View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Calhoun, Institutional Archive of the Naval Postgraduate School

Calhoun: The NPS Institutional Archive

Faculty and Researcher Publications

Faculty and Researcher Publications

2012

Connections in System of Systems

Osmundson, John S.

Published and used by INCOSE with permission.

http://hdl.handle.net/10945/44982

http://www.nps.edu/library

Calhoun is a project of the Dudley Knox Library at NPS, furthering the precepts and goals of open government and government transparency. All information contained herein has been approved for release by the NPS Public Affairs Officer.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

Connections in System of Systems

John S. Osmundson
Department of Information Sciences
Naval Postgraduate School
1 University Circle
Monterey, CA 93943, USA
josmundson@nps.edu

Gary O. Langford
Department of Systems Engineering
Naval Postgraduate School
1 University Circle
Monterey, CA 93943
golangfo@nps.edu

Copyright © 2012 by John Osmundson and Gary Langford. Published and used by INCOSE with permission.

Abstract. Systems of systems are becoming more important in today's global endeavors. In this paper use of model-based systems engineering is examined from the viewpoint of understanding connections, that is the transfer of items between system of systems. The conclusion is that systems engineering models must include executable models in order to best understand the effects of transfer of items in system of systems.

1.0 Introduction

In this paper the importance of SoS connections is discussed, SoS connections are characterized, and the connection characterization is related to challenges of applying model based systems engineering to the development of analysis of SoSs.

There is increased interest in what has been become known as a system of systems (SoS) and the systems engineering of SoSs. In addition to military SoSs such as command, control, computer, communications and information (C4I) systems (Pei, 2000), intelligence, surveillance and reconnaissance (ISR) systems (Manthrope, 1996), intelligence collection management systems (Osmundson et al., 2006), there are important commercial SoSs such as electrical power distribution systems (Casazza and Delea, 2000) and financial systems, such as the transportation logistics networks. These systems are usually comprised of a large number of component systems and subsystems, human operators, and software agents.

There are many definitions of SoSs. A SoS is described by Maier and Rechtin (Maier and Rechtin, 2002) as systems which are operationally independent, managerially independent, evolutionary developed, with emergent behavior and are geographically distributed. (Madni, 2007) says that: "A SoS is a complex ensemble of independent systems developed and introduced over different time frames by multiple independent authorities to provide multiple, interdependent capabilities in support of multiple missions. The capability of a SoS typically exceeds the sum of the capabilities of the member systems."

There is not universal agreement on a definition of the term system of systems but many definitions have basic thoughts in common. (Sage and Cuppan, 2001) describe an SoS as

having operational and managerial independence of the individual systems as well as emergent behavior. Other definitions include operational and managerial independence, and geographical separations of the component systems. Here we will consider an SoS to consist of separately developed systems that are usually operated by separate organizational entities and are usually geographically dispersed.

Operational independence of the independent system elements of the SoS implies that if the SoS is disassembled into its component systems the component systems must be able to operate usefully independently. The SoS is composed of systems which are independent and operate usefully in own right. The component systems are often separately acquired and integrated, rather than designed and built as interoperable, independent systems. A SoS's development and existence is usually evolutionary with its functions and purposes modified with experience. An SoS may exhibit emergent behavior that cannot be localized to any component system.

An SoS may be developed and managed to fulfill specific purposes and may be centrally managed during long-term operation. The component systems maintain an ability to operate independently, but their normal operational mode is subordinated to the central managed purpose. For example an integrated air defense network is usually centrally managed to defend a region against enemy systems, although its component systems may operate independently. Collaborative SoSs are distinct from directed systems in that the central management organization does not have coercive power to manage the system (i.e. command, control, communicate, plan, organize, and operate as a team).

The component systems must, more or less, voluntarily collaborate to fulfill the agreed upon central purposes. From the perspective of users, the Internet is a model of a collaborative system, whereby derived benefits and network externalities incentivize participation, with the net result to carry out collaborative research (for example). From the perspective of managing the protocols and standards that enable the Internet, agreements among the central players on service provision and rejection provide enforcement mechanisms to achieve various levels of network performances. The Internet began as a directed system, controlled by the Advanced Research Projects Agency (ARPA, now DARPA), to share computer resources over telephone lines. Over time it has evolved from central control through unplanned collaborative mechanisms to a decentralized network that routed data to some, but not every node. Virtual systems lack a central management authority and may lack a centrally agreed upon purpose. Large scale behavior emerges, and may be desirable, but the SoS must rely upon relatively invisible mechanisms to maintain it. A virtual system may be deliberate or accidental. Familiar examples are the World Wide Web and national economies. National economies and the social systems that surround them can be thought of as virtual systems.

Because of the nature of SoSs the interconnections of the constituent elements may be very complex, the dynamics of the interactions may be complex, and there may be a wide range of possible probabilistic behaviors at all levels within the SoS, especially if human

actors are among the constituent elements. All of these factors make the problem of SoS systems engineering particularly challenging.

2.0 The Importance of Connections in Systems of Systems

In order for an SoS to provide more capability than any of the individual systems, the systems have to interact and, therefore, the key to understanding and engineering an SoS, as well as understanding emergent behavior, is clearly defining the system-to-system interfaces, connectivity, and exchanges. Several approaches to modeling complex systems have been developed recently, including that of (Oliver, Keliher and Keegan, 2007) and (Osmundson et al., 2004) that provide a clear elucidation of SoS interfaces and connections.

Systems Modeling Language (SysML) (OMG, 2010) has been introduced to support the specification, analysis, design, verification, and validation of complex systems. Systems engineers can describe large complex SoSs using SysML in the same manner that software engineers describe large software systems using Unified Modeling Language (UML.)

System elements in modeling language representations can be described as executable modeling icons by using modern software applications. This applies to system elements such as people and software agents, as well as hardware elements. These icons can be graphically linked to form models of physically distributed SoSs. Models of SoSs can be constructed in a modular manner so that design factors are represented by an association with modeling application objects. System options are represented by rearranging the objects and by varying the object attributes from model to model. Design of experiments guides the development of executable models and the running of simulations.

Interactions that provide new capability and other interactions that might result in emergent behavior may occur at interfaces between systems, between systems and operators and/or between systems and software agents. Each of these interface elements can be considered as elements that seek to satisfy a goal governed by a set of rules whose inputs are provided through the SoS interactions. In some cases the goal-seeking element may have probabilistic behavior and/or may adapt to changing input conditions.

In normal system design, the goals and the functional performances of the system and system elements are defined, subject to further iterations and refinements. In systems engineering, the system is designed to meet functional requirements to specified levels of performance. Now consider a complex system of systems where SoS elements may include people and software agents whose goals may be the same as, or different from, the original system goals. Unanticipated SoS behavior might then be due to miss-application of the SoS's rules by a person within the SoS. Another source of unanticipated behavior might be due to the fact a hardware element, a person, or a software agent correctly applied the rules, but for a set of input conditions that were unanticipated by the systems of systems engineers. A third source of unanticipated behavior might be adaptation of a person or a software agent to sets of inputs.

The systems engineering challenge is to understand the interconnections of a complex SoS and the potential behaviors resulting from the interconnections well enough to ensure the desired SoS capability will be met and that undesirable behaviors will be minimized if not prevented.

3.0 Characterization of Connections

Key aspects of SysML that specifically apply to interactions are the concepts of ports and flows. A port is an interaction point between a block or part and its environment that is connected with other ports via connectors. The main motivation for specifying such ports on system elements is to allow the design of modular reusable blocks, with clearly defined interfaces and connectivity.

Ports in SysML can be standard ports or flow ports. A standard port specifies the services provided (offered) by the owning block to its environment as well as the services that the owning block expects (requires) of its environment. The standard port is typed by the provided and/or required interfaces in order to specify the services.

A flow port specifies the input and output items that may flow between a block and its environment. The flow port is typed with a specification of what can flow. A flow port may be typed with a single type representing the items that flow in or out, or typed with a flow specification which lists multiple items that flow. Flow ports send items to their owning block or to an internal connector that connects with the owning block's internal parts. If a block handles the port interactions itself, then the port is called a behavior port. Where a flow port is a behavior port, the flowing items must be relayed either to/from some feature of its owning block. Alternatively the block could have a delegation port where responsibility is delegated to a nested part or parts, via ports on their interfaces. (Friedenthal, 2008) If a flow port is connected to multiple connectors, then the items can be sent simultaneously to all connectors that have matching properties at the other ends.

Items (called item flows in SysML) are things that flow between blocks and/or parts and across connectors and type matching between the items and ports is used to assist in interface compatibility analysis. The items flowing between flow ports must match the flow properties of the ports at each end. Items flowing from objects may be depicted in activity diagrams or as signals sent from state machines across a connector.

The SysML convention described above is simplified, as shown in Figure 1, in order to facilitate the discussion of SoS interactions. Objects in Figure 1 correspond to blocks in SysML and connectors in Figure 1 correspond to SysML ports. Items are the things that flow from object-to-object through connectors.

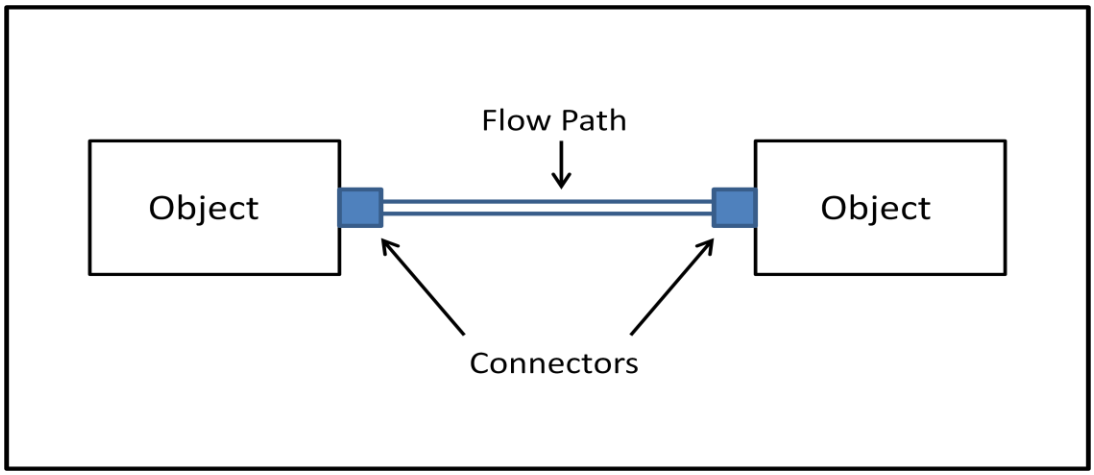

Figure 1. Object-to-object connections

Flow Item Types. When discussing flow ports the OMG's SysML specification (OMG, 2010) states that "Flow ports are interaction points through which data, material, or energy can enter or leave the owning block." (Langford, 2012) takes a broader view and states that objects interact with other objects through Energy, Matter, Material wealth, and Information (EMMI). EMMI expresses the interactions between objects. And, the conditions by which these flow ports send or receive items are termed the boundary conditions.

There is no attempt here to enumerate all of the possible flow item types, but rather give some examples categorized by major types. Adopting the concept of EMMI there are the following examples in each of the four major categories:

1. Matter. Items of matter include, as examples, parts, finished goods, fluids such as water or gas, and people. Typical SoSs involving the flow of items or matter are logistics systems, transportations systems and manufacturing systems.

2. Information. Information systems are perhaps the most commonly discussed type of SoS and examples of items of information include various forms of data, video and graphics.

3. Energy. Items of energy include electrical energy and kinetic energy. An example of a complex SoS that transmits electrical energy is the North American power grid (Osmundson, Huynh and Langford, 2008.) Automobiles are an example of a system (or some might argue a SoS) in which kinetic energy is transmitted through drive trains.

4. Material Wealth. Examples items of material wealth are the flow of material wealth are notes, currencies, and liens that are transmitted through banking systems andother monetary systems. The globalization of the economy is resulting in complex financial systems, some of which have exhibited emergent behavior (Osmundson, Langford and Huynh 2009), making this an area ripe for application of SoS systems engineering and analysis.

In addition to intended interactions (Ulrich and Eppinger, 2008) point out that there are unintended interactions, which they call incidental interactions that include vibration, friction, thermal interactions, RF interference, stray light and acoustic noise, as examples. These types of interactions may or may not be amenable to modeling in SysML but they do have to be included in a holistic SoS systems engineering analysis.

Interaction Protocols.. Interactions have requirements and constraints. Flows may be one-way or two-way. Flows may be synchronized or asynchronous. Flows may require a request to send, a request to be sent, and/or acknowledgements. Flows may be scheduled or associated with time constraints such as deadlines.

Connector Characteristics. Interface compatibility must be assured between connectors. This is most obvious for information system connectors that must assure that correct physical medium, link, and network layer requirements are met. A complication is that connectors may have state behavior; a simple example is a connector that can be in an "on" state or an "off" state. Further, a connector may be in an "on" state but unable to receive additional items because of a full buffer or lack of storage capacity.

Connectors may also have other time-varying conditions. Connectors may have some probability of failure or non-availability. A pipeline may be subject to corrosion over time that could affect its ability to transport matter.

Additional services that the responsible connector or block must provide are checks against constraints. Safety and security requirements must be met. Many monetary transactions must undergo validation checks: is the access ID correct, is there sufficient balance in the account, is the currency valid?

State Dependence. In addition to connector state dependence many SoSs have state dependence that impacts interactions. As an example, consider a layered missile defense system consisting of an outer layer, an intermediate layer and an inner – or point defense – layer. Each layer can and does operate as a separate system. Assume that the outer layer detects a long range missile attach aimed at a region defended by the intermediate layer, while at the same time the intermediate layer is engaged in responding to a mid-range attack on its defended area. The SoS battle manage determines an optimum target-weapon pairing solution that involves collaboration between the outer and inner layers and transmits the firing solutions to both layers. If the intermediate layer is in the midst of computing its own target weapon pairing solution it may be unable stop its current computations and switch to the SoS firing solutions within the engagement timeline constraints. This example would be difficult to describe using SysML typing and other conventions.

Probabilistic Conditions. SoSs are subject to probabilistic conditions and behavior. SoS conditions can go beyond normal limits of operation, people can exhibit unexpected behavior, there can be unanticipated confluences of conditions. Probabilistic conditions can effect SoS interactions in numerous ways. It would be difficult at best to describe

possible probabilistic conditions in a typical SysML model, yet these conditions may result in lack of SoS capability or undesirable behavior.

Timing. Blocks in a SySML model may have attributes associated with time, such as time of detection for a sensor, time that the detection message is sent to the communications network and time that the detection message is transmitted to a receiving node.

Active objects respond only when they execute a receive action and the point at which an active object responds is determined solely by the behavior of the active object. Once an active object has accepted a message, no further messages are accepted until the next receive action is executed. Waiting messages are stored in a buffer which can be a FIFO queue, priority queue, etc.

Passive objects respond when a message arrives, regardless of whether the object is occupied processing a previously arrived message.

In a SySML model the objects can have attributes, some of which are time-related. As examples, an object could have an item transmittal time and an item needed to receive time, both of these times being dynamic – in other words their values would be determined during execution of a thread. Assume an event triggers a thread, causing the execution of process. The length of time that process takes may be scenario dependent. For example, in a missile defense system, the length of time that a target-weapon pairing algorithm process requires can depend on the number of targets and the number of available weapons.

4.0 Implications for SoS Analysis

Model based systems engineering utilizing SysML offers many advantages: a means to describe systems and SoSs specifications, design elements and design rationale, test cases and interrelationships in standard modeling language using both text and graphics.

However, as (Soley, 2007) points out, SoSs are becoming more complex and that tendency leads to difficulties understanding designs and introduction of unexpected behaviors. (Madni, 2006) says that complexity in a SoS is imposed by the dynamic mix of systems and emergent behavior and the biggest challenge in SoSs is interoperability at the programmatic, constructive and operational levels. In order for an SoS to provide more capability than any of the individual systems, the systems have to interact and, therefore, the key to understanding and engineering an SoS, as well as understanding emergent behavior, is clearly defining the system-to-system interfaces and connectivity.

One of the important uses of models is to give insight into why things behave as they do and in this regard static SysML models do not give the necessary insight into potential complex behavior. Static SysML models representing systems and SoSs in terms of SysML constructs are useful in specifying interfaces and checking for interface consistencies. SysML also has mechanisms for specifying and representing time behavior

and probabilistic behavior, using constructs such as activity diagrams, sequence diagrams, state diagrams and system lifelines. These constructs, while very useful in specifying timing behaviors and timing constraints, do not easily lend themselves to consistency checking of complex time behavior. Likewise, while SysML models have mechanisms for specifying probabilistic behavior these do easily lend themselves to analyzing the effects of probabilistic behaviors on system and SoS performance.

SysML models must be augmented by dynamic models that are capable of incorporating time behavior, including state behavior, and probabilistic behavior. Some dynamic modeling approaches have been demonstrated that do in fact seem to allow analysis and understanding of system and SoS dynamic behavior. (Rao, Ramakrishnan and Dagli, 2008) have implemented executable models based on colored Petri nets to analysis of the global earth observation SoS while discrete event SoS models and simulations have been used to study emergent behavior in electrical power grids (Osmundson, Huynh and Langford, 2008), financial systems (Osmundson, Langford and Huynh, 2009) and regional economies (Osmundson, Huynh and Langford, 2011.)

System elements in SysML representations can be described as executable modeling icons by using modern software applications. This applies to people as well as hardware and software elements. These icons can be graphically linked to form models of physically distributed SoSs.

It is not always necessary to model an entire SoS in dynamic models. SysML models can be used to identify sequences of interactions that are end-to-end threads or segments of SoS end-to-end threads that are then converted into dynamic models. In some situations segments of large complex SoSs representing large numbers of similar elements – for example large numbers of people – might be treated as an aggregate with a range of probabilistic behavior.

As an example consider an electrical power generation and distribution system operating in a free market manner, that is, in an unregulated manner. Figure 2 shows a simple model of a highly abstracted power grid system, consisting of five power generating nodes, three trading agents, three consumer nodes, transmission lines linking the power generating nodes and the consumer nodes, and a transmission network router. The power generating nodes generate given rates of electricity at initial offering prices unique to each node. The consumer nodes have a rate of demand and initial buying prices unique to each node. The transmission lines have unique capacities and probability of failures that are dependent on their load relative to their capacity. The trading agents seek to maximize their profit by buying from the lowest price generator and selling to the highest bidder. If a generating node is successful in selling an increment of power, it raises its selling price, and lowers its price if it were unsuccessful. Consumer nodes lower their buying bid if they were successful in purchasing power and raise their bid if they were not successful. The behavior of the electrical power market can be analyzed by analyzing the dynamic behavior of the production, bidding and distribution process of the SoS.

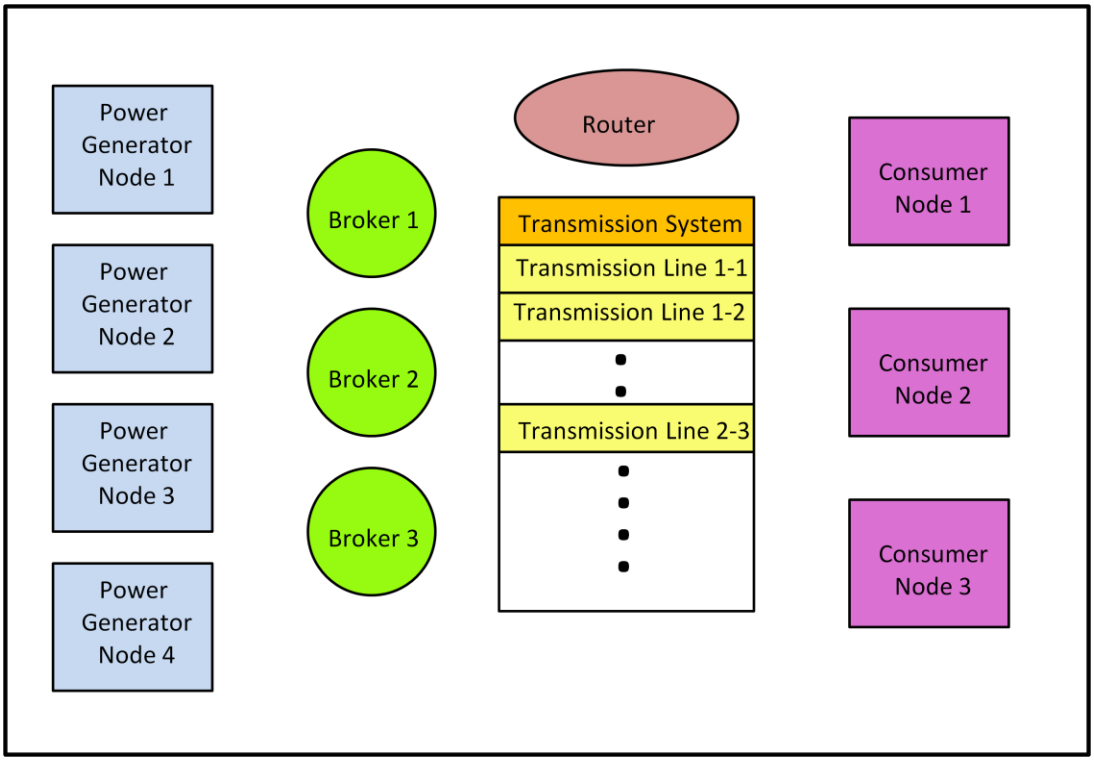

Figure 2. Representation of a model of an abstracted power grid system of systems

The model shown in figure 2 illustrates a transformation of a SysML representation of a electrical power grid SoS into a representation that can be further transformed into a discrete event model and simulation.

Figure 3 shows the bidding process involving one power producing node, one broker and one consumer in SysML activity diagram format. The actual bidding process, involving decision rules and delays cannot be captured in SysML diagrams nor can it be easily captured in supporting text. However the detailed bidding process can be captured in algorithms, delays and probabilistic behavior that is incorporated in dynamic models based on the SysML representation.

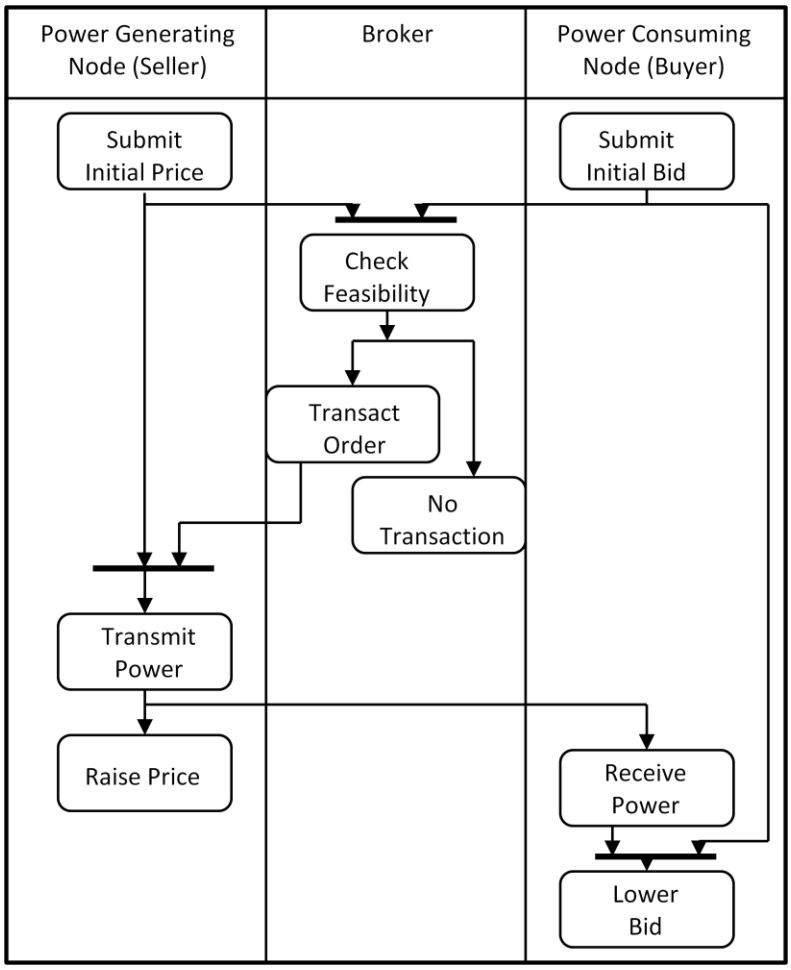

Figure 3. Simplified power system bidding process in SysML activity diagram format

Simulations of the dynamic model can be run and the time dependent price of electrical power can be obtained for various assumptions and operating conditions. Results could be used by systems engineers and systems analysts to study the effects on system behavior – in this example the market price of electricity – as a function of varying operating rules and conditions.

5.0 Conclusion

Understanding interactions is important in determining whether the desired new SoS capability can be realized and in attempting to determine whether any undesirable emergent behavior or potentially desirable emergent behavior is likely. Model based systems engineering using SysML has many of the features needed to develop a comprehensive view of a SoS that includes all of the important considerations of SoS interactions. However, the complexity many SysML SoS models may obscure the most important aspects of the interactions.

SysML models must be augmented by executable models that provide detailed representations of protocols, state behavior and other timing behavior, and probabilistic behaviors. SysML models can certainly inform the development of executable models, but the ability of executable models to include time dependent behavior and probabilistic behavior results in higher confidence in results of systems engineering analyses.

References

Choi, M. 2008. "Contesting *Imaginaires* in Death Rituals during the Northern Song Dynasty." PhD diss., University of Chicago (Chicago, IL, US).

Casazza, J. A. and F. Delea, *Understanding Electric Power Systems: An Overview of the Technology and the Marketplace*; Wiley: New York, 2003.

Friedenthal, Sanford; Alan Moore and Rick Steiner, *A Practical Guide to SysML*, Morgan Kaufmann OMG Press, Burlington, MA, 2008.

Langford, Gary, *Engineering Systems Integration: Theory, Metrics, and Methods*, CRC Press, Boca Raton, Florida, 2012.

Madni, A.M. "Architecture Tradeoff Analysis: A Disciplined Approach to Balancing Quality Requirements," Ground System Architectures Workshop *(GSAW)*, Architecture-Centric Evolution (ACE) of Software-Intensive Systems Presentation, March 27, 2007.

Madni, Azad M., "System-of-Systems Architecting: Critical Success Factors," USC-CSSE Convocation, Executive Workshop, 2006.

Manthrope Jr., W. H., "The Emerging Joint System-of-Systems: A Systems Engineering Challenge and Opportunity for APL," *John Hopkins SPL Technical Digest*, Vol. 17, No. 3, 1996, pp. 305-310.

Oliver, David W., Timothy P. Kelliher and James G. Keegan, Jr., *Engineering Complex Systems with Models and Objects*, McGraw-Hill, New York, 2007.

OMG Systems Modeling Language (OMG SysML^{TM}) Version 1.2, June, 2010.

Osmundson, John, Thomas Huynh and Gary Langford, "Systems Engineering of a Regional Economy", Systems Engineering, Test & Evaluation (SETE) Conference Proceedings, Canberra, Australia, 2-4 May, 2011.

Osmundson, John S., Gary O. Langford and Thomas V. Huynh, "Emergent Behavior in an Unregulated Financial System of Systems: Economic Meltdown," 2009 INCOSE International Symposium, Singapore, 20-24 July, 2009.

Osmundson, John S., Thomas V. Huynh and Gary O. Langford, "Emergent Behavior in System of Systems", CSER (Conference on Systems Engineering Research, Los Angeles, CA, 4-5 April, 2008.

Osmundson, John S., Nelson Irvine, Gordon Schacher, Jack Jensen, Gary Langford, Thomas Huynh and Richard Kimmel, "Application of System of Systems Engineering Methodology to Study of Joint Military Systems Interoperability," *Proceedings of the* 2nd Annual *System of Systems Engineering Conference*, Ft. Belvoir, VA, sponsored by the National Defense Industrial Association (NDIA) and OUSD AT&L, 25-26 July, 2006.

Osmundson, John, and Thomas Huynh, "Systems-of-Systems (SOS) Systems Engineering," *Procedings of the System of Systems Engineering Conference*,

Johnstown, PA, sponsored by the National Defense Industrial Association (NDIA) and OUSD AT&L, June 13-14, 2005.

Osmundson, John S., Russell Gottfried; Chee Yang Kum: Lau, Hui Boon; Lim,Wei Lian; Poh, Seng Wee Patrick; and Tan, Choo Thye, "Process Modeling: A Systems Engineering Tool for Analyzing Complex Systems," *Systems Engineering*, Vol. 7, No. 4, 2004, pp 320-337.

Pei, R. S., "Systems-of-Systems Integration (SoSI) – A Smart Way of Acquiring Army C4I2WS Systems", *Proceedings of the Summer Computer Simulation Conference*, 2000, pp. 574-579.

Rao, Madwaraj, Sreeram Ramakrishnan and Cihad Dagli, "Modeling and Simulation of Net Centric System of Systems Using Systems Modeling Language and Colored Petri-Nets: A Demonstration Using the Global Earth Observation System of Systems, *Systems Engineering*, Vol. 11, 3, pp. 203-220, 2008.

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

Soley, Richard M.,forward to: *Systems Engineering with SysML/UML* (by Tim Weilkiens), Morgan Kaufmann OMG Press, Burlington, MA, 2007.

Ulrich, Karl T., and Steven D. Eppinger, *Product Design and Development*, 4th Edition, pp. 176-77, McGraw-Hill, New York, NY, 2008.

Biographies

John S. Osmundson is an associate research professor with a joint appointment in the Systems Engineering and Information Sciences Departments at the Naval Postgraduate School in Monterey, CA. His research interest is applying systems engineering and computer modeling and simulation methodologies to the development of systems of systems architectures, performance models, and system trades of time-critical information systems. Prior to joining the Naval Postgraduate School in 1995, Dr. Osmundson worked for 23 years at Lockheed Missiles and Space Company (now Lockheed Martin Space Division) in Sunnyvale and Palo Alto, CA, as a systems engineer, systems engineering manager, and manager of advanced studies. Dr. Osmundson received a B.S. in physics from Stanford University and a Ph.D. in physics from the University of Maryland. He is a member of INCOSE.

Gary O. Langford is a senior lecturer in the Systems Engineering Department at the Naval Postgraduate School in Monterey, California. His research interests include the theory of systems engineering and its application to commercial and military competitiveness. Mr. Langford founded and ran five corporations – one NASDAQ listed. He was a NASA Ames Fellow. He received an A.B. in astronomy from UC Berkeley, and an M.S. in physics from Cal State Hayward. He is a member of INCOSE.