A Graph-Based Metamodel for Enabling System of Systems Engineering

Robert K. Garrett, Jr.
Missile Defense Agency
COS MDIOC-730 337 1
730 Irwin Ave
Schriever, CO 80912
719-522-3692
Robert.garrett2@mda.mil

Abstract--Systems of systems have received significant attention in the literature focusing principally on defining the limits of a system of systems, methods of system of system engineering, and the relationships between system of systems engineering and classical systems engineering. There are opportunities in systems of systems literature for proposing engineering execution processes along with enabling capabilities to address the complexity of the system of systems domain and the challenges of managing the dynamic relationships between the self-governing entity systems. This paper is established upon the notion that the performance of systems of systems is driven by the relationships between systems, and thus systems of systems are inherently graph-based. This graph basis enables system of systems engineering to be anchored on the foundation of a graph-based metamodel to manage vast stores of complex artifacts, i.e. big data, and enable model-based simulations. The systems of systems under development, or being modified during employment, have a model basis that should be common to a 'run time' environment of live, virtual and constructive simulations consisting of probabilistic, agent based and discrete event compositions. The metamodel can store and manage the big data associated with the simulation input and output such the run time environment can be quickly composed and executed providing for near real-time virtual testing of the system of systems.

Index Terms--graph-based metamodels, simulation, system of systems

I. Introduction

The domain of System of Systems (SoS) and Systems of Systems Engineering (SoSE) has been receiving attention, much of it from the United States Department of Defense (DoD). Initial attention was placed on differentiating between a collection of systems and a SoS. Subsequent efforts focused on SoS management and the classification of inter-entity system relationships. Much of this SoS work began with the notion of extending classic System Engineering (SE) processes that have been successfully used for engineering complex systems to the SoSE domain. A reoccurring thread through the SoSE literature is a lack of means to quantify and engineer complexity about relationships in a SoS, and the management of the ‘big data’ inherent in that complexity. Complexity naturally arises in a SoS not only due to the number of employed systems in the SoS, but also due to the uncertainty of interdependent relationships between systems as they are employed against a number of different stimuli and missions. Managing the resulting engineering data is a formidable task made more challenging as entity systems usually evolve independently of the SoS and mission goals within an ever changing environment. This paper proposes a graph-based metamodel as a key enabling capability for SoSE to manage relationships, complexity and uncertainty and form a basis for SoS modeling and simulations focused primarily on conceptualization and architecture in the complex SoS domain.
Maier (1998) defined a SoS architecturally using the following attributes:

- Operational Independence of the Entity Systems
- Managerial Independence of the Entity Systems
- Geographic Distribution
- Emergent Behavior
- Evolutionary Development.

These five criteria now provide a stable basis for most SoS criteria when addressing the development, integration and management of distributed systems. While the notion of emergent behavior is not defined as an architectural attribute, the role of emergence is discussed. Maier also realized that the performance of the SoS was dependent upon entity system relationships, i.e., the ability of the entity systems to communicate and collaborate in achieving a common mission goal. To help define the character of these inter-entity relationships, he introduced the concept of SoS management categories; Directed, Collaborative and Virtual. These categories define the span of managerial control across the SoS from centralized control - referred to as directed - to a lack of centralized control, defined as virtual. This architectural description sets the stage to define a SoS graph where the entity systems are nodes and the edges between the entities are relationships that can be defined by the SoS management categories.

In addition to Maier’s management types of directed, collaborative and virtual, a fourth type of SoS management, Acknowledged, is defined by Dahmann (2008). The acknowledged SoS has recognized SoS mission goals with less centralized control than a directed SoS. The acknowledged type of SoS is viewed as being more typical of DoD SoS than either directed or collaborative.

The United States Department of Defense, Office of the Deputy Assistant Secretary of Defense for Systems Engineering, DASD(SE), provides a definition of SoSE for the DoD and how it relates to traditional SE conducted on entity systems. In this description SoSE deals with the employment of existing systems focused on specific missions, and provides for contextual constraint to the acquisition-based, classical SE process. The DASD(SE) (2008) guidance defines seven SoSE core elements:

- Translating capability objectives
- Understanding systems and relationships
- Assessing performance to capability objectives
- Developing and evolving an SoS architecture
- Monitoring and assessing changes
- Addressing requirements and solution options
- Orchestrating upgrades to SoS.

These core elements of SoSE then work in collaboration with the entity systems and their respective classical SE processes. Key SoSE artifacts have been defined to include association with the above SoSE core elements (DASD(SE), 2008; Dahmann, 2011a).

The SoSE domain, unlike domain of classical SE found within the practices of the entity systems, is recognized to be dynamic in nature such that the seven core DASD(SE) SoSE elements are not necessarily conducted sequentially as found in the classic SE ‘V’. This creates a distinction between traditional SE processes about a multi-year acquisition program where there is a need to minimize changing requirements, and the SoSE domain where the ‘real world’ environment can result in continuously evolving or even dynamic mission requirements changes. This notion of the dynamic and iterative nature of SoSE is explicitly reinforced with Dahmann’s (2011b) introduction of the “Wave Model” where the period of SoSE change from perturbations in mission goals leading to change in SoS
employment is not necessary aligned with or is perhaps independent of the acquisition cycles of the entity systems.

SoSE is not unique to DoD. The manufacturing community has produced work relevant to the SoSE domain termed Holonic Manufacturing Systems (HMS) that has focused on inter-entity relationships. The entities, Holons, refer to a basic unit of organizations that are autonomous, self-reliant and cooperative. Holonic Manufacturing is characterized by distributed, autonomous and adaptive manufacturing systems, where hierarchy is needed to maintain focus on mission goals by facilitating inter-entity conflict resolution (Leito, 2009). The manufacturing environment is defined as being non-linear, complex, chaotic and uncertain and constantly adapting to stakeholder pressures. In this environment emergence and self organization are embraced where emergence provides both flexibility and robustness, and self-organization provides for dynamic adaptability. HMS are self-adaptive and self organizing and are limited by saturation in computation and communication capabilities, both human, machine and software-based, across the hierarchically organized holon networks or graphs (Valckenaers, 2008). Holons are architected so that they can take direction from above or react from stimuli from below. Individual holons can also be comprised of sub-networks, sub-graphs, of holons. From a management perspective HMS are dynamic, and akin to Dahmann’s Wave Model oscillate between cooperative (less centralized) or collaborative (more centralized) as triggered by changes in mission goals or the mission environment. Negotiation across the holons in HMS hierarchy is essential to successfully deal with conflict, and incentives to work toward the common mission goal are explicitly considered and acted upon. The negotiations tend be the driver for moving between cooperative and collaborative behaviors. Leitao (2009) brings in the biologically inspired technologies as enablers to provide for dynamic evolution in the HMS.

From the review of the literature SoSE is an emerging discipline that has set initial defining bounds. The practice of SoSE is reasonably attempting to extend SE tools, techniques and processes, e.g., Model-Based System Engineering (MBSE) ranging from the Systems Modeling language (SysML, 2013) and the Unified Modeling language (UML, 2013) to Department of Defense Architectural Framework (DoDAF, 2013), with an ad hoc collection of artifacts based on a variety of data structures ranging from text data to relational databases and MBSE. There is little evidence in the literature of work to establish common metamodels containing supporting databases for SoSE. In this paper such a metamodel is defined as the basis upon which the necessary SoSE engineering artifacts, to include relationships are explicitly defined, stored and managed across the lifecycle. The proposed metamodel is consistent with the DoDAF DM2 Metamodel consisting of conceptual, logical and physical layers. The concept layer, sometimes referred to as ontology, defines concepts and concept relationships. The logical layer adds technical information and unambiguously defines relationships, e.g., MBSE artifacts, different types of simulations at various levels of abstraction, etc. The logical level initially contains architectural detail and subsequently contains design, build, test, and employment artifacts. The physical layer is the central database repository that contains the content from both the conceptual and logical layers as well as simulation input and output data sets where large, complex data stores can be readily queried and reused. It is proposed that for a SoS the physical layer needs to be consistent with the graphical nature of a SoS to manage both entity and relationship data.

While the rich SE basis is necessary and essential to SoSE, it is insufficient. SoSE needs a holistic means, i.e., specific and quantitative tools techniques and procedures, to define ‘how to’ implement SoSE. It is proposed that the transition from the combined SE and SoSE domains must be based on a common metamodel and it is proposed that the necessary structure must be a graph-based. The metamodel is focused on the inter-entity systems interfaces, interactions and relationships. The metamodel provides for a common model basis within the logical layer from the build of the SoS and a corresponding run-time environment of live, virtual and/or constructive simulations. In addition the physical layer provides a database to contain SoS model data along with the big data associated with all the associated simulation input and output data. This enables near real-time testing during SoS
development and employment. The metamodel content must also include mission goals, a mission environment and mission threads, and be extensible across the SoSE lifecycle. The proposed process is outlined in Figure 2; a process description consistent with the current SoS literature, as well as and DoD SE and SOSE guidance.

II. Modeling and Simulation Enablers

SoSE is a very complex, diverse and dynamic domain. To interrogate the entirety of this SoS domain space the SoSE tools, techniques and processes will be modeling and simulation (M&S) centric and dependent upon the exploitation of high performance computing to access all the engineering artifacts in a timely, cost effective manner. An emerging capability that may be able to empower and integrate across the SE and SoSE domains by exploiting high performance computing is capability offered by graph databases (Hogg, 2013). Graph databases are becoming best practice where inter-entity relationships; very large, complex data sets; and query speeds are primary concerns. These application areas include the internet and social media. In social media, exploitation of graph database technology includes Google’s Pregel and Knowledge Graph, Facebook’s Social Graph, and Twitter’s Interest Graph (Malewicz, 2010; Google, 2013; Facebook, 2013; Russell, 2013). An example of a graph showing the reach of Twitter is shown in Figure 3. The Twitter graph (Smith 2013) represents a quantitative, graph-based analysis including text recognition of Twitter users whose tweets include the word “Hadoop” (The term Hadoop comes from the Apache Hadoop software library, a framework that allows for the distributed processing of large data sets across clusters of computers (Hadoop, 2013).

Perhaps the capabilities afforded by graph databases such as those offered commercially in neo4j and Titan graph database (neo4j, 2013; Titan, 2013) and described in the literature by Miller (2013) and McLendon (2010) can provide the disruptive technology need by the SoSE community in a manner similar to that social media has achieved.

SoS simulations have historically been centered on entity systems, reliant upon traditional equation-based models where interactions and relationships between entity systems tend to be implicitly represented. In this area of simulation DoD has driven the development and use of federated and distributed discrete event simulations as presented in a 1995 Modeling and Simulation Master Plan (M&S, 1995). These simulations tend toward being both high fidelity and high resolution in nature, and thus require significant time to develop, field and integrate, and run in their various compositions. There are three categories of these simulations referred to as Live, Virtual, and Constructive compositions used for a variety of intended uses, e.g., virtual prototyping and testing, training, exercises, etc. The definitions (M&S, 1998) for live, virtual and constructive are:

Live – Live participants operate operational systems in their physical operational environment,
Virtual – Live participants operate simulators and operational systems in a synthetic environment,
Constructive – Simulated people operating simulated systems in a synthetic environment.

Frequently a simulation composition is hybrid and has some characteristics of each type. Thus, these simulations tend to be about real-time in execution speed so the number of simulations that can be run is limited.

While success using discrete event simulation techniques has been achieved for systems and complex systems, extensions to the SoS domain, especially using federated models, frequently do not meet stakeholder expectations. These reasons include composability and management challenges, interoperability challenges between simulations in the federate, ever increasing complexity, inappropriate reuse of legacy models for other than the intended use, lack of management incentives for the members of the federation to bring forward the most appropriate models, and unrealistic expectations, e.g. the models being ‘plug and play’ capable. Literature on this topic (Davis, 2004; Nance, 1999; Rioux, 2002; Tolk, 2003; Henrksen, 2008) has been produced over the decades with significant actionable and constructive critique. As a result, the integration of the models into the federation composition involves much time consuming database manipulation that is unique to each composition and intended use. The cost of this integration time has been a consistent criticism of DoD federated simulations.
While much has been written from the perspective of simulation engineering about the need for common data models, the use of ontology, and/or managing inter-model relationships through metadata in an attempt to achieve semantics and timely integration of a composition, the literature hints at other potential issues. These federated simulation compositions as well as the associated entity models tend to be integrated using software wrappers, middleware and relational databases, perhaps the challenges are software engineering challenges associated in managing inter-database relationships across the simulation compositions via ‘join’ operations and timely queries over complex data stores.

Perhaps other simulation techniques could be added to the M&S toolbox to augment the current reliance on the federated discrete event simulation. When a SoS graph can be poised explicitly as a directed acyclic graph probabilistic techniques such as a Markov analysis or Bayesian statistics can assess the conditional probabilities of success over the graph. From these probabilistic methods the inherent causal interdependencies and uncertainty found in a SoS can be explored directly (Han, 2012; Mane, 2011). These probabilistic techniques are dominated by matrix algebra and very fast running, perhaps millions of runs per day can be realized. The probabilistic simulations can readily establish bounds about a multi-parameter performance space and/or guide construction of architectures and models for subsequent higher resolution and fidelity simulations.

Agent-based simulations are suited to the SoS domain when inter-entity relationships have similar or greater importance than the performance of individual entities. Agents are network nodes with prescribed entity system behaviors, and the network edges are prescribed inter-agent relationships. Agent-based techniques provide a means to look at the SoS from a different perspective and readily deal with networks and inter-agent interactions to include human social factors, non-linearity in agent behavior and/or coupling, or the absence of explicit mathematical solutions. Thus the agent-based simulation is inherently graphical. Further, due to their fast running nature, hundreds to thousands of runs can be realized per day. Agent-based simulation can be used to investigate emergence and stochastic behavior of the inter-entity relationships (Rouse, 2003; Axtel, 2000; Bonabeau, 2002; Axelrod, 2006; Shen, 2006).

Models and any subsequent simulations have inherent uncertainties based on abstraction from the real world. DeLaurentis (2000) defined uncertainty as “the incompleteness in knowledge (either in information or context), that causes model-based predictions to differ from reality in a manner described by some distribution function” (p. 3) Oberkampf (2002) emphasizes uncertainty in models based focusing on both aleatory and epistemic uncertainty and provides for a distinction between the system models and the environment. Aleatory uncertainty is based upon having a priori knowledge, while epistemic uncertainty is based on a lack of knowledge. Both DeLaurentis and Oberkampf discuss from the perspective of aerospace design analysis that uncertainty is involved throughout modeling and simulation process to include uncertainty in empirical measurements, conceptual modeling, mathematical modeling algorithm development, computer programming, numerical solutions to include input parameters and the representation of the simulation results. Ferson has looked at uncertainty through the lens of risk analysis using a fault tree approach to model system failure. Fault trees are directed acyclic graphs where each node in the tree can be expressed as a scalar, probability distribution function, interval, or probability box for performance, schedule, and/or cost, Figure 3. The data on which these four representations are based can be empirical measurements, models or simulations, or even human intuition (Ferson, 2003; Ferson, 2004; Ferson, 2007). Thus the quantitative artifacts for any/all simulations, whether inputs or outputs, should be considered uncertain, and presented using one of these representations. This will then explicitly provide uncertainty bounds for behaviors represented in a simulation.
III. The Interstitials: a Move to a Graph Modeling Basis for SoSE

Garrett, et al. (2010) presented the idea that the essence of SoSE is based not on the characteristics of the entity systems, but is instead based on the relationships and interactions between the entities that comprise the SoS. These relationships and interactions are defined as the interstitials. To enable SoSE, the use of a graph mathematics basis was proposed to explicitly model the SoS providing for bounding context for MBSE based architectures, models and simulations. To describe the SoS in its physical space, the SoS graph is considered to consist of physical entity systems, e.g. ships, airplanes, facilities, people, etc., as nodes and the interactions between these nodes are the edges, i.e. the interstitials. When the SoS is employed to engage external stimuli to achieve a mission goal, the graph is expanded to a multigraph where the possible aggregate event sequences are explicitly represented as multiple, plausible edges between two nodes. As an example the event basis used to define each edge can be that of a traditional ‘kill chain’ function, i.e., Detect, Track, Engage, or Assess; a control model, i.e. control, sense, decide, or respond (Johnson, 2013); or as an OODA Loop, i.e. Observe, Orient, Decide, or Act (Boyd, 1976; 1987; 2007). An OODA based decomposition will be used throughout this paper for the sake of demonstration.

To describe the SoS in its Event Space this graphical construct can then be used to propose that a mission is composed of multiple threads, and each mission thread is a composite vector of linked active walks through the mission graph (Garrett, 2010). Each mission thread is then a sub-graph, or digital thread, that explicitly defines the active event sequencing through the prosecution of a mission starting with the introduction of a stimulus and ending with the completion of the mission goal; this graphical representation of the SoS will be referred to as an event space. Explicitly the edges, i.e. the inter-entity relationships, of the physical space multigraph are transposed to the nodes of the event space directed acyclic graph.

Transforming the SoS graph from its physical space to the associated event space then provides for a mathematical formalism that can be used to begin to quantitatively and explicitly model a SoS from both the perspective of an entity centric physical space and that of a relationship centric event space. The results forms a quantitative basis explicitly providing a means to build probabilistic, agent-based and discrete event simulations from a common graph-based architecture. The event space also provides a means to quantitatively architect testing about an event-based decomposition with a means to readily translate back to the physical space.

The DoD testing community has invested significant effort to define a basis and an approach for testing components or entity systems within an SoS under clearly defined mission context (JITEM, 2009). Two key concepts are used regarding the mission context: the mission environment (ME) and the mission thread (MT). The ME is the set of physical entities, conditions, circumstances and influences to meet a specific mission goal that can be represented as a set of interdependent networks or graphs. The MT is the path or walk within the ME set that provides the operational and technical description of the end-to-end set of activities and systems consistent with the formulation of the SoS physical space and event space as described above.

Returning to the physical space formalism, an example is the simplistic, notional multigraph of Figure 4. Here events explicitly occur between the entities in the prosecution of a notional mission with each edge an explicit event. Where an entity must conduct an internal process before interacting with another entity a self referential edge or loop is added. Sub-graphs are also shown that represent event threads, called walks, in the graph. The walks can be viewed as individual critical subcomponent models in the SoS with the potential for reuse in subsequent SoS modeling activities.
Using the walks of Figure 4 as an example, Figure 5 shows two walks in both physical space and an event space in a notional mission environment. The event space is a directed acyclic graph in that it has only single directed edges and no directed cycles within the graph. The probability of success for each node in the event space can then be represented probabilistically as a scalar, probability distribution function, interval, and probability box representation as shown previously. Both the physical and event space graphs represent the interaction of the SoS with the stimulus, and are initiated upon the insertion of a stimulus. In this example, node 3 is a command and control system (C2), and node 5 is an actor that can ultimately ‘engage’ the stimulus. OODA is used to define the events; sensors observe, C2 decides and the interactions between nodes are orients.

The 1-3 walk ends on the same ‘decide’ event as walk 3-4 begins to facilitate merging the walks into a larger path. The new walk can then be merged with other walks as appropriate to create mission threads. Two additional plausible event space threads each with a different employment concept are constructed from the same physical space of the Figure 4 multi-graph and shown in Figure 6. It should be noted that the end point of these walks is the decision as whether to ‘act’. The role of the orient function is to provide for the communication between each entity system in physical space and between each observe, decide and act event space node and also includes functions like discerning, correlation and fusion.

This approach indicates that each entity system / node within the SoS must perform their own local orient functions prior to inter-system communication within the SoS. The orient function then has a dominant role in the domain of command, control, communications and computers (C4) a domain of software, information technology networks, human operators and human decision makers. The number and morphology of orient and orient-decide-orient functions within the SoS architecture then becomes a key attribute in quantifying inter-system relationships and thus driving the type of SoS management function; i.e., virtual, collaborative, acknowledged or directed as defined previously.

IV. The SoS Domain, multi-dimensional complexity

To this point, the SoS domain has been described as consisting of the set about mission goals, the mission environment, and mission thread that is an entity system and interstitial centric graph. The MT has been shown to be an interstitial centric graph. In this section a domain decomposition of the ME will be proposed adding significant content to the overall SoS graph. The complexity of this additional content will be a key driver for the move to a graph metamodel for SoSE. DeLaurentis et.al. (2004, 2005) introduced some specific dimensionality to the SoS domain proposing a lexicon of categories of systems and levels of organization. These interdependent categories are listed below with definitions expanded upon by the author:

- **Stakeholders-** Organizations and their inputs on mission goals, requirements and initial architectural components,
- **Resources** – The entity systems and sub-systems that comprise the nodes of the SoS Physical Space,
- **Operations** - Employment schemes for the SoS to meet mission goals in a ME for given MT, i.e. the SoS Event Spaces, and
- **Policies** – Constraining governance.

DeLaurentis goes on to discuss this basis about categories and levels with respect to the notion that the entity systems tend toward being dynamic and uncertain, and that decision making and control methods range from hierarchical control (directed) to fully distributed control (virtual).

This SoS domain defined by DeLaurentis (2004, 2008) and shown in Figure 7 is inherently graph based in its totality. Each of the categories at the α-level is a graph; intra and inter-category edges can be added to define
relationships. The $\beta$-level, $\gamma$-levels, $\delta$-levels, etc. add substructure to the required level of resolution. This space is proposed to represent the SoS mission environment where Mostafavi (2011) proposed it as a SoS framework from which to bound and build SoS models. Mission threads can now be explicitly linked in a common metamodel through and across all the categories, i.e. the ME, and to the mission goals to include relationships. Effects of changes in one part of the metamodel can be readily observed and their effects quantified by testing along the mission threads using models and/or simulations.

An example of an intra-category structure for the stakeholders was presented by Rowley (1997) where he modeled, based on social network theory, the domain as a graph using metrics based on network density and centrality to measure stakeholder influence. In Rowley’s work density is defined as the ratio number of edges in the undirected graph to the total possible number of edges. Centrality is essentially defined as ‘betweenness’; a measure of influence or power a node exerts on the network. The social science community has built a significant graph-mathematics basis to quantify centrality based on network flows (Freeman, 1997; Freeman, 1991).

This graph basis provides a quantitative means to define complexity of a SoS. In its simplest form complexity can be defined by the functions of the number of nodes, edges, and active paths/walks in a graph. McCabe (1976) defined a complexity measure for software, the cyclomatic number $V(G)$ of a graph $G$ with $n$ nodes, $e$ edges, and $p$ connected components where:

$$V(G) = e - n + p.$$  \hfill (1)

$V(G)$ then provides an upper bounds on the number of test cases/scenarios that must be employed to fully exercise the ME. Bonchev and Buck (2005) presented an extensive review of the topic covering graph based complexity measures from the perspective of chemical and molecular structures where the notions of sub-graph count and overall connectivity that are absolutely relevant to SoS complexity modeling.

V. Pulling together the pieces into a SoS graph-based metamodel

A Graph-based SoS domain of big data, interfaces and relationships, complexity and uncertainty has been described. As shown previously in Figure 2, The SoSE process is built around a SoS graph-based Metamodel where analyses are conducted on mission threads within the context of the mission environment to meet mission goals to define SoS architectures, both tactical and virtual (from which simulations will be composed). This graph-based metamodel provides a means to manage artifacts and conduct concurrent engineering for both the SoS and the SoS run-time environments enabling simulation-based design as shown in Figure 8. This metamodel approach is consistent with the vision for simulation based design presented by Jones, Graves and Gersh (1997) where the system/SoS model is used to create run-time simulations in near real-time. This approach uses MBSE tools to quickly compose simulations with minimal human intervention from a virtual collaborative environment, i.e., the graph-based metamodel. Thus virtual prototype testing can be accomplished in near real-time through the development cycle. This needs to be accomplished by building the conceptual, logical and physical layers of the metamodel concurrently so that the physical layer database is extensible over both the anticipated SoS complexity and lifecycle. Graves (2009) published the results of a pilot effort that demonstrated this simulation-based design methodology for an aircraft simulation composed directly from executable SysML design models.

Figure 9 maps the SoSE process with the DoD core SoSE elements previously presented (DASD(SE), 2008). The challenge is then one of appropriately converting a traditional document based set of artifacts to a graph basis for incorporation into the SoS graph metaodel, and using the metamodel as a means of establishing a common basis from which to manage the various entity and integration MBSE artifacts. While not within the scope of this paper, it appears that the knowledge management and self-adaptive software systems communities have tools and
techniques available to establish an initial baseline capability to translate the documents, and coordinate the MBSE data management activities.

The mission thread analyses, particularly with the probabilistic and agent-based simulations, can provide a description of the SoS performance space to include measures of uncertainty based on the nature of the casual inter-entity relationships. The uncertainties are defined as ranges about the probability of success across the mission threads for single and multiple stimuli. The uncertainty bound also includes both aleatory and epistemic uncertainties from multiple sources. Uncertainty quantification is conducted at both the micro and macro levels. At the micro level, uncertainty quantification interrogates the aleatory and epistemic uncertainties about nodes, edges and walks (both physical and event space representations) and aggregates these uncertainties. At the macro level uncertainty tolerance bounds are established for event space-based mission threads and physical space-based sub-graphs, both inter and intra-category within the mission environment. It is proposed that these uncertainty analyses could be aggregated within the SoS metamodel to create uncertainty contours about sub-graphs. By iterating through and evolving the SoS metamodel, these graphs with uncertainty contours and subsequent analyses can guide the evolution of mission goals and mission environment, in any combination of categories, and/or evaluate the effect of changes to the goals or environment from external influences by using optimization techniques to deliberately reduce uncertainty in a given performance parameter space.

These uncertainty analyses can then interrogate the effect of SoS management concepts on performance. For example, in a SoS with a given number of sense and act functions, the architectural differences between centralized and decentralized management appears to be in the complexity of the hierarchical C4 infrastructure, i.e., the orient-decide-orient functionality. Here complexity is a function of the number of nodes, the number of edges, and the number of active walks involved in meeting a specific mission goal. It is proposed that increasing centralization of management is directly proportional to increasing SoS complexity. To that end the mission success of a decentralized ‘swarm’ of entities conducting a mission goal will be strongly dependent upon the success of the sense and act functions of the individual, distributed entities. While in an acknowledged or directed SoS the mission success will be driven primarily by the orient functions (needed interactions), and secondarily by the decide functions necessary to get a message to the central decider, and back down to the sensors and actors. As such the success of the acknowledged or directed SoS in meeting a mission goal is not necessarily dictated by the ability to sense or act, but in fact is driven by uncertainty caused by latency, saturation, lack of semantics, and the total time it takes to make and propagate decisions up and down the hierarchy of the graph. Agent-based simulations then can be used to inter-entity relationships, i.e., what incentive do the self-governing sensors and actors have act upon a distant decision that is inherently uncertain, and thus poses local risk to performance success, or even to their survival? Perhaps the work on Holonic Manufacturing Systems where the SoS is managed dynamically between the bounds of cooperative and collaborative management can achieve a higher probability of mission success with smaller levels of uncertainty than the management structure of an acknowledged or directed SoS.

From the architecture, quantitative SoS interface definitions can be defined and bounded. These interfaces are explicitly OODA orient functions, either a single orient, or thread of multiple orient functions, to include both inter and intra element functionality to assure semantics. Other constructs can be used as long as the communications between nodes address interface behaviors both syntactically and semantically. Using the HMS approach these interface descriptions need to address the nonlinearity and uncertainties associated with C4 saturation for machine, hardware, software, and human contributions. The inter-entity interfaces behaviors can be defined quantitatively as bounded probability distribution functions with uncertainty bounds as appropriate, e.g. those defined by Ferson in Figure 3, and need to address both the syntactic and semantic characteristics of the
relationships. The interfaces also need to have supporting metadata to further populate the edge characteristics of the graph database to include standards and specifications and engineering detail about the relationship.

A SoS performance assessment plan can be developed serving two purposes: to define SoS context for entity system testing and to provide explicit test conditions for testing the SoS performance. From the dynamically evolving metamodel, test threads can be defined from mission threads and prioritized using uncertainty measures. For each test thread statistically based metrics and measures can be defined and bounded. From the uncertainty analyses test points can then be established and prioritized about well defined interfaces, and the requirements for ‘test probes’ can be obtained if necessary. Uncertainty can also guide and prioritize verification and validation activities since the dynamic nature of the mission goals and mission environment, emergence, and self-adaptive evolution render traditional V&V practices problematic (VV&UQ, 2012). This approach can enable uncertainty based test definition and validation planning much earlier in the SoSE process, creating a potential break from the classic approach of testing (real or through simulation) and subsequent verification and validation toward the end of the SE process. In fact, the establishment of a test plan can begin as part of concurrent, dynamically iterative, requirements engineering, architecting, and test planning phases linked through the metamodel (Tamura, 2013).

VI. Conclusions

SoSE needs a wide focus on the totality of a very complex SoS domain of multiple entities, and complex and dynamic inert-entity relationships to include the evolving nature of a ME over a SoS lifecycle. The bounds of the SoS need to be carefully considered, and the extensible metamodel created capable of spanning both the SoS and the run-time environment to include mission goals, the totality of the ME and MT content that describe various SoS employment strategies involved in meeting the goals. This is a domain of big data, particularly that data being generated by simulations. For these SoS simulations to be useful in SoSE virtual testing construction of the needed compositions must be timely necessitating the need for model-based simulations. Exploiting the graph database capabilities being applied by the internet and social media allow for managing the relationship rich, big data of the run-time environment to include models and the simulations input and output data stores.

The decomposition of the ME into bounding MT form the basis for the synthesis of SoSE artifacts: architectures, defined interfaces, and established performance assessment. Model-based simulation tools techniques and procedures are the heart of the SoSE toolbox. Graph-based probabilistic simulation tools need to be considered to address the probability of mission success to include causality, and both aleatory and epistemic uncertainty tolerance bounds. These probabilistic analyses provide bounding context for architectures, models and simulations. When inter-entity system relationships are important, agent-based simulation techniques are appropriate tools. For example, the performance of the SoS in the prosecution of a mission goal needs to be evaluated across the various approaches to SoS management as this may drive emergence. Bounding probabilistic simulations followed by agent-based simulations would be the preferred approach given the statistical nature of management schemes. When details of intra-entity systems need to be considered, classic discrete event simulation are most appropriate. For a given level of abstraction each of these types of simulations can be composed from a common architecture and model basis. A balanced use of the three techniques can provide for a robust perspective of the SoS performance space. A SoS with a given number of sense and act functions, the architectural differences between centralized and decentralized management appears to be in the complexity of the hierarchical C4 infrastructure. It is proposed that increasing centralization of management is directly proportional to increasing SoS complexity. To that end the mission success of a decentralized ‘swarm’ of entities conducting a mission goal will be strongly dependent upon the success of the sense and act functions of the individual, distributed entities. While in an acknowledged or directed SoS the mission success will be driven primarily by the orient functions, and secondarily by the decide functions necessary to get a message to the central
decider, and back down to the sensors and actors. As such the success of the acknowledged or directed SoS in meeting a mission goal is not necessarily dictated by the ability to sense or act, but in fact is driven by uncertainty caused by latency, saturation, lack of semantics, and the total time it takes to make and propagate decisions up and down the hierarchy of the graph. Perhaps this metamodel enabled, model-based simulation portfolio can readily provide quantitative insight into SoS architecture challenges/questions of what does an effective successful SoS architecture look like and what are the core enabling SoS characteristics?

The SoSE community would greatly benefit from open empirical data sets to guide the development of SoSE tools techniques or subsequent verification and validation. What is needed are data with sufficient complexity, yet highly abstracted, statistically significant, and perhaps most importantly, spanning many types of C4 hierarchical architectures. These data would need to also be diverse in character to include text, metadata, model-based artifacts (e.g., DoDAF, SysML, UML), simulation inputs and outputs, etc.
References


Miller (2013). Miller, Justin J. "Graph Database Applications and Concepts with Neo4j.". Retrieved from sais.aisnet.org/2013/MillerI.pdf


Figure 1: The proposed SoSE implementation model based on a SoS Graph-Based Metamodel

Figure 2: An example of a big data graph; a visual graph showing 1000 Twitter users whose tweets contain the word “Hadoop” on the right (Smith, 2013).

Figure 3: Scalar, probability distribution function, interval, and probability box representations for performance indicating a notional probability of success based on a priori knowledge or lack thereof.
Figure 4: A SoS multigraph indicating event sequencing and a set of walks from the multigraph

Figure 5: The creation of an Event Space from physical space using OODA by translating Physical Space edges to Event Space nodes
Figure 6: Plausible event space representation given different employment schemes from a single physical space.

Sensor (node 2) independently acquires information informing the controller (node 3).

Sensor (node 3) is tasked to discriminate the stimulus observed by both sensors (node 1 & 2).

Sensors 1, 2 & 3 and the node 3 C2

Sensors 1,2 & 3 and the node 3 C2, different architecture

Sensors 1 & 2 attempt to fuse their tracks.

Figure 7: The SoS Enterprise domain defined as a graph based on categories and abstracted by levels.
Figure 8: The metamodel provides the enabling means for simulation-based design where the SoS simulations and the SoS are composed from a common basis containing models as well as the run-time inputs and outputs.

Figure 9: The mapping of the SoSE implementation model with the DoD defined SoSE core elements (shaded boxes).