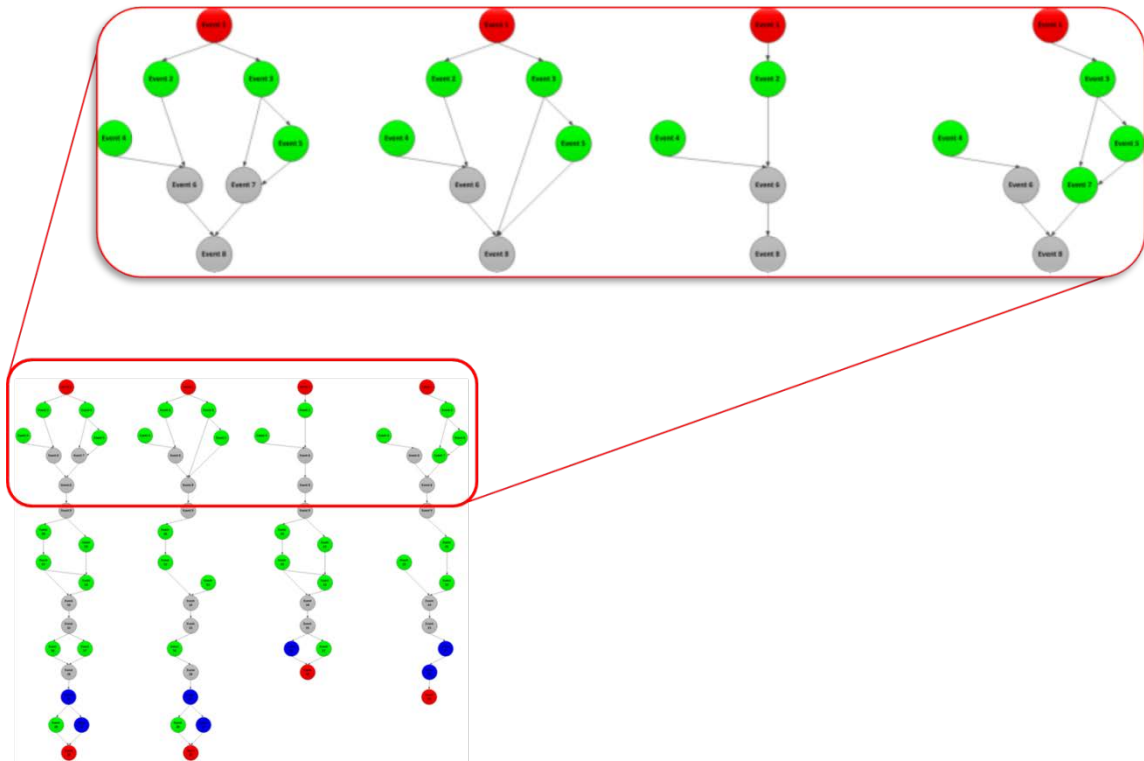
Figure 26: Stimulus and Employment Variations in Event Space
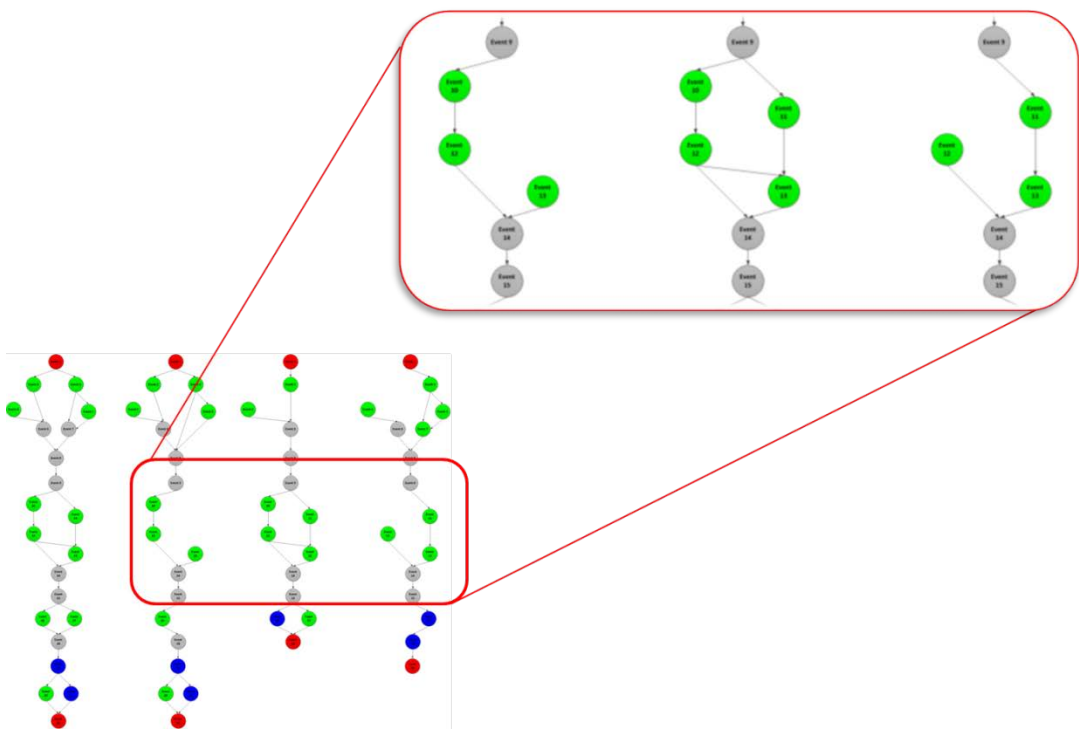
Figure 27: Employment Differences in Event Spaces
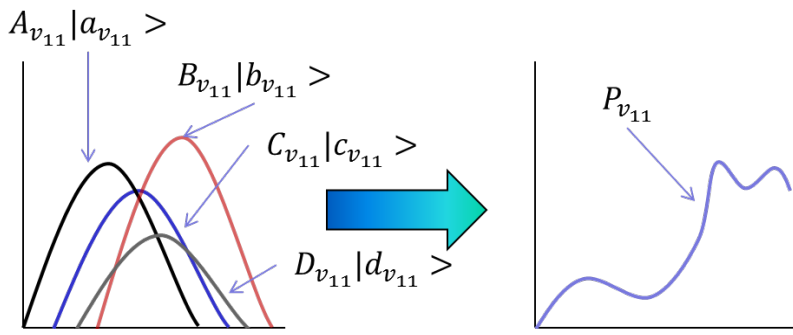
Figure 28: Superposition of PDF's
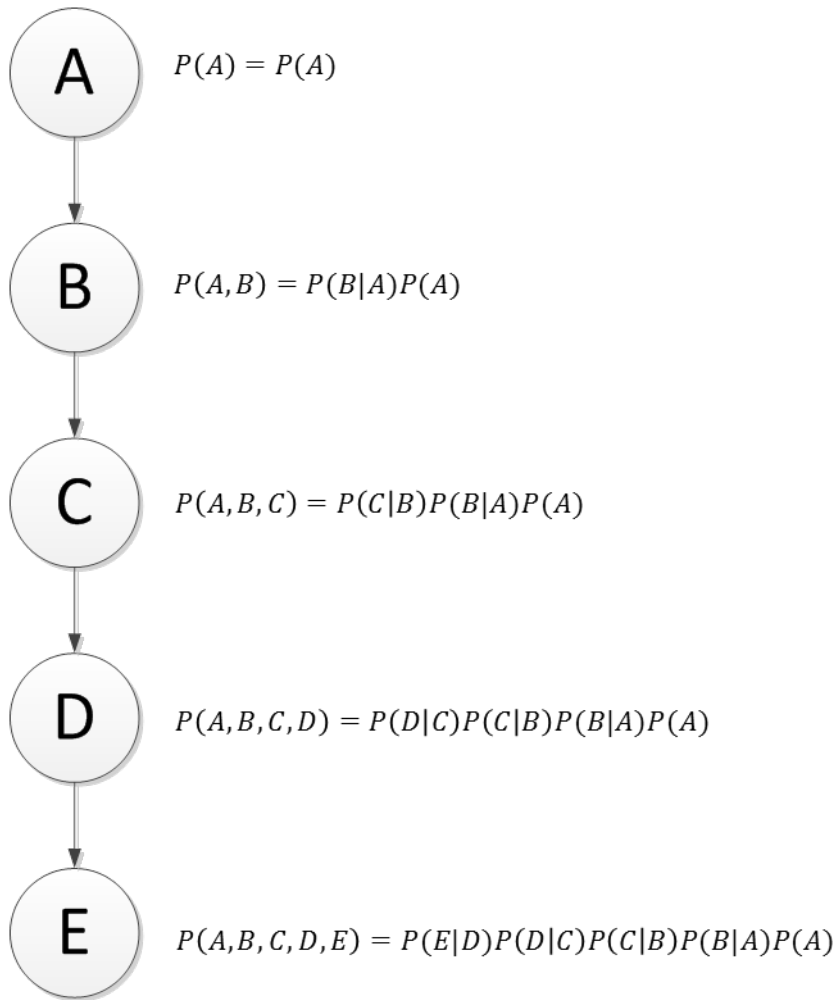
Figure 29: Serial Bayesian Network
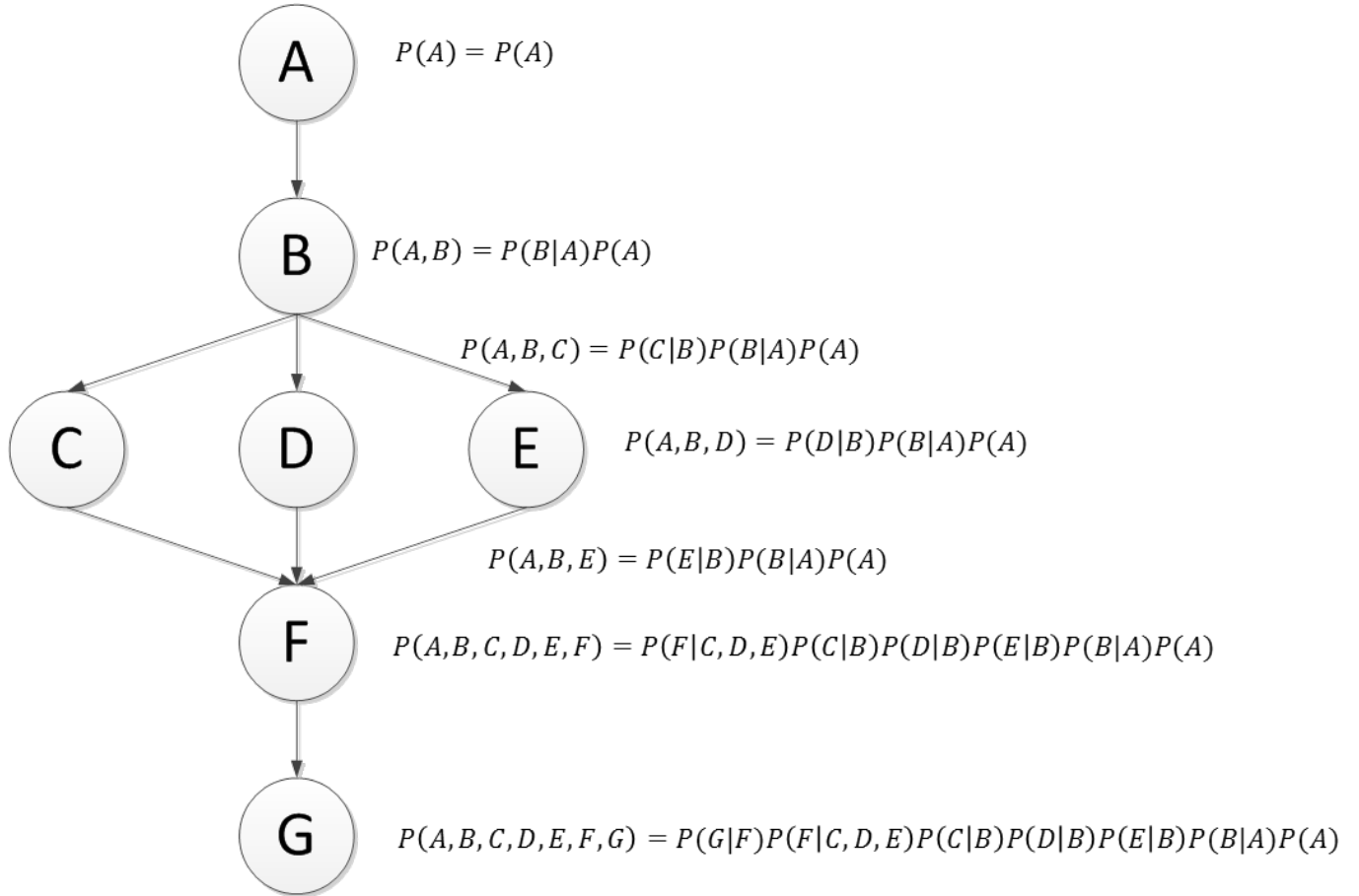
Figure 30: Parallel Bayesian Network Formulation

The following equations accompany the figure:

$$P(A) = P(A)$$

$$P(A,B) = P(B|A)P(A)$$

$$P(A,B,C) = P(C|B)P(B|A)P(A)$$

$$P(A,B,D) = P(D|B)P(B|A)P(A)$$

$$P(A,B,E) = P(E|B)P(B|A)P(A)$$

$$P(A,B,C,D,E,F) = P(F|C,D,E)P(C|B)P(D|B)P(E|B)P(B|A)P(A)$$

$$P(A,B,C,D,E,F,G) = P(G|F)P(F|C,D,E)P(C|B)P(D|B)P(E|B)P(B|A)P(A)$$
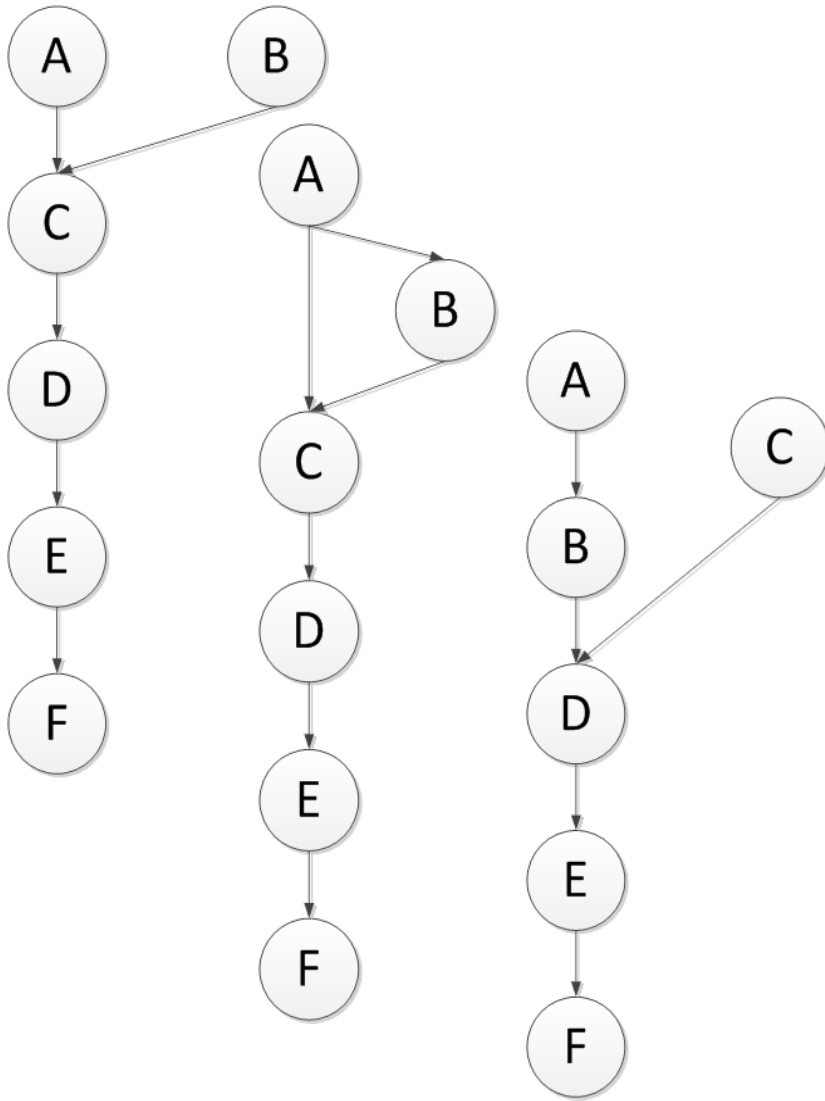
Figure 31: Bayesian Networks with Indirectly Related Parents
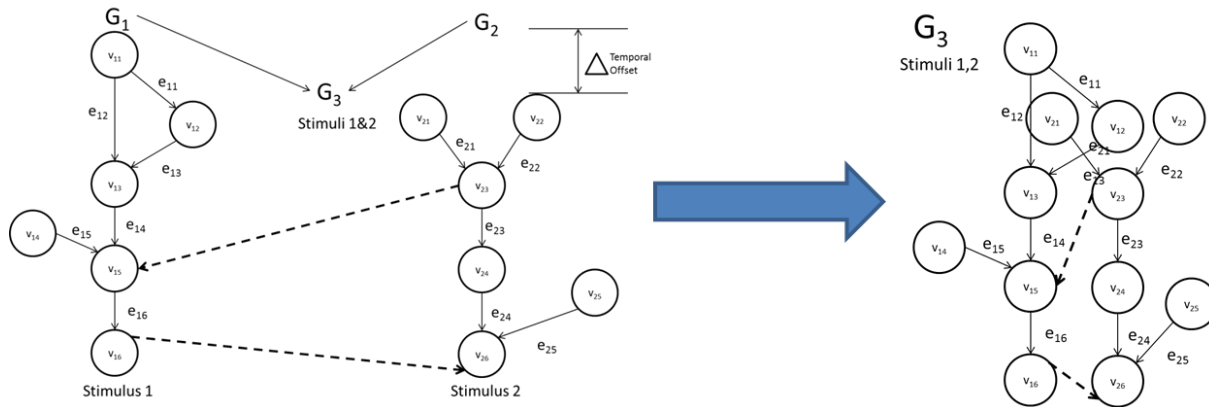
Figure 32: Convolution of Event Spaces
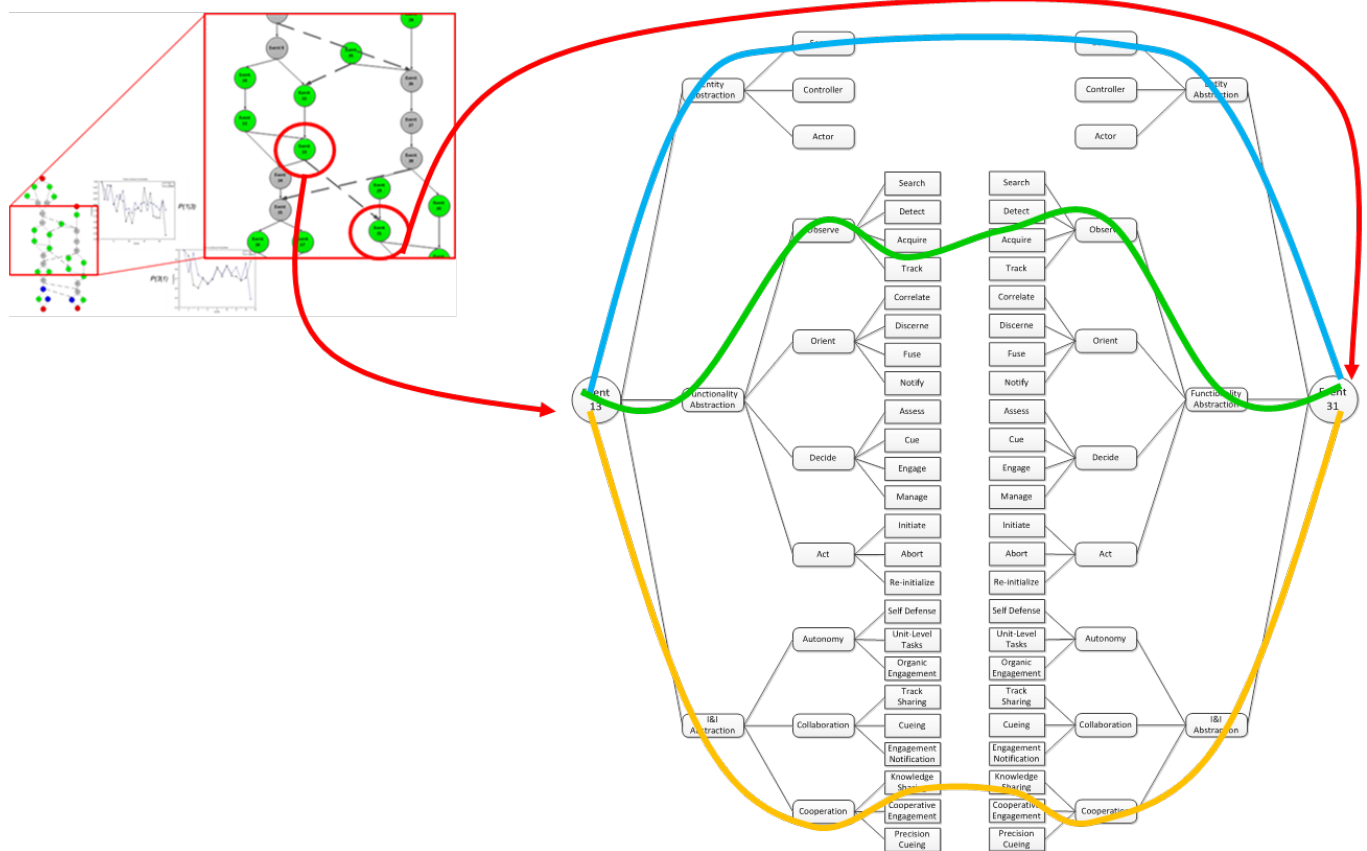
Figure 33: Convolution Through Abstraction
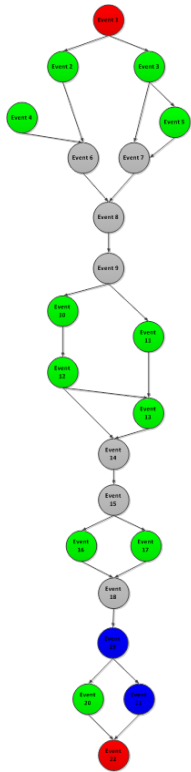
57

Node	Draw	Correlation Factor (ψ)	Independent Form	Conditional Form
1	1.00000	0.00000	1.00000	1.00000
2	0.75700	0.70638	0.75700	0.92865
3	0.57280	0.67181	0.43361	0.83999
4	0.96372	0.00000	0.41788	0.96372
5	0.62284	0.58904	0.26027	0.80297
6	0.50822	0.72572	0.13228	0.87580
7	0.91639	0.92999	0.12122	0.84723
8	0.63421	0.81841	0.07688	0.91805
9	0.53812	0.57355	0.04137	0.75603
10	0.77548	0.63569	0.03208	0.76312
11	0.62061	0.76697	0.01991	0.72447
12	0.86685	0.52984	0.01726	0.81189
13	0.90448	0.65655	0.01561	0.77039
14	0.69744	0.94999	0.01089	0.80799
15	0.95267	0.92073	0.01037	0.81946
16	0.63584	0.63047	0.00659	0.75161
17	0.81032	0.66385	0.00534	0.81639
18	0.63263	0.53189	0.00338	0.80377
19	0.57759	0.90407	0.00195	0.78207
20	0.66622	0.75007	0.00130	0.75312
21	0.90561	0.57645	0.00118	0.83440
22	0.52967	0.51823	0.00062	0.75222
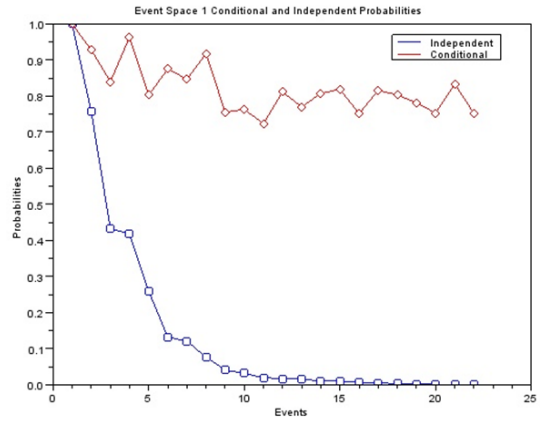
Figure 34: Notional Event Space 1 with Computations
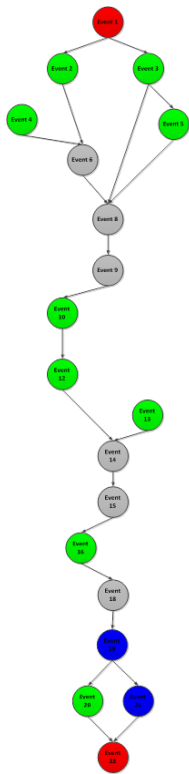
| Node	Draw	Correlation Factor (ψ)	Independent Form	Conditional Form
1	1.00000	0.00000	1.00000	1.00000
2	0.75700	0.81178	0.75700	0.954263705
3	0.57280	0.69410	0.43361	0.869318214
4	0.96372	0.13138	0.41788	0.96372
5	0.62284	0.89976	0.26027	0.793918304
6	0.50822	0.89197	0.13228	0.930680859
8	0.63421	0.17027	0.08389	0.671975148
9	0.53812	0.49374	0.04514	0.604208678
10	0.77548	0.59395	0.03501	0.673755541
12	0.86685	0.49639	0.03035	0.770999159
13	0.90448	0.27493	0.02745	0.90448
14	0.69744	0.81613	0.01914	0.776507694
15	0.95267	0.86039	0.01824	0.801101666
16	0.63584	0.32579	0.01160	0.689683631
18	0.63263	0.58568	0.00734	0.66604411
19	0.57759	0.75683	0.00424	0.644534025
20	0.66622	0.78808	0.00282	0.649129271
21	0.90561	0.84789	0.00256	0.68424514
22	0.52967	0.61585	0.00135	0.803547235
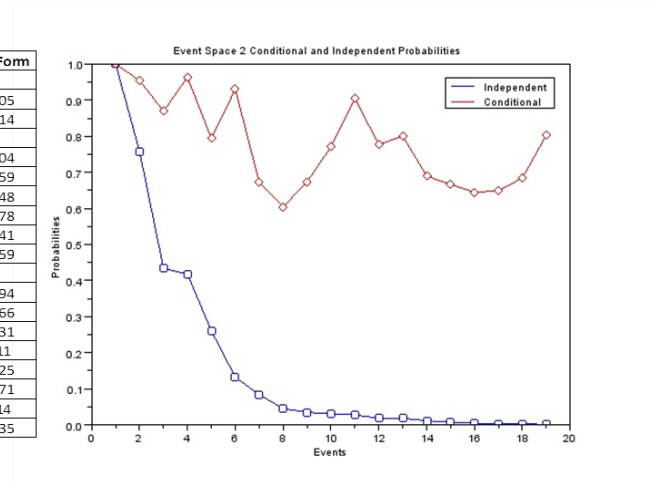
Figure 35: Notional Event Space 2 with Computations
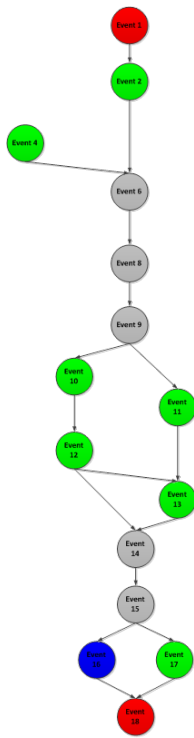
Node	Draw	Correlation Factor (ψ)	Independent Form	Conditional Form
1	1.00000	0.00000	1.00000	1.00000
2	0.75700	0.05298	0.75700	0.76988
4	0.96372	0.28893	0.72954	0.96372
6	0.50822	0.11957	0.37077	0.74003
8	0.63421	0.55067	0.23515	0.69248
9	0.53812	0.65534	0.12654	0.63928
10	0.77548	0.61659	0.09813	0.69150
11	0.62061	0.27354	0.06090	0.80460
12	0.86685	0.67947	0.05279	0.74771
13	0.90448	0.78679	0.04775	0.82955
14	0.69744	0.74459	0.03330	0.75174
15	0.95267	0.67369	0.03172	0.81730
20	0.66622	0.12461	0.02114	0.68504
21	0.90561	0.20875	0.01914	0.85956
22	0.52967	0.19960	0.01014	0.48530
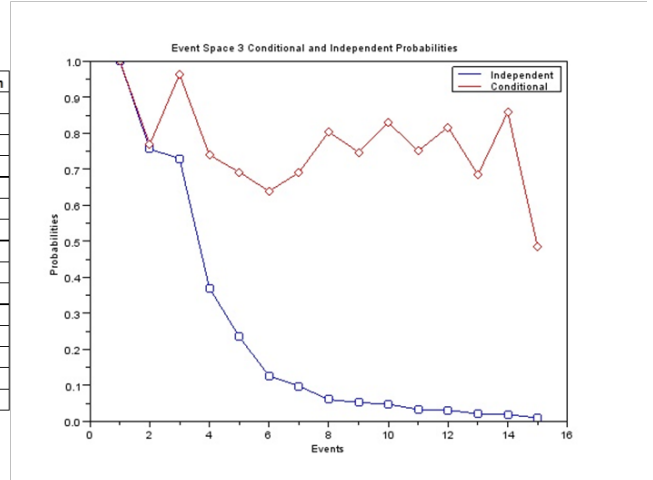
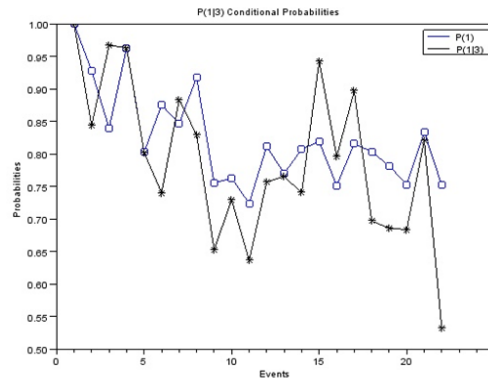**Figure 36: Notional Event Space 3 with Computations**

60
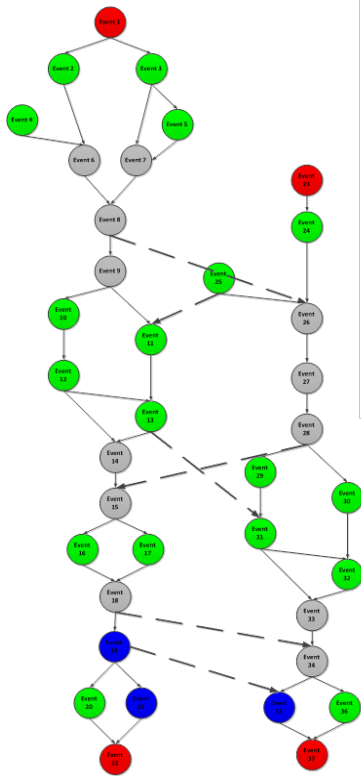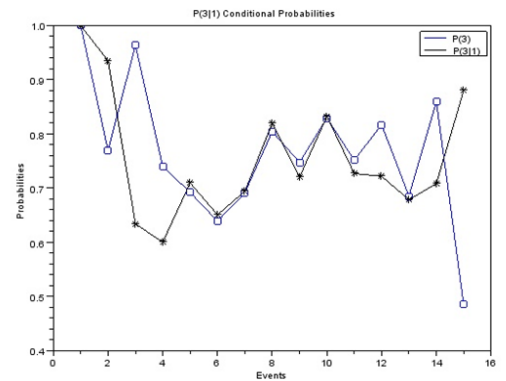
Figure 37: ES 1 and 3 Convolved

$P(1|2,3)$

$P(2|1,3)$

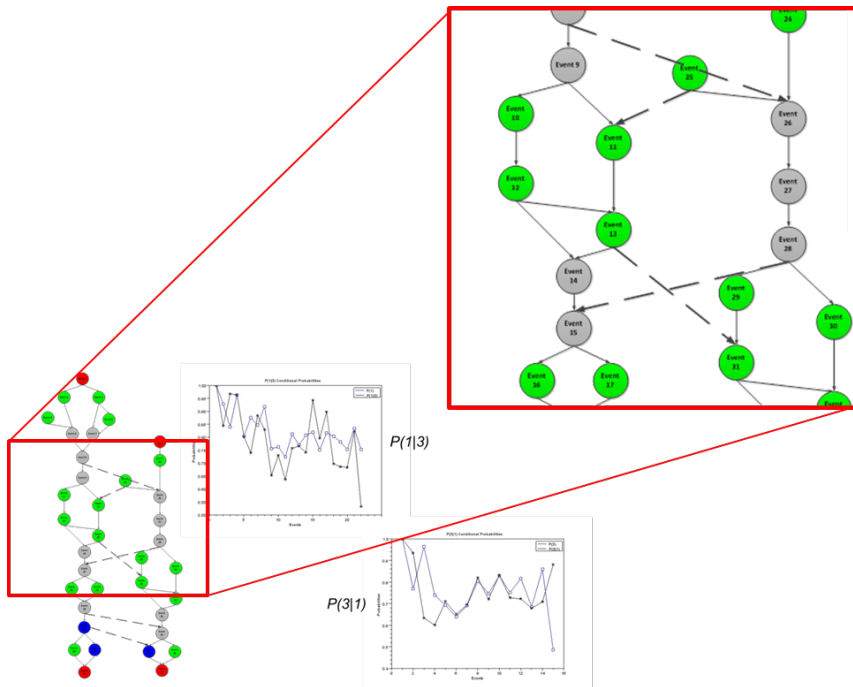$P(3|1,2)$
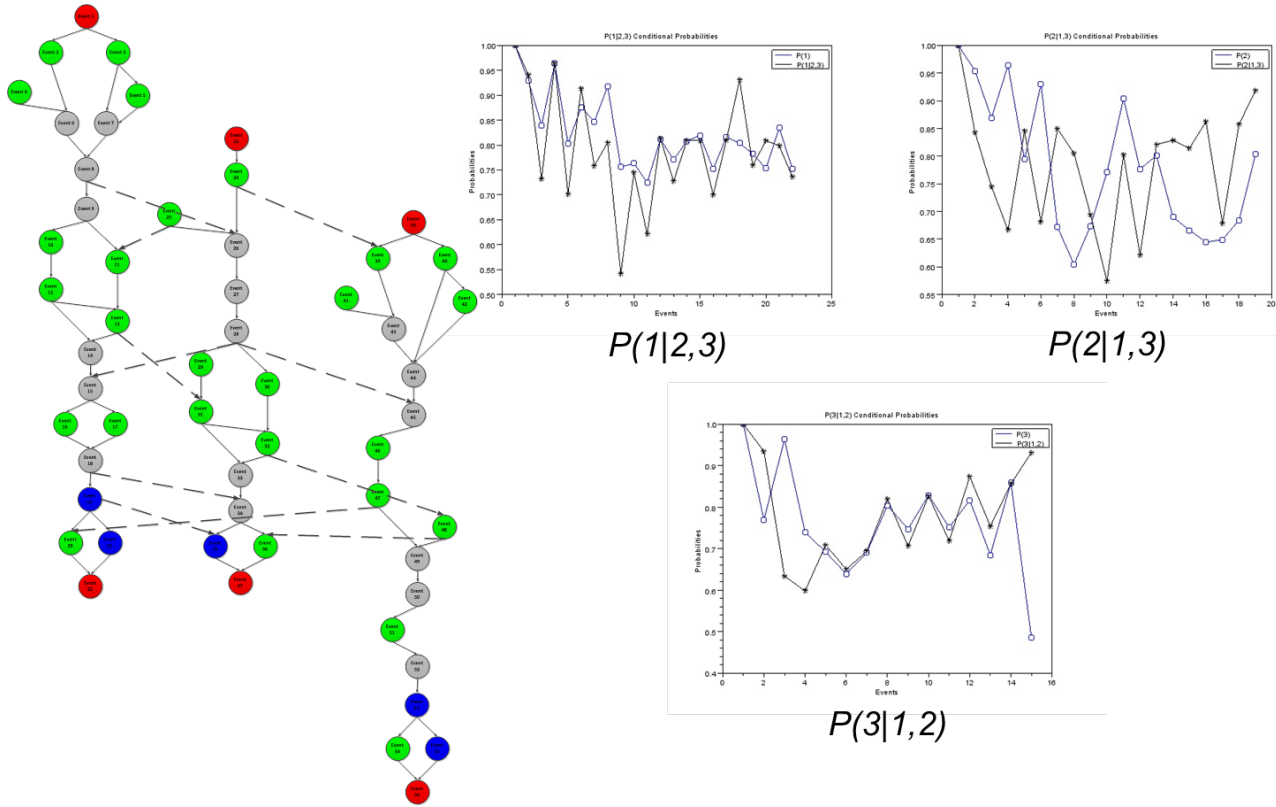
Figure 38: Convolution of ES1, ES2, ES3

$P(1|3)$

$P(3|1)$

Figure 39: Details of Controller and Sensor Convolution

62