System of Systems Complexity: Modeling and Simulation Issues

Paul N. Lowe
The Boeing Company
paul.n.lowe@boeing.com

Michelle W. Chen
The Boeing Company
michelle.w.chen@boeing.com

Keywords:
Capability-based Acquisition – Complexity – Measures – Simulation – System of Systems

ABSTRACT: *Capability-based acquisition changes the general approach to systems engineering. Instead of buying threat-based, service-specific systems, a mobile target-weapon pairing system, for example, and then identifying how to integrate that system with other similar threat-based systems, we now identify the warfighting capabilities we want to achieve. Then we start with a 'blank sheet of paper' to develop the systems architecture and technical standards necessary to allow seamless interaction using shared data and applications. [1] Capability-based acquisition focuses on the development of alternative system of systems (SoS) architectures that link (i.e., network) diverse interoperable systems to optimize overarching capability effectiveness while minimizing development costs.*

It is up to the systems architect and engineer to find the best solution by examining inter-relationships between various configurations of warfighters, platforms, sensors, weapons, command and control, communications, while considering external factors that potentially impact full capability effectiveness and sustainment, such as operational doctrine, joint operations, government and societal organizations, terrorism, and public opinion. The emergence of asymmetric warfare, military-operations-other-than-war, effects-based and network-centric operations, national crisis response (and other homeland security missions), and other large scale problems where many diverse systems (often human individual and organizational decision-making systems) interact in complex ways, serves to complicate the process of developing an effectively interoperable system.

It is a complex undertaking to integrate a complex system of systems. Accordingly, the concept of system of systems has become increasingly important to systems engineering. One needs to understand the nature of the complexity of system of systems to gain insight into the design of robust, interoperable systems that integrate seamlessly into a system of systems environment.

Complexity - regardless of whether a system is hardware, software, or people - is generally a function of the number of entities (nodes) in the system, the number of relations (interfaces, interactions, networks), and time dependence (of interactions). Added complexity emerges when systems are capable of adapting their behavior to meet changing environmental conditions (complex adaptive systems) and when systems are capable of learning or evolving new behaviors. This paper explores a set of metrics that define the complexity of a system of systems simulation and presents preliminary work underway to apply these ideas for improving the quality and efficiency of the systems engineering process.

1. Systems of Systems

This paper provides a comprehensive, survey-level introduction to the relationships between systems of systems, complexity (including systems of systems as complex adaptive systems), and modeling and simulation. Some SoS complexity measures are identified and described in terms of (1) how well they capture important SoS complexity characteristics and (2) how modeling and simulation can be used to address SoS complexity issues.

1.1 System of Systems Definition

Systems of systems have been described in different ways, depending on the problem domain [2]. One of the simplest definitions is "a system of systems connects multiple systems to solve a large scale problem" [3].

Other definitions [2] emphasize the complexity of systems of systems.

As the problems addressed become more broadly scaled, involving large numbers of diverse, independently-operating systems (i.e., hardware, information, and human systems), SoS boundaries become more fluid, and systems become exposed to unanticipated situations due to changing environmental conditions. In order to remain operationally effective, a system therefore must alter its behavior or change its relationships with other systems. How this is done depends on the nature of the SoS – its complexity, order, and level of control over component systems.

Many modern systems – hardware and software – are complex in terms of large numbers of inter-connections between large numbers of components, but systems of systems can go beyond this into the realm of complex adaptive systems.

1.2 System of Systems Taxonomy

Gideon [4] provides a taxonomy for systems of systems that encompasses the continuum of SoS engineering, SoS operation, and SoS problem domain. Figure 1 depicts this taxonomy.

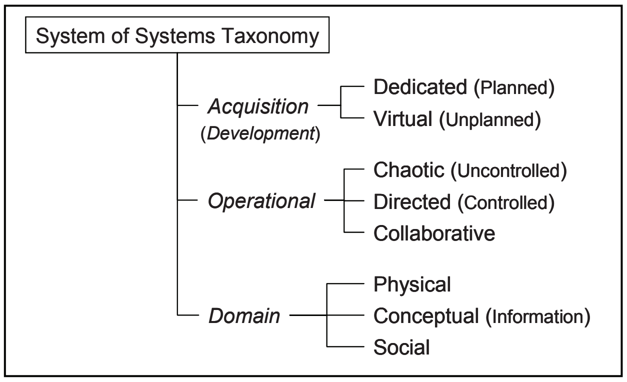


Figure 1: System of Systems Taxonomy

Systems of systems may be planned (e.g., the Global Climate Observing System) or unplanned (e.g., US educational system). A planned SoS is one where all component systems are purposefully designed and engineered, at the same time, to meet SoS requirements. An unplanned SoS is one where, over time, new systems are designed (and old systems replaced or removed) without benefit of a guiding set of SoS requirements, at times resulting in poorly integrated systems that do not interoperate to a fully effective SoS.

Operationally, systems of systems are either centrally controlled (e.g., network centric systems), collaborative (e.g., family or church), or uncontrolled (e.g., open source software development). Collaborative SoS provide standards to govern the behavior of component systems, yet are relatively helpless if a system chooses to disregard established procedure. Controlled SoS have the ability to enforce standards, so while systems may operate independently, they generally acquiesce to central mission control. Systems in an uncontrolled SoS operate completely independently, and it is only through their interactions with each other do SoS effects emerge. Naturally, controlled SoS are the most predictable and uncontrolled SoS the least predictable in achieving SoS effectiveness.

Systems of systems address physical (e.g., electrical grid) and information (e.g., intelligence gathering) problem domains. The physical domain includes those SoS which pass tangible, other-than-information material between systems and involve humans and hardware. The information domain, on the other hand, includes SoS that pass information, concepts, ideas, or plans between systems – these SoS may include human systems. The social domain addresses SoS that encompass both the physical and information domains, such as government or other socio-political entities, and focuses on human individual and organizational decision-making.

Gideon defines a SoS as a "collection of independently useful systems where the whole is greater than the sum of the parts; have emergent properties and behaviors that are not necessarily designed in nor expected; and continually evolve with new functionality added, removed, and modified through time and with experience." [4] His taxonomy shows that SoS are acquired and operate in a wide variety of circumstances and exist in different problem domains. Further, this taxonomy shows that overall SoS complexity varies with classification.

Maier [5] provides a simpler alternative taxonomy, which classifies SoS as directed, virtual, or collaborative, and relates to Gideon's taxonomy in the following way:

Maier	Gideon
Directed	Planned acquisition; Controlled operation
Virtual	Unplanned acquisition; Uncontrolled operation
Collaborative	Planned or Unplanned acquisition; Collaborative operation

Table 1: Taxonomy Comparison

1.3 System of Systems Characteristics

Maier [5] and other authors [2] distinguish a SoS from a monolithic system by describing SoS operational characteristics:

Operational system independence – when decoupled from the SoS, a system can and will operate independently in performance of its own mission (a system is useful in its own right).

Managerial system independence – each system in the SoS may also be acquired and integrated into the SoS separately from other systems.

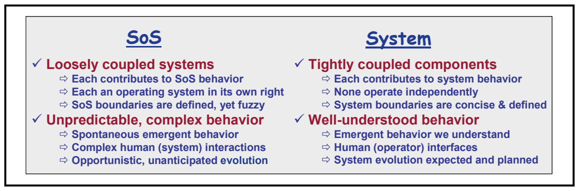
Evolutionary development – SoS functions are added, removed, and modified over time, as the need arises (e.g., technology insertion, changing environment, evolving requirements).

Emergent behavior – a SoS behaves, often unexpectedly, in ways that cannot be localized to a component system or anticipated during system or SoS design. Emergence addresses the spontaneous appearance, at the system of system level, of structures, patterns, and properties as a result of system interactions with other systems and the environment.

Geographic Distribution – Systems interact primarily through information exchange only.

Inter-disciplinary structure – SoS development requires application of multiple disciplines, such as modeling and simulation, system architecting, network theory, robust probabilistic design, and others.

Heterogeneity of systems – Diverse system capabilities interact to meet an emergent need, which is beneficial operationally, but complicates development (interfaces) and coordination.



SoS	System
✓ Loosely coupled systems	✓ Tightly coupled components
⇨ Each contributes to SoS behavior	⇨ Each contributes to system behavior
⇨ Each an operating system in its own right	⇨ None operate independently
⇨ SoS boundaries are defined, yet fuzzy	⇨ System boundaries are concise & defined
✓ Unpredictable, complex behavior	✓ Well-understood behavior
⇨ Spontaneous emergent behavior	⇨ Emergent behavior we understand
⇨ Complex human (system) interactions	⇨ Human (operator) interfaces
⇨ Opportunistic, unanticipated evolution	⇨ System evolution expected and planned

Figure 2: System of Systems Characteristics

Systems of networks – SoSs consist of nested networks of networks of systems, which if not present at initial deployment, will emerge over time due to system adaptation (to stay effective), self-organization, and environmental instability. Networks here refers to more than just communications – it includes collaboration networks and interaction chains, involving dynamic network formation and dissolution and aggregation of effects into self-organized clusters of cooperating systems.

Not all SoSs exhibit every characteristic all of the time; these characteristics are guidelines that serve to differentiate SoSs from complicated systems [6] and, further, point out some SoS features that have much in common with complex adaptive systems.

1.4 System of Systems Relationship to Complexity

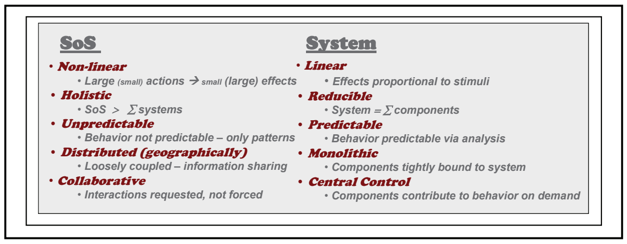
Complexity – regardless of whether a system is hardware, software, or people – is generally a function of the number of entities (nodes) in the system, the number of relations (interfaces, interactions, networks), and time dependence (of interactions) [7]. Added complexity emerges when systems are capable of adapting their behavior to meet changing environmental conditions (complex adaptive systems) and when systems are capable of learning or evolving new behaviors.

1.4.1 System versus System of Systems Complexity

A system, such as an aircraft or ground radar, may be complex in the sense of complicated and hard to understand, but it is ultimately amenable to traditional engineering analysis techniques. A system of systems, however, may exhibit additional complexity resulting from its operational behavior

Traditional systems engineering disciplines address complexity at a structural level. System interactions are limited to those external systems directly impacting the subject system’s mission, in an atmosphere of environmental conditions that change only within an anticipated range. Aggregate effects at the whole system level, if present at all, are typically ignored or passed off as random noise, treatable only by statistical averaging [8].

Systems resident in a SoS, on the other hand, often exhibit (or require) adaptive behaviors to remain effective in an atmosphere of changing environmental conditions and emergent SoS-level effects which impact system requirements. Systems and systems of systems exhibit different behavior operationally (Figure 2) and require different techniques to engineer (Figure 3).



SoS	System
• *Non-linear*	• *Linear*
• Large (small) actions → small (large) effects	• Effects proportional to stimuli
• *Holistic*	• *Reducible*
• SoS $>$ $\sum$systems	• System $= \sum$components
• *Unpredictable*	• *Predictable*
• Behavior not predictable – only patterns	• Behavior predictable via analysis
• *Distributed (geographically)*	• *Monolithic*
• Loosely coupled – information sharing	• Components tightly bound to system
• *Collaborative*	• *Central Control*
• Interactions requested, not forced	• Components contribute to behavior on demand

Figure 3: System vs. SoS Complexity

Systems comprising a SoS interact with each other and their environment, resulting in emergent effects and behavior at the SoS level. Some of these emergent effects are predictable, and some are not. Each component system has its own mission goals and operates independently, with varying levels of centralized control over its actions. Systems may self-organize dynamically into local behavioral networks to meet situational needs. Over time and often unpredictably, a system's environment changes, prompting the system to adapt in order to remain operationally effective. The SoS boundary is fuzzy and changes over time, as a function of evolving levels of interaction between SoS systems, the environment, and external systems. External systems enter the SoS boundary when their interactions begin to affect SoS behavior and leave when their contribution is negated.

Analysis of SoS performance is not amenable to classical linear (Newtonian) engineering analysis techniques. Nonlinear effects are common. SoS behavior prediction is difficult, if not impossible, due to continuing co-evolution of systems and environment. Determination of SoS effectiveness requires holistic analysis techniques, such as agent-based modeling and simulation, to capture locally dynamic system interactions and changing patterns of emergent SoS behavior over time. When we design a new system meant to interoperate with other systems within an integrated SoS environment, we must evaluate the system's impact on the SoS and vice versa– for that we essentially rely on modeling and simulation to capture dynamic system and SoS behavior.

1.4.2 System of Systems Engineering Complexity

Systems of systems impact system engineering in two ways –

- When engineering a system that will be integrated into a SoS
- When integrating multiple, independently-developed systems into an SoS as a Lead System Integrator (LSI)

More and more often in today's world of development, system engineers are being given the task of engineering systems that not only run on their own, but are planned for integration into a collection of other systems, creating a system of systems that synergistically is more useful than any of the component systems. The inherent complexity of a SoS is not necessarily tied to the complexity of any given component system; rather, the complexity results from the plethora of interactions among the components, some of which are old and some created specifically for the SoS.

Sheard [9] describes complexity in the systems engineering process, making a case that evolution of a SoS from scratch is slow and that it is better to start from something that is close (architecturally, functionally) to what you want. Key in the development of the SoS is the creation of the right environment rather than a very specific product, a SoS characteristically will evolve so such growth should be anticipated.

Schoening [10] discusses techniques to manage complexity in SoS engineering, focusing on contributors to significant disorder in the SoS. The top five identified contributors are:

- Requirements instability
- Inability to enforce behavior
- Changing objectives, strategy and tactics of external players – including threats
- Unpredictable environments
- Aging of "system" – fatigue

It is his contention that "efforts should focus on dealing with these contributors to decreased order and increased complexity rather than [looking at SoS] as some slightly vague entity whose only real difference with respect to a system is it is composed of individual systems." Of these contributors, requirements instability, influenced strongly by inability to enforce behavior, is the greatest challenge to engineering SoS.

Ernstoff [11] recommends system complexity as a first-order technical performance parameter in system development. The use of a SoS complexity measure turns out to have some interesting properties that are directly traceable to the characteristics of a SoS, in particular the large number of entities and paths between them. The measure discussed turns out to be relatively insensitive to input data errors because there are so many inputs, so as the central limit theorem suggests an input error in any one or two is not going to dramatically impact the final predictive results of the complexity measure.

Stutzke [3] looked at factors that contribute to making all phases of SoS development, from design through integration, a very complex enterprise. A subtlety of SoS complexity is that its challenges are qualitatively different than those of simple or merely complicated systems. These challenges arise because of the size of a SoS as it continues to grow larger and larger. The Global Climate Observing System (GCOS) is an observing network established in 1992 for international atmospheric, oceanographic, and terrestrial climate, created to ensure that the observations and information to address climate-related issues is available to potential users. Spread across dozens of countries, the logistics and complexity of GCOS continue to grow as technology evolves, the number of supporting countries grows and requirements for information become more sophisticated.

There is significant agreement in the literature that as the size of a SoS grows, the challenges associated with designing, developing, testing, validating, and maintaining a SoS moves from being just incredibly complicated to being so complex that they are on the border of being chaotic in all phases of development.

Cynefin [12] has described this continuum from a simple, well understood, system through complicated systems and complex systems into chaotic systems, in a decision making framework which characterizes the degree of complexity, particularly cause and effect, involved in the analysis of different systems or processes.

As seen in Figure 4, a simple system is one in which the relationship of cause and effect is obvious, and already-established best practices are immediately applicable. On the other end of the spectrum, a chaotic system is one where there is no relationship between cause and effect, at the system level, and one can only watch it operate. The relationship between cause and effect in a complicated system requires analysis or some other form of investigation to understand it – systems engineering typically addresses complicated systems. A complex system, on the other hand, is one in which the relationship between cause and effect can only be ascertained in retrospect – as with systems of systems analysis, we approach understanding it by probing (e.g.., modeling) the system and sensing (e.g. simulating) the patterns arising from its operation.

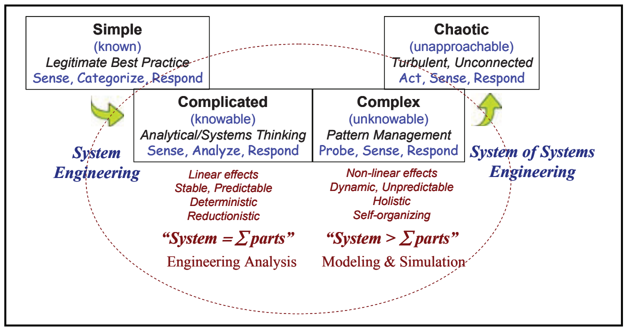


Figure 4: SoS Engineering Complexity

Krummenoehl [13] looks at six different views of a SoS. The functional view looks at the SoS from the classical perspective of functions performed, a view with which system engineers are well versed. The organizational view examines what can be accomplished with the SoS, an important facet in understanding the real value (return on investment) of development of the SoS. The operational view gives details of how the SoS will be used and what dependencies there may exist. The physical view is a detailed look at the "piece parts" needed to realize the design of the SoS, giving a hint at the complexity associated with the interconnections of the components of the SoS. The architecture view, the top level design of how the SoS functions are effectively achieved, is perhaps the most critical view. Most SoS can recover from badly implemented functions or even problems with the physical environment, but rarely can a SoS be saved from a bad architecture. Based on the architecture and the physical views, the information flow view describes what data or knowledge needs to be where and when. All five of the previous views impact this flow of information.

Meilich emphasizes modeling and simulation as a primary means of discovering the robustness of a SoS, identifies agility as a leading measure of SoS effectiveness, and states the goal of developing effective SoS is "embrace, manage, and hide [the] complexity of SoS …" [14].

1.5 System of Systems Analysis via M&S

Systems of systems are explicitly modeled in two different ways during the systems engineering lifecycle

Early SoS concept and architecture analysis is ordinarily conducted using as-fast-as-possible, non-distributed M&S infrastructures, such as system dynamics, discrete event simulation (DES), and agent based simulation (ABS). In these simulations, each system in the SoS is typically represented by a single simulation entity (i.e., subsystems are not modeled as independent entities), and the interfaces exhibited by an entity are limited to its potential interactions with other system entities.

The Department of Defense Architecture Framework (DoDAF) provides a means to graphically describe an SoS architecture and may be sufficient to estimate the static complexity of a SoS; however, elaboration of the DoDAF description into executable models and simulations is necessary to address SoS situational dynamics (i.e., runtime behavior). Osmundson [15] describes a SoS analysis methodology based on the use of executable DoDAF UML models, with several examples.

Downstream SoS integration analysis applies distributed, real-time simulation (e.g., federation) to assure component system integration at the SoS level, using industry-standard simulation infrastructure middleware, such as High Level Architecture (HLA) or Distributed Interactive Simulation (DIS). These simulations may involve human operators and/or system hardware, as opposed to constructive agent-based representations encountered in the as-fast-as-possible simulations. In these real-time simulations, subsystems may be represented as discrete simulation entities, and subsystem interfaces are likely to be explicitly modeled, affecting the degree of exhibited complexity.

The SoS complexity issues addressed in this document apply to either employment, but the implementation details will differ. Vendor-developed system simulations, integrated into a real-time federation, will provide enough system detail to enable measurement of additional sources of complexity, relative to the complexity measures possible with the relatively low-fidelity models used in the as-fast-as-possible architecture simulations.

It's important to emphasize that measurement of SoS complexity using M&S is in actuality the measurement of a simplified, modeled representation of a SoS. Contributors to complexity include those at the levels of model specification and software implementation.

Incorporation of the complexity measures, described below, into other-than-M&S engineering tools will provide additional understanding of the complexity inherent in a given SoS.

2. Modeling & Simulation Complexity

Modeling and simulation is a multi-disciplinary activity, touching on systems engineering, mathematical system modeling, and software engineering. Accordingly, software, model, and real-world system complexities are intertwined. While software complexity can be rationalized away (i.e., software must implement the same system-level functionality regardless of its own level of complexity), model complexity cannot. The system representation, as modeled, is always simpler than the system itself. This impacts the absolute complexity of the SoS – in other words, the complexity of two distinctly different SoS cannot be directly compared. However, the relative complexities of alternative architectures of the same SoS, using the same or similar models, should be comparable.

The following paragraphs describe complexity in terms of static (potential) complexity inherent in the mathematical models representing a system and dynamic (runtime) complexity exhibited by the models when exposed to simulated situations.

2.1 Simulation Software Complexity

The software engineering community has extensively addressed software complexity. Halstead complexity metrics measure software complexity directly from its source code, with emphasis on computational complexity involving numbers of operators and operands [16]. Cyclomatic complexity represents a broad measure of software soundness, emphasizing the number of linearly-independent paths (i.e., execution networks) through a program module [17].

2.2 System Model Complexity

Model complexity has been defined in various ways [18], related to the difficulty in understanding the system due to the level of detail expressed in the model. While simpler, more abstract, models are generally to be preferred, the development of complex system models is often unavoidable – the questions to be answered by running the model sometimes require higher fidelity system representations. Frequently, models are made more complex than they need be, due to uncertainty on the part of modeler about which system aspects to include or leave out – so they err on the side of caution – or the modeler builds in more system features than required, just in case these features are needed in future analysis.

Model complexity, including DoDAF models, has much in common with software complexity, in that both are statically complex in terms of number and convolution of internal element relationships and execution paths.

2.3 System and Subsystem Simulation Complexity

Systems and subsystems exhibit static complexity in correspondence to their modeled representations. During a simulation, system and subsystem entities are exposed to situations which require some sort of behavioral response. The resulting dynamic complexity is a function of the simulation's initial conditions. As noted above, as-fast-as-possible simulations typically model system-level behavior only, while real-time simulations frequently represent subsystems explicitly.

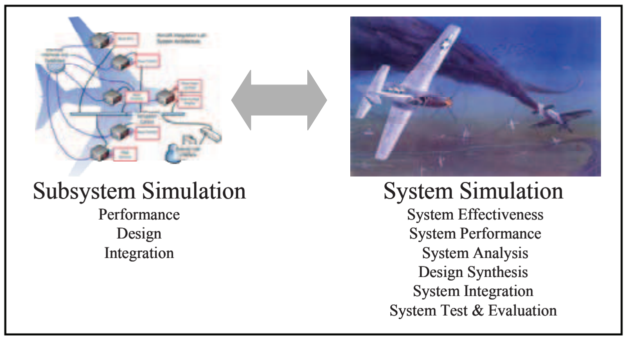


Figure 5: Simulation Complexity

2.4 System of System Simulation Complexity

System of systems models and simulations exhibit the same static and dynamic complexity as already described, but also add *adaptive* or *evolving complexity* to the mix.

The example shown in the Figure 5 illustrates a multi-tiered (nested) system of systems (SoS) architecture, wherein a particular system (e.g., aircraft, surveillance system, etc) is being engineered for operation within the inner, theater warfare tier. Simulation of the inner tier SoS focuses on interoperability of the subject system with other military systems in the SoS at various operational levels – mission, engagement, C^4ISR, basing and logistics, etc. It is common to model inner tier system behavior only to the extent it affects the system under development – the SoS inner tier boundary is assumed to be fixed and includes only those systems directly interacting with the subject system. System adaptation and co-evolution issues are generally ignored or factored out as noise, and only known emergent effects related to subject system mission effectiveness are captured. This level of modeling is encountered in traditional mission analysis and simulation.

However, a robust SoS simulation differs from the system-centric simulations described above. All systems within the SoS boundary (friend and foe) are modeled to roughly the same level of detail, in that the modeled representation of each system includes its mission capability, courses of actions, interaction behavior, and structural and performance fidelity. The SoS boundary is elastic, in that external system influences on the SoS are recognized, evaluated, and answered in a cross-effect give and take.

In Figure 6, the outer SoS tier includes geo-political and public opinion systems (i.e., external human individual and organizational systems) that may impact the inner SoS tier's effectiveness, in terms of funding, schedule, capability, or geographical employment. Whether this level of modeling is important to the engineering of a particular system is an issue for program management; however, strategic consideration of these outer tier impacts remains a factor.

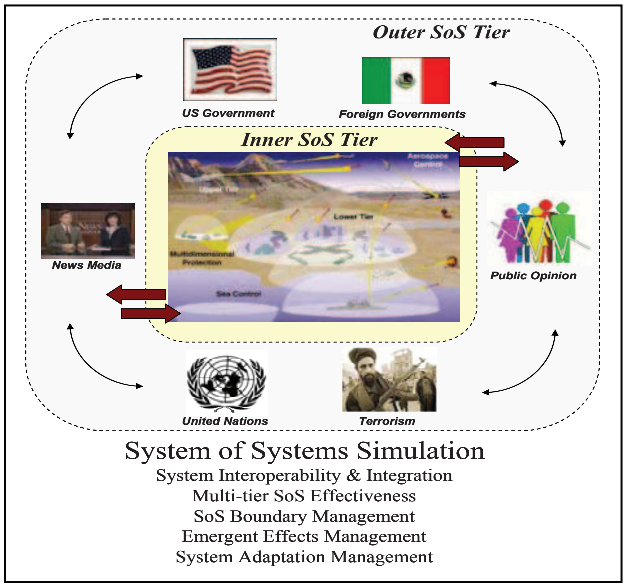


Figure 6: SoS Simulation Complexity

3. System of System Complexity Measurement via M&S

The complexity measures addressed in this section apply to both as-fast-as-possible SoS architecture M&S and real-time SoS integration M&S.

Static complexity addresses the system models – structure, interfaces, potential interactions, network formation (e.g., hierarchical command and control networks, and potential dynamic networks, such as kill chains). Static complexity can be measures via a simulation tool or from architecture descriptions, such as DoDAF.

Dynamic complexity addresses the simulation runtime – what actually happens during a simulation run, based on input scenario and behavior data. Multiple runs are required to adequately measure Monte Carlo (stochastic) model execution (e.g., complexity measure mean and standard deviation are calculated from multiple runs). Measurement of complex adaptive characteristics also requires multiple simulation runs separated by close analysis of opportunities for adaptation, co-evolution, or emergence due to unanticipated effects (Appendix B, Section 5). .

The complexity measures described below are among many measures and metrics uncovered in open literature. Many measures are similar but described in ways peculiar to a particular problem domain or analysis community. In terms of system of system, there is much flexibility in how these measures are applied – to the SoS as a whole, to particular types of systems, to selected networks of interacting systems – and depends on the purpose of the simulation and overarching intended use (e.g., analysis, integration, etc).

All in all, these complexity measures can be fruitfully applied across the SoS modeling and simulation continuum. Complexity is related to the number of agents, the number of relations between those agents, and the time-dependence of agent interactions [a4]. In simulation terms, agents are entities, relations are interfaces and interactions, and time-dependence is a function of the simulated processes and events.

3.1 Algorithmic (Kolmogorov) Complexity

Static complexity at its simplest can be measured structurally as the number of entities within a system of system and how many paths connect the entities. However, a system that merely has many entities with massive connectivity cannot always be classified as complex. For example, a small n-ary tree is more complex than a nearly infinite binary tree, because the binary tree displays a repetitive pattern. This higher complexity is measured using algorithmic information content, which is "the amount of information contained in a string of symbols given by the length of the shortest program that generates the string", as defined by Kolmogorov and Chaitin [20]. This measurement of compression is unfortunately not computable.

3.2 Effective (Gell-Mann) Complexity

Effective complexity is an extension of Kolmogorov complexity. Gell-Mann extends an observed representation into a subjective representation using a model. This model describes a system using probability distributions based on observations through collected data [21]. When the probability distribution is simple, such as in a uniform distribution of equally likely events, then the effective complexity is low. On the other hand, when the probability distribution is complicated with varying degrees of values, then the effective complexity becomes higher.

3.3 Entity Interface Complexity

Ernstoff [11] defines system complexity, from a systems engineering viewpoint, as a function of the number of interfaces between subsystems – "the weighted sum of interface complexities" – where the complexity of each interface is ranked according to the degree of subsystem coupling. Subsystem weighting is varied according to subsystem importance, redundancies, prior use, and other situational factors.

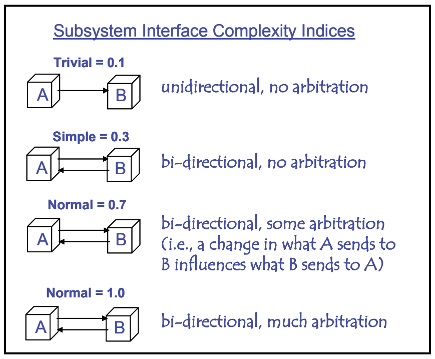

Subsystem Interface Complexity Indices
Trivial = 0.1
A
B
unidirectional, no arbitration
Simple = 0.3
bi-directional, no arbitration
Normal = 0.7
bi-directional, some arbitration (i.e., a change in what A sends to B influences what B sends to A)
Normal = 1.0
bi-directional, much arbitration

Figure 7: Interface Complexity

Ernstoff maintains that overall system complexity generally remains constant as a system is decomposed – adding subsystems increases the number of interfaces offset by simpler components.

3.4 Cyclomatic Complexity

Static complexity can also represent a snapshot of a dynamic system. A measurement that describes this is a software measurement known as cyclomatic complexity, developed by Thomas McCabe. The cyclomatic complexity, $V(G)$, measures the number of paths through a program and is formally defined as:

$$V(G) = e - n + p,$$

where e is the number of edges, n the number of vertices, and p the number of connected components of a flow graph, G [20]. Cyclomatic complexity can measure the number of decisions points of a dynamic system by counting the number of conditional software statements such as IF, CASE, WHILE, and FOR.

3.5 Time-Space Complexity

Time and space (i.e., memory) serve as two measures of effort or resources required for a system of systems to solve a problem. These measures provide relative views of the computational difficulty of a problem. Both are static measures based on algorithmic approaches used within the systems of systems for addressing a problem. Complexity theory differs from computability theory, which deals with whether a problem is solvable at all, regardless of the resources required. After differentiating which problems are solvable and which are not, it was natural to ask about the relative computational difficulty of computable functions. This is the subject matter of computational (time-space) complexity.

Time complexity of a problem is a measure of the number of steps required to solve an instance of that problem as a function of some given input, usually measured in bits. This measure is expressed using "Big-O" notation to give an "order of" approximation to the number of steps. For a simple linear problem, the time complexity would be of order, $O(n)$. For a more complicated problem like a binary search, the time complexity would be of order, $O(n \log n)$. For a Turing machine we could say that the time complexity is of order $O(n^2)$, where n is the length of the input. For a system of systems it is recognized that embedded in the system are feedback loops, switches and flows, which change over time (dynamic complexity of the system) and make the time complexity of the system only indirectly computable. Advantages of using systems approaches as a complementary method for addressing complex problems include the fact that nonlinear relationships, unintended effects of intervening in the system, and time-delayed effects often fail to be addressed with traditional reductionistic approaches, while system of systems approaches excel at detecting these.

The space complexity of a problem is a related concept which measures the amount of space, or memory required by the algorithm to solve the problem. Measurement of Space complexity utilizes the Big-O notation, similar to measurement of time complexity.

3.6 Shannon Entropy

Entropy is the amount of information contained in an object of a system. Like the energy given off from a moving molecule, an object in a system gives information to another object within that system. Every once in a while, information can escape from that system and join another system. Shannon took the concept of entropy and created a measurement that describes the failure of a prediction given the amount of information an observer of the system has. Given x is a discrete variable of $i = 1,2,\ldots,m$ possible values, the equation for entropy is:

$$H(x) = -\sum_{i=1}^{m} p_i(x) \log_2 p_i(x)$$

When a system's outcome can be determined by only 1 value, then $H(x) = 0$, and entropy is at its minimum. Entropy is at the maximum when all m values are taken into account to determine the next outcome [23].

3.7 Effective Measure (Grassberger) Complexity

Effective Measure is closely related to entropy. In a system, effective measure calculates the amount of

$$\mathrm{EMC}(X) = \sum_{n=1}^{\infty} n\delta h_n$$

information needed to generate the probability distribution that determines the next set of signals or possible events. The variable, $h_n = H_{n+1} - H_n$, represents the additional information needed to predict the next event. This value is monotonically decreasing and eventually reaches a saturation point, represented by *EMC* in the following equation:

4. Current Work

Every journey starts with a first step and we are now several steps down this path of exploring system of systems complexity. We have looked at some of the research done over the past several decades and tried several different approaches for measuring system of system complexity. Following a well used mantra to keep things simple, we will most likely need to employ multiple "simple" metrics to account for the richness of the system of system complexity.

The problem seems ripe for a graph theoretic approach, building dependency graphs of the system of system functional decomposition. One of our approaches currently underway is building a UML model of s system of systems, using UML's Object Constraint Language (OCL) to capture the dependency relationships in graph form. This graph form becomes the framework for investigating dimensions of interest such as size, cyclomatic complexity, entropy, and the Hausdorff dimension as a measure of Kolmogorov complexity.

4.1 Hausdorff Dimension

The Hausdorff dimension gives another way to define dimension, which considers the topological metric. To define the Hausdorff dimension for some space X as non-negative real number (that is a number in the half-closed infinite interval $[0, \infty)$), we first consider the number $N(r)$ of balls of radius at most r required to cover X completely. Clearly, as r gets smaller $N(r)$ gets larger. Very roughly, if $N(r)$ grows in the same way as $1/r^d$ as r approaches zero, then we say X has dimension d. In fact the rigorous definition of Hausdorff dimension is somewhat roundabout, since it first defines an entire family of covering measures for X. It turns out that Hausdorff dimension refines the concept of topological dimension and relates it to other properties of the space such as area or volume. [25]

5. Recommendations

This paper is aimed at providing a comprehensive, survey-level introduction to the relationships between systems of systems, complexity, and modeling and simulation. The measures of complexity identified here are only a bare start – it's certain that other (and better) measures can and will be defined as our understanding of complexity evolves in relation to system of systems engineering methodology and problem domains of interest. In addition, potentially fruitful avenues of research include both the incorporation of complexity measurement in other-than-M&S engineering and analysis tools and the application of complexity measures as first-order technical performance parameters in products.

If anything is to be taken away from this paper, it is that (1) modeling and simulation is a robust methodology for comprehensive system of systems analysis and that (2) operational system of systems complexity may be a significant measure in successful system of systems engineering.

6. Acknowledgements

Special thanks to Michael Anderson who has retired and chooses to stay in a relaxed mode while serving as an inspiration.

7. References

[1] Charles, Turner, *Capabilities Based Acquisition – from Theory to Reality*, CHIPS Magazine, Summer 2004.

[2] Wikipedia, System of Systems, http://en.wikipedia.org/wiki/System_of_systems

[3] Stutzke, *Factors Affecting Effort to Integrate and Test a System of Systems*, 20th International COCOMO and Software Cost Modeling Forum, Los Angeles, 25 October 2005.

[4] Gideon, Dagli, Miller, *Taxonomy of Systems-of-*Systems, Conference on Systems Engineering Research, 2005.

[5] Maier, *Architecting Principles for Systems-of-Systems*, http://www.infoed.com/Open/PAPERS/systems.htm

[6] Anderson, Griswold, Aiken, Born, *SoS-101 – Intro to Systems of Systems Thinking*, October 2006.

[7] Meijer, *To Manage or Not To Manage Complexity*, International Engineering Management Conference (IEMC), 1998.

[8] Rechtin, *Systems Architecting of Organizations*, CRC Press, Boca Raton, 2000.

[9] Sheard, *Bridging Systems Engineering and Complex System Sciences*, Third Millennium Systems, May 2006.

[10] Schoening, *Some observations on SoS, complexity, and lack of order*, Internal Boeing Note, 22 May 2007.

[11] Ernstoff, Estimation of System Complexity, Raytheon, May 2001, http://www.ispa-cost.org/c7_pres/

[12] Wikipedia, *Cynefin*, http://en.wikipedia.org/wiki/Cynefin

[13] Krummenoelh, *System of Systems Engineering*, 1st Annual System of Systems Engineering Conference, John Hopkins University, June 2005.

[14] Meilich, *Architecture Challenges in a Net Centric Environment*, NDIA M&S Committee Meeting, February 2007.

[15] Osmundson, Huynh, *A Systems Engineering* Methodology *for Analyzing Systems of Systems*, Naval Postgraduate School, undated.

[16] Carnegie Mellon, Software Engineering Institute, *Halstead Complexity Measures*, http://www.sei.cmu.edu/str/descriptions/halstead_body.html

[17[Carnegie Mellon, Software Engineering Institute, Cyclomatic *Complexity*, http://www.sei.cmu.edu/str/descriptions/cyclomatic_body.html

[18] Chwif, Barretto, Paul, *On Simulation Model* Complexity, Winter Simulation Conference, 2000.

[19] Anderson, *Agent Based Simulation Capability* Integration *into Discrete Event Simulation of Systems of Systems*, IASTED MS 2007 conference, May 2007.

[20] Sporns, Olaf, *Complexity*, http://www.scholarpedia.org/article/Complexity

[21] Gell-Mann, Murray, and Lloyd, Seth, *Information* Measures*, Effective Complexity, and Total Information*, *Complexity*, Vol. 2, Issue 1, March 1996.

[22] McCabe, Thomas J., *A Complexity Measure*, IEEE Transactions on Software Engineering, Vol. SE-2, No. 4, December 1976.

[23] Lemay, Phillipe, *Entropy and Complexity*, http://tecfa.unige.ch/~lemay/thesis/THX-Doctorat/node139.html

[24] Funes, Pablo, *Complexity Measures for Complex* Systems *and Complex Objects*, http://www.cs.brandeis.edu/~pablo/complex.maker.html

[25] Wikipedia. Hausdorff Dimension, http://en.wikipedia.org/wiki/Hausdorff_dimension

8. Biographies

PAUL N. LOWE is a Simulation System Engineer for the Boeing Company. He received his B.S. in mathematics (1979), and his M.S. in Geography (1984) from the University of California, Riverside. Paul has worked in the computer industry for over 31 years, specializing in the areas of high performance computing, simulation, image processing, geographic information systems, database management, and knowledge management.

MICHELLE W. CHEN is a systems engineer in Modeling and Simulation working with SP&A in Phantom Works for the Boeing Company. She has a B.S. and M.S. in Computer and Electrical Engineering, specializing in scientific visualization and virtual reality technology, from the University of California, Irvine. She is also pursuing a second M.S. in Applied Mathematics at Columbia University. She has designed and built several wireless devices for use within augmented and virtual environments. She currently supports analysis, design and software development of simulation environments on SBInet.