A Complexity Primer for Systems Engineers[1]

INCOSE Complex Systems Working Group White Paper

Sarah Sheard, Stephen Cook, Eric Honour, Duane Hybertson, Joseph Krupa, Jimmie McEver, Dorothy McKinney, Paul Ondrus, Alex Ryan, Robert Scheurer, Janet Singer, Joshua Sparber, and Brian White

July 2015

I. Introduction

Complexity is nothing new to systems engineers and managers. The discipline of systems engineering evolved to improve our ability to deal with scale, interdependency, and complexity in systems development. Few systems engineers would doubt that complexity is increasing every year. The rate of change, the increasing interdependence and adaptability of systems, and the increasing ambitions of our clients ensure that complexity keeps expanding to the limits of our capacity to cope with it.

Complex systems science provides a strong foundation for understanding, coping with, and even exploiting complexity. There is a large and rapidly expanding literature on networks, complexity, and complex adaptive systems that can guide systems engineering practice. But busy systems engineers rarely have the time to keep up with this literature, which is diffused across the many interdisciplinary applications of complex systems science. This paper is written for systems engineers and program/project managers who suspect they may encounter complexity-related challenges. This paper applies key concepts from complex systems science to systems engineering to suggest new methods that can handle complexity rather than assuming it away. It is not a complete or even extensive treatment, but is intended as an introduction to the subject for systems engineers as they encounter and work with complexity-related phenomena.

Section II of this paper defines complexity and describes how we can identify complexity in an environment, a problem space, or a solution space. We also address the extent and types of complexity that a system or situation may exhibit, so that systems engineers can seek approaches that can better address that kind of complexity.

Section III then discusses how engineers address the problem of complexity, in two sections. The first approach requires thinking differently about the environment, the problem, and the solution. The second approach involves implementation of specific tools and techniques.

II. What is Complexity?

In ordinary language, we often call something complex when we can't fully understand its structure or behavior: it is uncertain, unpredictable, complicated, or just plain difficult. Sillitto (2009) described the inability of a human mind to grasp the whole of a complex problem and predict the outcome as s*ubjective*

[1] This primer was created from more extensive material, including draft papers previously written by this working group, in order to provide a reasonably short accounting of what systems engineers (and their managers) need to know about complexity. Comments and other feedback can be submitted to the current Complex Systems Working Group chair, Dr. Jimmie McEver, at jimmie.mcever@jhuapl.edu.

complexity. John Warfield's (2006) "frustration" in the mind of the system's builders and users also fits into this bucket. While there are ways to reduce this complexity and improve the fit of technical systems into the complex environment, they are not the focus of this primer.

Sillitto's Objective Complexity describes technical or system characteristics that lead to the subjective complexity or difficulty. As systems engineers, we have the ability to modify these characteristics; they are also the ones most frequently addressed by complex systems science.

The standard systems engineering process breaks down a problem into parts, recursively, until the parts are simple enough that we understand them and can design solutions; we then re-assemble the parts to form the whole solution. The approach works well for systems whose parts interact in fixed ways (also known as "complicated" systems – an example might be a car), even when there are many interacting parts and the systems may have unpredictable behavior.

Other systems, however, present significant problems when analyzed in piecewise fashion. Systems such as transportation networks have autonomous parts whose interactions lead to emergent self-organized patterns of behavior. In these systems, defined here as "complex" systems, the emergent properties that we really care about are not understandable form the perspective of the parts in isolation. It is especially for these systems that we are providing guidance to additional tools and techniques created in complex systems sciences and recommend their use in systems engineering of complex systems.

A. Characteristics of complexity

Complexity is a characteristic of more than just a technical system being developed. It is often created by the interaction of people, organizations, and the environment that are part of the complex system surrounding the technical system. Complexity results from the diversity, connectivity, interactivity, and adaptivity of a system and its environment. Constant change makes it difficult to define stable goals for a project or system. Technical systems that worked well in the past to solve an environmental problem become obsolete quickly. Intricate networks of evolving cause-effect relationships lead to subtle bugs and surprising dynamics. Unintended consequences can overwhelm or even negate the intended consequences of actions.

When systems are complex, their structures cannot be described at a single level or with a single view; multi-scale descriptions are needed to understand complex systems. Their emergent behavior, derived from the relationships among their elements and with the environment, via internal and external feedback loops, gives rise to observed patterns that may not be understood or predicted. Describing the behavior of a system as a response function may require an unobtainable amount of information. It is often impossible to predict future configurations, structures, or behaviors of a complex system, given finite resources.

A complex system may have multiple stable states (meaning each state is actually meta-stable), transient states, or even no lasting stable states, exhibiting continuous evolution. Perturbations in the system may result in recovery to the former state but may also lead to transitions to another state and consequent radical changes of properties. In addition, details seen at the fine scales can influence large-scale behavior. Dynamics of different parts and patterns cannot be reproduced using simple averages. Complex systems are perpetually generating novelty, many key variables are opaque, boundaries are indeterminate, and weak ties can have a disproportional effect on system behavior. Duality is common:

tension between large and small, distributed and central, agile and planned calls for perpetual seeking of balance. In short, complex systems are different..

Although complexity can present challenges, complexity is often inherent and may even be a necessary or desirable attribute of a solution system. Systems that have been engineered to rule out any but deterministic behaviors are necessarily limited by the prescribed behaviors, and do not extend well into unplanned environments. In contrast, complex systems can be engineered to have sufficient adaptability to operate well in a changing environment, responding to change in appropriate and effective ways. A complex system provides a welcome kind of variety that can help provide control of different dimensions and enables the system to adapt to environmental change. To provide new capabilities or graceful degradation, a complex system can adapt by re-organizing its structure, responses, or patterns of parts.

B. Identifying the Right Level of Complexity

Science and engineering help humans to make successful systems because they provide ways to understand, predict, and control technology, in order to create desirable effects on the world. All science involves abstraction of the complexity of the world into approaches and models that use simplifying assumptions. This allows us to generalize from one complex situation to another. The best engineering methods take advantage of the simplicity in the models without diverging so far from reality that behavior can no longer be predicted and controlled.

As a system's diversity, connectivity, interactivity, or adaptivity increases, the risk associated with using simpler methods and simplifying assumptions also increases, and more advanced techniques may be needed. Tools and techniques apply differently to systems on a spectrum of increasing complexity. At the less complex end, the waterfall model for top-down sequential design applies well. At the more complex end, tools such as agent-based models for model-based systems engineering can be used to understand and address complex, dynamic systems design challenges. Techniques at the lower end of the spectrum tend to be easier to learn, and simpler and faster to apply, because they make simplifying assumptions that ignore some of the complexity. The practitioner must apply judgment to utilize a mixture of tools along this spectrum that satisfies Einstein's razor: make things as simple as possible, but not simpler.

It is rarely possible to fully assess in advance what complexity within a project must be addressed and what can be assumed away. Because complex systems perpetually generate novelty, systems engineers will often have to adapt their approach to unfolding conditions and use flexible tools. Systems engineers should also recognize that the complexity of a system, as manifested in many diverse types of parts and relationships, provides a welcome kind of variety that can help facilitate control of different dimensions and enables the system to adapt to environmental change.

III. Solutions for Complexity

Throughout its history, systems engineering has been the primary method for engineering in the face of complexity. As the complexity of systems and their contexts has grown, systems engineering methods and tools have increasingly fallen short of what is needed in the face of this reality. A common approach has been to seek clever ways to simplify, or reduce, the subjective complexity so that the problem and the system are understandable. Scientific advances have, in fact, often come from elegant simplifications that model the important variables or forces that dominate behavior. However, this is not always possible – complexity often cannot be simplified away without losing the essence of the problem or possible solutions. Further, this simplification leads to an inability on the part of the solution to be able to engage with the complexity that remains despite our preference to assume it away.

Ross Ashby's Law of Requisite Variety shows that a system controller must have at least as many degrees of control as the degrees of freedom in the environment to be controlled. If a system operates within an environment of human processes, as in today's air traffic control, then the system solution must have sufficient complexity to do so. In such a system, it is difficult and even dangerous to ignore the complexity.

Therefore it behooves systems engineers to acknowledge, understand and learn to work better with complexity. A first step is to identify the kinds of complexity in a system and its environment (Section II). A second step is to create appropriate new ways to think about complexity that guide the approaches used (Section IIIA). A third step is to evolve and publicize methods to deal with different types of complexity in different situations (Section IIIB).

Working with complexity will never be a trivial task that can be reduced to following a checklist. The complex, adaptive neural network of our human brains will always be needed to supplement documented lessons of any sort. A long-term goal of the INCOSE Complex Systems Working Group is to facilitate the systems engineering community's collective learning with non-trivial lessons, heuristics, and shared stories, all of which are outside the scope of this Primer. We acknowledge that the world is complex, and that our orderly, simple views are inadequate. We must grow toward a new appreciation of the implications of complexity – in our systems, in their ecosystems (both technical and socio-political), and in the interplay between the two – in order to be successful.

A. Complexity Thinking: Guiding Principles

Complex systems engineering requires both a shift in thinking and an expanded set of tools and techniques. In this section, we summarize the shift in thinking needed to acknowledge and embrace complexity within systems engineering. Several principles are provided below that encourage systems engineers to think differently about how to engage with complexity.

1. **Think like a gardener, not a watchmaker.** Consider the complexity of the environment and the solution, and think about evolving a living solution to the problem rather than constructing a system from scratch.
2. **Combine courage with humility.** It takes courage to acknowledge complexity, relinquish control, encourage variety, and explore unmapped territory. It takes humility to accept irreducible uncertainty, to be skeptical of existing knowledge, and to be open to learning from failure. A combination of courage and humility enables the complex systems engineer to risk genuine innovation and learn fast from iterative prototyping of solutions in context.
3. **Take an adaptive stance.** Systems engineers should mimic how living systems cope with complexity by identifying and creating variation, selecting the best versions, and amplify the fit of the selected versions. This means, for example, to think "influence" and "intervention" rather than "control" and "design." Designing or evolving a complex system requires recognition that the designer may not ever be able to control or even understand the system completely.

4. **Use free order.** In architecting and designing solutions, build in "order for free" using self-organization, presuming it has been modeled and can be limited to desired effects. This in particular applies when the system being designed must be resilient.
5. **Identify and use patterns.** Patterns are exhibited by complex systems, can be observed and understood, and are a key mechanism in the engineering of complex systems. Patterns are the primary means of dealing specifically with emergence and side effects—that is, the means of inducing desired emergence and side effects, and the means of avoiding undesired emergence and side effects.
6. **Zoom in and zoom out.** Because complex systems cannot be understood at a single scale of analysis, systems engineers must develop the habit of looking at their project at many different scales, by iteratively zooming in and zooming out. Can problems be solved more elegantly by addressing them at a higher or lower hierarchical level? The complex systems engineer must be especially open to solutions that arise from the bottom-up through self-organization, rather than only seeking to impose order from the top-down.
7. **See through new eyes.** A complex situation often looks very different from the perspectives of the variety of stakeholders. By empathizing with these multiple perspectives, a systems engineer can sometimes find creative ways to solve several problems at once.
8. **Collaborate.** Collaboration includes information sharing, active listening, establishment of trust to enable candid dialogue, and making decisions transparent. A collaborative mindset can lead to deeper stakeholder engagement practices that may include crowdfunding and crowdsourcing, to enable co-creation and co-evolutionary systems design.
9. **Achieve Balance.** Optimization is often counterproductive within a complex system. Either the whole is sub-optimized when a part is optimized, or an optimized whole becomes rigid, unable to flex with changing conditions. Instead of optimizing, complex systems engineers should seek balance among competing tensions within the project. Systems engineers can leverage integrative thinking to generate improved solutions and avoid binary either/or tradeoffs. The goal is a system that would continue to meet the need even if a number of current conditions change.
10. **Learn from problems.** In a changing context, with an evolving system, where elements are densely interconnected, problems and opportunities will continually emerge. Moreover, they will emerge in surprising ways, due to phase transitions, cascading failures, fat tailed distributions, and "black swan" (Taleb, 2007) events. A traditional approach to risk management and mitigation should be augmented by a complexity mindset that balances risk management with exploiting opportunity and expects and learns from error.
11. **Meta-cognition**. Meta-cognition, or reflecting on how one reflects, helps to identify bias, make useful patterns of thinking more frequent, and improve understanding of a complex situation.
12. **Focus on desired regions of outcome space rather than specifying detailed outcomes.** Instead of zeroing in on an exact solution, focus on what range of solutions will have the desired effects, and design to keep out of forbidden ranges.
13. **Understand what motivates autonomous agents.** Changing rewards will shape collective behavior. Implement incentives that will move the system toward a more desired state.
14. **Maintain adaptive feedback loops.** Adaptive systems correct for output variations via a feedback mechanism. Over time, feedback loops can either hit the limit of their control space, or may be removed in the interest of maintaining stability. To maintain robustness, periodically revisit feedback and ensure that adaptation can still occur.

15. **Integrate problems**. Focus on the relationships among problems rather than addressing each problem separately. This allows fewer solutions that take care of multiple problems in an integrative fashion.

B. Specific Methods

The principles and approaches described above are important points of departure for systems engineers facing complexity in their SE activities. It is also useful to describe how these concepts apply to particular aspects of the systems engineer's work – aspects that are fundamental regardless of the nature of the system being developed or the problem being addressed. Systems engineers' toolkits should include a wide range of methods and processes to address environmental and system complexity in appropriate and useful ways.

A key first step is one of diagnosis – the systems engineer must identify the kind and extent of complexity that bears on the problem set. As we have seen, complexity can exist in the problem being addressed, in its environment or context, or in the system under consideration for providing a solution to the problem. The diagnoses made will allow the systems engineer to tailor his/her approaches to key aspects of the systems engineering process: requirements elicitation, trade studies, the selection of a development process life cycle, solution architecting, system decomposition and subsystem integration, test and evaluation activities, and others.

In addition, the diagnosis will allow the systems engineer to consider whether there may be mechanisms for shifting complexity to a more desirable region of the problem space. There may be choices available or investments that can be made to allow the decoupling of aspects of the system or of the system to its environment. Likewise, there may also be options for shaping the feedbacks within and across problem-environment-solution elements, allowing the complexity of the situation to be harnessed via the leveraging of beneficial adaptation and self-organization.

The tables below show examples of the kinds of choices that can improve the success of engineering complex systems when complexity of different types and from different sources exists. These lists are by no means exhaustive; indeed, astute readers will note that some entries in the tables below are blank. These tables are intended to convey the types of approaches and departures from traditional SE methods that may be required when dealing with complexity in problem, solution, and environment contexts. The authors expect that this table will be augmented and evolved in future updates to this primer.

These approaches are offered to help systems engineers ensure that their processes are appropriate for dealing with the dynamics, uncertainty and behaviors that can arise when significant complexity exists in the problem, the solution, or the environment. Experience suggests that when SE activities do not account for these factors when they exist, projects fail and problems go unaddressed. Our goal in this primer is to provide systems engineers with techniques for better recognizing complexity and its consequences as it pertains to their activities, and for expanding the envelope of the degree and types of complexity that can be dealt with. Without doubt, there is additional work to do to expand our understanding of these phenomena – and to provide practical tools to allow the systems engineering community to leverage and better apply emerging insights. The concepts and approaches in this section, however, should provide a useful starting point.

Table 1. Candidate approaches to address complexity in problem context or environment.

	Requirements Elicitation and Derivation	Trade Studies	Solution Architecture and Design	Development Process
Complexity in the environment - General	Use multiple methods for requirements elicitation Elicit requirements from multiple perspectives and at multiple levels of aggregation Emphasize capture of system objectives and desired outcomes rather than thousands of detailed requirements.	Emphasize robustness over local efficiency and performance	Include both positive and negative feedback mechanisms to provide mechanisms to compensate for the effects of higher-than-linear positive feedback and runaway system behavior	Employ soft systems methodologies to surface the nature of the problem space, its internal structure and information flows, and produce simple representations, eg 'rich pictures' to communicate these.
Intricate and evolving/self-organizing interactions with the environment	Include requirements for the system to provide adaptive local control, rather than strong, deterministic control	Trade end-to-end system perfor-mance and behavior against problem space complexity. Think hierarchy rather than flat networks.	Early implantation (or at least prototyping) of external interfaces	Early deployment of system functionality with feedback to developers
Environment susceptible to "black swan" events (unlikely, unpredictable, high-consequence events) and/or recursive complexity	Use power laws rather than Gaussian distributions to characterize phenomena in requirements and sell-off criteria Focus requirements elicitation on resiliency, robustness and adaptiveness vice optimizing to particular assumptions	Make resilience a key trade-space attribute and use trades to identify aspects of the problem space that will drive the system architecture	Design for resilience to "beyond-design-envelope" events to provide robustness and timely recovery to a minimally functional state.	Resilience analysis. Enterprise develop-ment: study how enterprises or societies survive catastrophes.
Complexity in the problem/mission	Emphasize identification of constraints as well as requirements Capture scenarios and mission threads in preference to large numbers of requirements	Use scalability and agility as criteria in appropriate trade studies	Use solution elements which are adaptable and/or reconfigurable. Design to achieve scenarios rather than detailed requirements. Satisfice at the system level rather than satisfy detailed requirements.	Use Agile, evolution-ary SE processes instead of Waterfall SE processes. Define multi-layer processes and their interface points.
Complexity in stakeholder relationships	Use multiple scales (or a Balanced Scorecard approach) instead of a single utility function to determine "goodness" or fitness for use	Seek stakeholder buy-in to trade studies	Use modeling and simulation to enable stakeholders to experience (rather than just be briefed about) interactions of solution elements and the environment	Employing a multi-methodological approach, i.e., soft systems methodol-ogies plus SE plus boundary critique, identify stakeholders and achieve buy-in.
Complexity in interactions between different mission elements	Capture scenarios and mission threads in preference to large numbers of detailed requirements. Develop understanding and means of controlling non-linearities, disruptive events.	Make minimizing interactions/interd ependences between complex systems elements a key trade-off attribute. Model with system dynamics.	Use exploratory modeling and simulation to assess ability of solution elements to address mission elements under wide range of conditions and assumptions. Use system dynamics to link economics and system changes.	Establish and main-tain an interdepend-dency database between constituent systems that is used for all major design decisions

Table 2. Candidate approaches to address system/solution complexity.

	Requirements Elicitation and Derivation	Trade Studies	Solution Architecture and Design	Development Process
Complexity in System Design & Development (General)	Use multi-scale modeling (linking macro- and micro-level models), including exploratory analysis and agent-based modeling, and experimentation: • To generate insight into the implications of derived requirements • As the basis for trade studies and to inform trade-off decisions		Emphasize selection of robust and adaptive elements and structures over optimizing to meet current requirements	Use SoSE methodologies to synchronize constituent systems; incentivize collaboration. Ensure prototyping and experimentation are used.
Emergent Properties or Behaviors in Solution System	Maximize description of emergent properties in scenarios and mission definition. Elicit requirements from multiple perspectives and at multiple levels of aggregation; ensure requirements and constraints at all relevant levels are understood Understand the real value of predictability at different levels; encourage the definition of requirements at higher "essential value" levels	Employ modeling and experimentation to ensure relevant effects of trades are explored at different levels of aggregation	Build in feedback mechanisms to enable the system and system elements to adapt to the environment and each other in effect ways. Acknowledge the limits to the value of decomposition-based methods; emergence is a collective phenomenon that requires aggregation – emergence will not be observed until the system is considered as a whole	Conduct development activities always within context of the whole Employ collaborative development processes so that information about design decisions are visible throughout the project Prototyping and holistic testing are critical to explore and check for the manifestation of emergent behavior
Complexity in System Deployment & Operation	Employ soft systems methodologies to surface the nature of the deployed solution, and its internal structure and information flows; produce simple representations, e.g., 'rich pictures' to communicate these. Use problem definition methods from an evolutionary SE or SoSE methodology.	Trade criteria need to value cost and ease of training and logistical support over acquisition cost. Model system evolution with genetic algorithms.	Use self-organizing and self-repairing elements when possible. Model the cost of change, the benefits, and the balance.	Employ soft systems methodologies to surface issues, engage stakeholders, identify approaches to improve the deployed system, and to achieve stakeholder buy-in to the solution. Use an evolutionary SE or SoSE methodology. Identify utility and cost of using and modifying legacy systems.
Complexity in System Evolution & Support	Focus on capabilities, not requirements.	Use resilience and robustness as criteria in trade studies	Ensure that the most important system elements are composable (that is, capable of being re-configured adaptively with other elements in the future to satisfy emergent operational needs not previously envisioned) and have clear and accessible interfaces.	Emphasize understanding the problem, and at each stage of this iterative process, intervene to add an incremental capability and watch carefully over time to see whether there is improvement. If not, try something else.

In addition to new approaches to SE activities, different approaches to systems modeling and simulations are also needed to deal with complexity. INCOSE Complex Systems Working Group member Stephen Cook suggested development of a matrix showing modeling methods, their benefits, and conditions for which each method is and is not appropriate. A subset of applicable modeling methods have been extracted from this matrix and organized into categories relevant to systems engineering: Analyze, Diagnose, Model and Synthesize. Additionally, Shalizi's (2006) review of complexity tools suggested new tools not in the original Cook Matrix. The tools in each column generally range from the simplest at the top to the most complex at the bottom. The resulting matrix, shown in Table 3, provides a starting point for systems engineers seeking complexity-appropriate modeling approaches to systems engineering.

Table 3. Selected modeling methods for complex systems from the Cook Matrix.

Analyze	**Diagnose**	**Model**	**Synthesize**
Data Mining	Algorithmic Complexity	Uncertainty Modeling	Design Structure Matrix
Splines	Monte Carlo Methods	Virtual Immersive Modeling	Architectural Frameworks
Fuzzy Logic	Thermodynamic Depth	Functional / Behavioral Models	Simulated Annealing
Neural Networks	Fractal Dimension	Feedback Control Models	Artificial Immune System
Classification & Regression Trees	Information Theory	Dissipative Systems	Particle Swarm Optimization
Kernel Machines	Statistical Complexity	Game Theory	Genetic Algorithms
Nonlinear Time Series Analysis	Graph Theory	Cellular Automata	Multi-Agent Systems
Markov Chains	Functional Information	System Dynamics	Adaptive Networks
Power Law Statistics	Multi-scale Complexity	Dynamical Systems	
Social Network Analysis		Network Models	
		Agent Based Models	
		Multi-Scale Models	

IV. Summary

Although the many different meanings of complexity vary, from "confusion" to measurable characteristics of technical systems, it is most useful to systems engineers to identify characteristics that can be resolved and whose resolution will improve the development and operation of modern systems. This primer has attempted to describe complexity in such a usable and useful manner. Complexity is an attribute of the technical system being developed but also of the problem space (including people and organizations), and the environment. Complexity is associated with size, diversity, dynamism and with emergence. It is a challenge to systems engineers not to over-simplify in pursuit of representations and capabilities that can be understood and controlled; the right level of complexity is key.

We have shown principles to guide the extension of systems engineering practices to handle ever-more-complex situations and systems. We have also catalogued a number of systems engineering approaches based on concepts from complex systems science. These are shown on a matrix of type of complexity and type of systems engineering practice (which may occur across any specific life cycle). We also list modeling methods that may be of use and are under consideration for longer explanatory articles.

Recommended Reading

Bar-Yam, Y., Making Things Work: Solving Complex Problems in a Complex World, New England Complex Systems Institute, 2005.

Boccara, N., *Modeling Complex Systems*. New York: Springer-Verlag, 2004.

Checkland, P. B. and Scholes, *Soft Systems Methodology in Action,* Wiley & Sons: Chichester, 1990.

"Principles of Systems Thinking." in A. Pyster and D.H. Olwell (eds). 2013. *The Guide to the Systems Engineering Body of Knowledge (SEBoK), v. 1.1.1*. Hoboken, NJ: The Trustees of the Stevens Institute of Technology. Accessed 16 June 2013. www.sebokwiki.org/

Hybertson, D., Model-Oriented Systems Engineering Science: A Unifying Framework for Traditional and Complex Systems, Auerbach/CRC Press, Boca Raton, FL, 2009.

Shalizi, C. R., "Methods and techniques of complex systems science: An overview", in *Complex Systems Science in Biomedicine*, pp. 33-114, Springer US, 2006.

Sillitto, H. G., "On Systems architects and systems architecting: some thoughts on explaining the art and science of system architecting," *Proc. of INCOSE IS 2009*, Singapore, 20-23 July 2009.

Taleb, N. N., *Antifragile – Things that Gain from Disorder*, New York: Random House, 2012.

Taleb, N. N., The Black Swan – Impact of the Highly Improbable, New York: Random House, 2007.

Warfield, J. N. 2006, *An Introduction to Systems Science,* World Scientific Publishing Co. Pte. Ltd., Singapore, 2006.