Education in Complex Systems for Systems Engineers

June 2012

Gregory A. Miller
Naval Postgraduate School
Systems Engineering Department
777 Dyer Rd.
Monterey, CA 93943

Abstract

This paper explores the potential benefits and disadvantages associated with providing a course (or part of a course) in complex systems as part of a systems engineering graduate degree program. An overview of systems engineering education programs and a history of SE curricular design provides context for this exploration. Curriculums other than systems engineering that include courses in complexity, nonlinear dynamics, emergence, chaos, decentralized synchronization through stigmergy, scale-free networks and related topics are also examined. The application of these concepts to design problems is assessed, particularly for engineers in the defense domain. Complex systems learning objectives tailored for systems engineers are proposed. Finally, a framework on which to base a trade-off analysis to determine if such topics should be included in an existing or developing graduate degree program is recommended.

Introduction & Background

The need for SE education and training has long been an issue for defense acquisition, "The quantity and quality of systems engineering expertise is insufficient to meet the demands of the government and the defense industry" [NDIA, 2006]. Several studies that have examined this technical workforce problem from the perspective of human capital management in the shipbuilding industry [Todd & Parten, 2008], US Army acquisition [Clayton, et al., 2011], and NASA [Menrad & Larson, 2008] indicate that a holistic approach addressing all aspects of recruiting, developing, retaining and managing intellectual resources is necessary to provide SE education and training. This paper examines the role of complexity theory as part of a graduate program in providing an integrated solution. An overview of existing systems engineering programs and curriculum structures provides a context for this study.

There is little doubt that the design, development and deployment of complex systems have become a significant part of systems engineers' tasking and will continue to be [Calvano & John, 2004]. However, it is not really a new trend on its own. Rather, it is recognition that all engineered systems, as part of larger socio-technical systems, exhibit behavior dominated by lateral influences [Calvano & John, 2004]. Indeed, in a 2006 study of the top systems engineering issue within DoD, the National Defense Industrial Association indicated that SE skills should support increasing complexity and that future issues would revolve around systems of systems and complexity [NDIA, 2006]. DoD's own Systems Engineering Guide for Systems of Systems recommends applying "complexity theory to problems of large-scale, heterogeneous" systems [OUSD AT&L, 2008]. It defines emergence as a key property of systems of systems and encourages the use of models to understand complex behavior. Some authors [Minai, et al, 2010; Bar-Yam, 2010; Norman & Kuras, 2010] go so far as to argue that the traditional practices of systems engineering are of limited usefulness. Only by applying the concepts of complexity itself will engineers be successful.

It is assumed the reader is familiar with the definitions of complex systems and related topics, so a discussion reviewing those definitions, their origins and recent cross-disciplinary efforts is not warranted here. Precise definitions and examples for the systems engineering community are provided by Sheard and Mostashari [2009], Sheard [2005], Minai, et al [2010] and Beckerman [2000]. A more rigorous and expansive discussion is presented by Maier and Fadel [2010].

Generic Systems Engineering Curricula

A curriculum framework for SE was first described by Squires [2007] based on an analysis of over 200 courses related to systems engineering. The framework includes 16 topic areas which are further grouped into one of four categories. Further work which included more universities, some in Europe, which connected this framework with a set of systems engineering competencies, resulted in a proposed reference curriculum [Jain, et al, 2007]. Using Quality Function Deployment (QFD) -inspired techniques, a correlation was established between course topics and industry-required competencies. The result was a set of recommendations and a two-dimensional framework for graduate-level courses which is repeated as Figure 1 [Jain, et al, 2007].

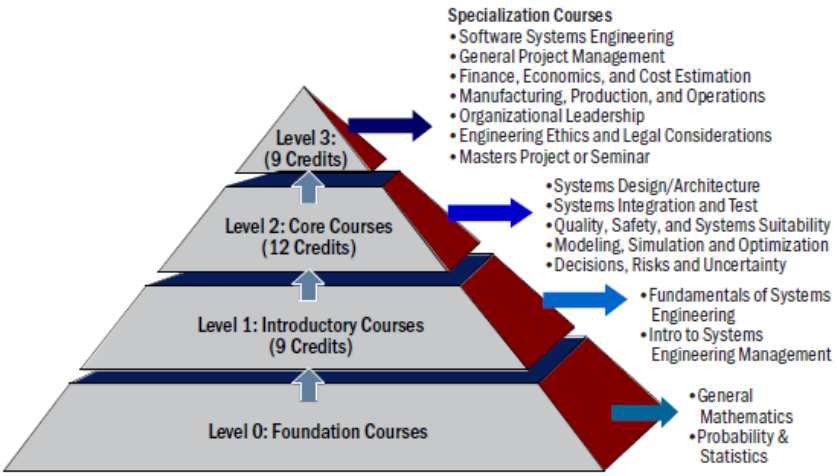

Figure 1. Proposed Reference Framework for a Systems Engineering Curriculum [Jain, et al, 2007]

The final evolution of the framework included extensions based on ISO/IEC 15288 [ISO/IEC, 2008] and INCOSE's Systems Engineering Handbook v3.2 [INCOSE, 2010]. Table 1 outlines the new framework [Squires & Cloutier, 2010].

Table 1. SE Curriculum Framework Course Categories and Types [Squires & Cloutier, 2010]

Level	Category	Course Type
0	Pre-requisite Courses	Probability & Statistics
0		Linear, Matrix, Differential Equations
1	Fundamentals: Generic or Domain Specific	Fundamentals of Systems Engineering
1		Fundamentals of Software Systems Engineering
1		Introduction to Systems Engineering Management
1		Introduction to Domain Specific
2	System Life Cycle Technical Processes	Mission Needs, Systems Concept, System Requirements, Requirements Analysis
2		Systems Architecture, Systems Design and Development
2		Modeling, Simulation and Optimization
2		System Integration and Test, Field Testing
2		Manufacturing, Production, Operations, Retirement
2		Systems Suitability: Quality, Safety, Reliability, Supportability
3	System Life Cycle Project Processes	Decisions, Risks and Uncertainty
3		Configuration Management, Information Management
3		Project Management, Finance, Economics, Accounting
4	Other System Life Cycle Processes	Enterprise Systems
4		Acquisition and Supply
4	Other Broad Areas Applicable to Systems Engineering	Systems Thinking
4		Creativity and Problem Solving
4		Subject Matter Expert Domain Specific
5	Capstone	Masters Thesis, Project or Seminar

The development of a more complete framework that encompasses systems engineering education is currently being pursued by collaborating researchers within the Systems Engineering Research Center (SERC), formed under a DoD University Affiliated Research Consortium (UARC) and led by the Stevens Institute of Technology, as a Body of Knowledge and Curriculum to Advance Systems Engineering (BKCASE) and a Graduate Reference Curriculum for Systems Engineering (GRCSE - pronounced "Gracie"). This is supported by the professional societies of the International Council on Systems Engineering (INCOSE) and the Institute of Electrical, and Electronic Engineers (IEEE) through their Systems Council, as well as the National Defense Industries Association (NDIA) Systems Engineering Division [BKCASE, 2009; Squires, et al, 2011; Ferris, et al, 2010; Chittister and Haimes, 2011].

Guidance that informs SE Education Development

OSD, in conjunction with DAU, INCOSE, NDIA, several academic institutions, and Federally Funded Research and Development Centers (FFRDC) defined a series of systems engineering competencies, and a subset of these competencies for the Systems Planning, Research, Development and Engineering (SPRDE) career field. The SPRDE model is defined through 29 competencies with 45 elements. An excerpt showing the first ten elements is provided in Table 2.

Table 2. SPRDE-SE/PSE Systems Engineering Competency Model.

Competency #	Name	Element #	Definition
1	Technical Basis for Cost	Element 1	Provide technical basis for comprehensive cost estimates and program budgets that reflect program phase requirements and best practices using knowledge of cost drivers, risk factors, and historical documentation (e.g. hardware, operational software, lab/support software).
2	Modeling and Simulation	Element 2	Develop, use, and/or interpret modeling or simulation in support of systems acquisition.
3	Safety Assurance	Element 3	Review Safety Assurance artifacts to determine if the necessary SE design goals and requirements were met for: Safe For Intended Use (SFIU), warfighter survivability, user safety, software safety, environmental safety, Programmatic Environmental, Safety and Health Evaluations (PESHE), and/or Critical Safety applications.
4	Stakeholder Requirements Definition	Element 4	Work with the user to establish and refine operational needs, attributes, performance parameters, and constraints that flow from the stakeholder described capabilities, and ensure all relevant requirements and design considerations are addressed.
5	Requirements Analysis	Element 5	Ensure the requirements derived from the stakeholder-designated capabilities are analyzed, decomposed, functionally detailed across the entire system, feasible and effective.
6	Architecture Design	Element 6	Translate requirements into alternative design solutions. The alternative design solutions include hardware, software, and human elements; their enabling processes; and related internal and external interfaces.

6	Architecture Design	Element 7	Track and manage design considerations (boundaries, interfaces, standards, available production process capabilities, performance and behavior characteristics) to ensure they are properly addressed in the technical baselines.
6	Architecture Design	Element 8	Generate a final physical architecture based on reviews of alternative designs.
6	Architecture Design	Element 9	Conduct walkthroughs with stakeholders to ensure that requirements will be met and will deliver planned systems outcomes under all combinations of design usage environments throughout the operational life of a system.
7	Implementation	Element 10	Manage the design requirements and plan for corrective action for any discovered hardware, software, and human deficiencies

ABET accredits academic programs in engineering, including systems engineering. Two institutions currently have Masters of Science in Systems Engineering (MSSE) programs accredited by ABET: NPS and the Air Force Institute of Technology (AFIT). The ABET EC2010 Criterion 3 on "Program Outcomes and Assessment" lists the common aspects that engineering programs must demonstrate for their graduates, frequently referred to as the ABET (a) – (k) criteria, Table 3.

Table 3. ABET EC2010 Criterion 3.

(a) an ability to apply knowledge of mathematics, science, and engineering
(b) an ability to design and conduct experiments, as well as to analyze and interpret data
(c) an ability to design a system, component, or process to meet desired needs within realistic constraints, such as economic, environmental, social, political, ethical, health and safety, manufacturability, and sustainability
(d) an ability to function on multidisciplinary teams
(e) an ability to identify, formulate, and solve engineering problems
(f) an understanding of professional and ethical responsibility
(g) an ability to communicate effectively
(h) the broad education necessary to understand the impact of engineering solutions in a global, economic, societal and environmental context
(i) a recognition of the need for, and an ability to engage in, life-long learning
(j) a knowledge of contemporary issues
(k) an ability to use the techniques, skills, and modern engineering tools necessary for engineering practice

These outcomes must be addressed within systems engineering programs in order to achieve and maintain ABET accreditation. They provide useful high-level outcomes for systems engineering education. The NPS SE programs address these outcomes via a decomposed structure of learning objectives, which form the basis for instruction and assessment of student mastery of these aspects.

The Conceive-Design-Implement-Operate (CDIO) Initiative has developed a syllabus to use as a context for engineering education, Table 4 [Crawley et al, 2011].

Table 4. CDIO Syllabus v2.0 at Second Level of Detail

1 DISCIPLINARY KNOWLEDGE AND REASONING	**3 INTERPERSONAL SKILLS: TEAMWORK AND COMMUNICATION**
1.1 KNOWLEDGE OF UNDERLYING MATHEMATICS AND SCIENCE	3.1 TEAMWORK
1.2 CORE FUNDAMENTAL KNOWLEDGE OF ENGINEERING	3.2 COMMUNICATIONS
1.3 ADVANCED ENGINEERING FUNDAMENTAL KNOWLEDGE, METHODS AND TOOLS	3.3 COMMUNICATIONS IN FOREIGN LANGUAGES
2 PERSONAL AND PROFESSIONAL SKILLS AND ATTRIBUTES	**4 CONCEIVING, DESIGNING, IMPLEMENTING, AND OPERATING SYSTEMS IN THE ENTERPRISE, SOCIETAL AND ENVIRONMENTAL CONTEXT**
2.1 ANALYTICAL REASONING AND PROBLEM SOLVING	4.1 EXTERNAL, SOCIETAL AND ENVIRONMENTAL CONTEXT
2.2 EXPERIMENTATION, INVESTIGATION AND KNOWLEDGE DISCOVERY	4.2 ENTERPRISE AND BUSINESS CONTEXT
2.3 SYSTEM THINKING	4.3 CONCEIVING, SYSTEMS ENGINEERING AND MANAGEMENT
2.4 ATTITUDES, THOUGHT AND LEARNING	4.4 DESIGNING
2.5 ETHICS, EQUITY AND OTHER RESPONSIBILITIES	4.5 IMPLEMENTING
	4.6 OPERATING

This syllabus extends what is present in ABET outcomes by informing engineering educators of the engineering contextual elements that should be addressed in an engineering curriculum otherwise not explicitly addressed. Whereas ABET focuses on organization to ensure that engineering programs have a structure in place to constantly maintain, monitor, and improve their educational outcomes and objectives, the CDIO syllabus provides a detailed list for the context of what should be in an engineering education program. A major philosophy of the CDIO initiative is to motivate universities to address the professional *and* personal skills: in addition to disciplinary and engineering knowledge, there is an emphasis on having students learn to be engineers, and not simply learn engineering science.

Complex Systems (and Related) Education in non-SE Programs

There are several undergraduate and graduate degree programs labeled as Complex Systems available in North America. These are mostly associated with computer science and networking (like the one at Indiana University [2012]) or associated with biologic sciences (like the one at Florida Atlantic University [2012]). Additionally, there are several certificate programs, like those at the University of Michigan [2012] which is aimed at students “in the physical, biological and social sciences” and at Duke University [2012]. Most claim to be interdisciplinary, and do include courses in physics, math and computer science. However, the “genealogy” of these programs and their host departments or centers within their respective larger universities seems

apparent from their required courses. That is, one can tell the Indiana University program is managed by their computer science department because many courses cover "informatics" and "biologically inspired computing" [Indiana University, 2012]. The author could find no degree programs in complex systems engineering or in design of complex systems.

Courses in complex systems are offered at virtually every major university in the United States. They are mostly offered as part of degree programs in applied math, computer science, physics and economics. Rather than present an exhaustive list of courses and their content, a brief sampling is provided to describe the material currently being taught. Typical courses are named "Phys 580 Empirical Analysis of Nonlinear Systems," "Math 550: Introduction to Dynamical Systems" [University of Michigan, 2012], "INFO I690 Mathematical Methods for Complex Systems," "INFO I585 Biologically Inspired Computing" [Indiana University, 2012] and "MAP 6211 Introduction to Dynamical Systems and Chaos" [Florida Atlantic University 2012]. Typical topics include "fractals, emergent behavior, chaos theory, cooperative phenomena, and complex networks" at Indiana University [2012] and "one-, two- and higher dimensional flows, oscillator theory, maps, attractors, bifurcations, chaos" at Florida Atlantic [2012]. One syllabus offered a broad yet tangible set of learning objectives:

- Understand the theoretical foundations of complexity: mathematical, computational and information-theoretic;
- Reason about complexity with fluency and using scientific terminology; appreciate it as a post-Newtonian paradigm for approaching a large body of phenomena;
- Apply the terminology and methodology of nonlinear dynamics and chaos to study the time-evolution of complex systems;
- Describe some of the well-known models of complex systems, such as artificial life and Self-Organised Criticality (SOC);
- Construct new variations on models for complex systems using the modelling techniques taught in this course;
- Implement these models elegantly and efficiently using Java or MATLAB;
- Appreciate the interdisciplinary nature of complexity and the pivotal role that computer scientists play in its study. [Adalat, 2012]

Some economists seem to have been the first to suggest application of nonlinear dynamics to complex system design outside the physical sciences [Richardson, 2011]. The concept of a system's behavior being modeled by a set of interdependent differential equations in state-space form has been second nature to physicists, mathematician and engineers. The breakthrough for economists was Jay Forrester [1968] representing the equations graphically as stocks and flows [Richardson, 2011] rather than as state variables and rates of change. Forrester [1998; 1968] recommends educating "enterprise designers" who would use quantitative approaches to describe endogenous system feedback and computer modeling of nonlinear system dynamics. He recognized the key to correct design is not integration of a plant with a separate control system, but determination of system parameters that result in acceptable fluctuations around a desired equilibrium point (or fixed point, to use applied mathematics terminology). Thus, a corporation could be designed to be less vulnerable to outside influences and be more stable [Forrester,

1998]. This approach to design, in which optimality is traded-off in favor of robustness-by-structure, is a key principle in engineering complex systems [Minai, et al, 2010].

Potential Applications of Complexity Theory to System Design

Complexity lies between perfect chaos and perfect order [Miller & Page, 2007]. Systems can be considered complex because they exhibit attributes that we would say is neither completely chaotic nor completely orderly. Problems arise in this no-man's-land. This goes beyond considering stochastic processes (or noise) during design, which can be estimated and 'designed around.' Rather, the focus is on the appearance of near-periodic behavior and extreme sensitivity to initial conditions. It is less about including control subsystems and more about designing in parameters that will lead to the desired performance of the system. However, as intimated above, there is little information available on how complexity theory and principles have been successfully applied to real problems which have resulted in operational systems. The only exception is the development of models used to predict and analyze existing systems which include ecosystem preservation, pandemic prediction, and population growth and migration. Some have been so bold as to examine global warming and state-sponsored terrorism [Richardson, 2011]. Those computer-based models have obvious utility in providing insight for understanding and even predicting behavior. But, the problem remains: what to *do* about those actual systems that are being modeled? The actual implementation of systems including hardware, software, people-ware based predominantly on complexity theory lags. Nevertheless, many possible uses present themselves. A complete list of every effort in this area would fill volumes. Instead, the author just describes a few to illustrate the breadth of applications.

Nonlinear dynamics applies to the analysis of virtually every system which can be modeled as a set of interconnected, nonlinear or time-variant equations. That is virtually every system known, artificial and natural. Whenever we can describe or model a system in terms of nonlinear differential equations (or difference equations in discrete-time), then these concepts can be brought to bear in design. Most engineering courses provide tools to explicitly remove nonlinearities. These include manipulating equations in state-space models, producing a linear approximation by expanding a series around an operating point or by simply assuming it away. While this is acceptable in many instances, it simply is not generally applicable. Instead, engineers should embrace nonlinearities. The techniques explored by Strogratz, Wiener, Belousov & Zhabotinsky, and Kuramoto (just to name a few) are not too mathematically daunting. If one can use one of the built-in ordinary differential equation solvers in MATLAB or write a simple Euler integration or second-order Runge-Kutta routine in Excel, one can use those to explore nonlinear systems. However, traditional approaches for time- and frequency-based analysis can break down. Instead, use of state-space representation is preferred to identify fixed points and to consider basins of attraction. (As an aside, armed with nothing more than a copy of Strogatz [1994] and built-in help menus, the author created a MATLAB simulation that animated the famous Lorenz attractor in less than 15 minutes.) Of course, the same caveats that apply to all models regarding garbage-in and garbage-out also apply here. And, that's the trick – to discover the correct aspects to include in a model and to replicate them with appropriate fidelity. However, with education in complexity theory, it becomes a manageable trick rather than being just beyond the reach of traditional techniques. One recent example with an immediate defense application was described by Behdad, Al-Joumayly and Li [2011]. In their work, a pair of loosely-coupled nonlinear circuits was designed to enhance the time difference of arrival of RF

waves between two antennas in an electrically small array in support of direction-finding. A set of nonlinear equations were simulated via computer model and used to explore the system behavior and optimize design parameters for a 300MHz signal and an array of only 5cm. Several prototypes were built and tested. Measured results agreed relatively well with the theoretically predicted ones. More importantly, the output phase differences of the new arrays were enhanced and their sensitivity patterns were more directional compared to the non-coupled counterpart. Having a small form-factor is self-evidently important for applications in which small physical size or low power consumption are architecturally relevant, so designing such an array that retains high direction-of-arrival (DOA) sensitivity is beneficial. This is but one simple example. Other successful applications have come in the fields of mechanical vibrations, lasers, superconducting circuits, chemical oscillators, genetic control systems, and communications [Strogatz, 1994].

Emergence and decentralized systems concepts may be applied to the design and evaluation of cooperative autonomous systems. Autonomous system swarm designers are drawn to the study of systems that can organize without an organizer and can coordinate without a coordinator. Research in this area is so prolific that it has practically become its own discipline. Agent-based modeling with both homogenous and inhomogeneous populations and the use of probabilistic finite-state automata interacting with each other through stigmergy are the leading tools in this realm. To bring some more traditional mathematical models, we could consider our agents as really fulfilling Markov properties – their future state is based only on their current state (not what came before) and some triggering event (transition) along with the probability of transitioning to another state having sensed the event. Then, if we say the number of agents in each state is a variable, it should be possible to write a set of differential equations (or difference equations for discrete-time models) in which the rate of change for each variable is a function of the number of agents in each state and the likelihood of an event. This methodology has been shown to agree well with experimental data [Winfield, et al, 2009]. However, that only gets us part-way there. The problem is that finding a clean relationship between swarm population-description (in terms of number of agents in each state) and its purposeful action is difficult or impossible. If we can define purposeful action directly in terms of the number of agents in a given state(s) or define a new ‘state’ to capture the number of agents individually engaged in some desired activity, then we at least have an approach to design: to simulate such a model and explore the impact of state-transition triggers and probabilities. However, this methodology is not generally applicable. It seems more promising if the agents are somewhat more intelligent. If they are able to make some computations and predictions and then communicate with a richer set of information, they can find a ‘good enough’ localized behavior more quickly than a globally-computed optimum solution [Palmer, et al, 2003].

The assertions that swarms are more reliable and more robust than their traditional, centralized counterparts have gone mostly unchallenged. Only a few researchers have sought to quantify this robustness [Bjerknes & Winfield, 2010]. In turns out that swarms are not a panacea. That is, they suffer from the same issues that plague centralized systems – they can fail, and sometimes in unanticipated failure modes. Further, the simplification that all agents are identical (or vary according to some probability function regarding their propensities to react to stimuli) may not be completely satisfactory. Rather, a mix of diverse types of agents is expected to demonstrate improved performance in tasks relying on both exploration and exploitation [Page, 2010].

Small world networks exhibit properties that can be applied in network design and analysis. This is not just computer-communication networking, but is more generally applicable to populations in which individuals are represented as nodes and their connections with each other are links. Thus, we can have a business network, a family support network or a terrorist network. Small-world networks are characterized by a non-uniform distribution of links between nodes. Most nodes are just connected to the nodes in their immediate neighborhood while a few are connected to far nodes. (Here, the concepts of near and far are based on the number of nodes in between, not on geographic distance.) The short-distance nodes rely on adjacent long-distance nodes to bypass many node-to-node-to-node interchanges. Having just a few long-distance connections dramatically reduces a network's average path length. That means the network can enjoy the speed of information transfer while incurring a relatively small cost of maintaining only a few long-distance links. Scale-free networks can be described as a special case of small-world networks. In scale-free, or scale-law, networks, a very few nodes have many links while many nodes have few links and there is a large variety of different numbers of links. It is self-similar and follows a power-law distribution. This kind of network is very robust. Many of the low-link nodes can fail without significantly impacting the performance of the rest of the network. However, if only one of the many-link nodes (or hubs) fails, network performance degrades. The implications for distributed systems are profound. Communication systems, power grids, disease control and corporate strategies have all benefited from modeling and analysis based on small-world network concepts [Mitchell, 2011]. However, it is not a panacea for improving distributed system design or exploiting weaknesses in an enemy's network (such as disrupting terrorist cells or engaging a near-peer competitor in command and control warfare). A system's behavior is not completely described by its network properties. Even if it were, the models of networked systems should be viewed with some skepticism until properly validated.

Discovery of *real* problems, ideation of alternative solutions and comparative assessment of these alternatives are all in the realm of traditional systems engineering. This can also be said about issues regarding life-cycle balance like reliability, maintainability, supportability and total ownership cost. So, the argument might be made that it is not systems engineers who would benefit most from formal education in complexity; but it is physicists, economist and domain-engineers that would benefit from a formal education in systems engineering. Forrester [1998] himself emphasizes that corporate policy, government laws and regulations and even business mergers "constitute major redesign of the world economy" when viewed as a system. "The shortcomings of those systems result from defective design, just as the shortcomings of a power plant result from erroneous design" [Forrester, 1998]. It seems if practitioners in the fields of complexity applied the most fundamental principles of systems engineering (define *the* problem, identify functions of a system solution to resolve the problem, compare alternative designs, and recommend a solution based on its quantitative impact on the problem versus its life-cycle cost), applications would be more forthcoming.

Recommended Learning Objectives for Systems Engineering Students

Learning objectives should explicitly define the knowledge and skills a learner can demonstrate upon completion of a course. They should be precise, measureable and verifiable. Being able to apply principles and theory of complexity in the above-listed applications requires more than the skill set defined by the traditional competencies and program outcomes of systems engineering education programs which are based on a deconstructionist approach. Articulating such a skill

set in terms of course learning objectives follows directly from what one would expect of a student intending to design complex systems. For this paper, the author has attempted to sort them by affinity.

Nonlinear dynamics

- Define nonlinear systems and differentiate them from linear systems
- Create 1-, 2-, and 3-dimension phase portraits
- Determine fixed points of sets of nonlinear systems; characterize them in terms of their stability and apply Jacobian matrices to quantify their behavior; map basins of attraction and eigenvectors
- Use phase portraits to characterize limit cycles
- Create computer models of nonlinear systems based on sets of nonlinear equations
- Use computer models of nonlinear equations to explore the effects of changing system parameters and initial conditions

Complex Adaptive Systems and Decentralized Systems

- Create models of probabilistic finite state automata
- Given a description of individual agent behavior, draw a state-transition diagram including states and triggers
- Describe how adaptive agents impact a system's macroscopic behavior
- Define the terms 'robustness' and 'stability' in the context of complex systems
- Characterize a system based on its diversity (across types and within types)
- Describe the strengths and weaknesses of decentralized systems compared to centralized control; determine in which situations one would be better than the other

Small world networks

- Define the terms 'node,' 'link,' 'hub' (or 'high-degree node'), 'clustering,' 'small world network' and 'scale-free network'
- Determine average path length, clustering coefficient, and degree distribution of a given network
- Describe how network resilience is related to average path length and clustering
- Model the impact of node failure in different kinds of network configurations

Applied Complex Systems Design (some adapted from Sheard [2009])

- Apply complexity theory to 'evolve' rather than to 'design' a system; describe the impact of environmental forces and mutation
- Describe the impact of self-modifying components
- Leverage emergence by designing in parameters that lead to stable desired outcomes even in changing environments
- Apply diversity by employing different types of elements (homogeneity is efficient, but not robust); explore trade-offs in diversity within types and across types
- Identify elements that work on different time-scales in different layers
- Apply the concept of "satisficing" for second-priority components
- Consider network effects, especially human-centric networks; model changes and alternative designs to quantify robustness and identify vulnerable nodes
- Describe how design and development organizations are themselves complex systems.

A Framework for Assessing Benefits and Costs of Complexity Education

Adding a formal course in complexity for systems engineers may not be the best approach for all programs. There should be some means to explicitly consider its benefits as well as its costs. Many programs are already overloaded. Considering the number of skills and desired outcomes of traditional systems engineering described in the opening sections of this paper, a 2-year program can barely cover it all. So, the greatest cost is in what topics or courses or program outcomes we would be willing to give up in exchange for complexity topics.

Benefits should be judged based on the intended fields and applications of graduating students. Grouping these fields cleanly into non-overlapping sets of "traditional" and "complex" engineering is no easy task. In fact, the concept of a spectrum from complete order through complexity to chaos is useful [Beckerman, 2000; Sheard, 2009]. At one end, the systems of interest exhibit inter-element interaction dominated by hierarchical relationships, linear and time-invariant relationships, managed boundaries and obvious direct connections between cause and effect. At the other, behavior is almost completely stochastic and individual elements are not of interest, but statistical approaches can "average out" individual actions so a designer can focus solely on macroscopic properties [Beckerman, 2009; Miller & Page, 2007]. In between the ends are systems that are characterized by many diverse elements interacting strongly with each other in nonlinear ways, intra-system interactions that do not follow any system hierarchy and can change over time, apparent disconnect between cause and effect, unknown or unmanageable boundary conditions, and an operational environment that is not static. This is the realm of complexity.

Working in the middle ground, knowledge of complexity principles and mastery of the learning objectives outlined in the previous section would serve a systems engineer well. At either ends of this spectrum, formal education in complexity would be of limited value. The traditional reductionist approaches apply and are well-known. Now, that is not to say that design and development of such complicated systems are easy. It is just a means to characterize the problem and solution spaces in terms of complexity theory in order to assess the expected success of a systems engineer faced with different challenges. In the middle of our spectrum, the systems engineer would rely on his education to apply nonlinearity, emergence and decentralized self-organization, multi-scale influences, and diversity. Real success may only be achieved via recognition that complex systems are never really completed, but evolve continuously. They are self-integrating and operated over ambiguous boundaries. Thus, a designer's task is to identify those most critical, selective pressures and provide for individual components to interact in such a way that their relationships themselves can change in response. In a domain of self-organizing and evolving systems, the designer should focus on establishing and nurturing a development environment to address overall coherence rather than complete control over all design variables. However, even in complex domains, many system components exhibit traits that are fit for traditional systems engineering. Thus, complex systems engineering should complement and not replace traditional approaches.

List of References

A. Adalat. “Complex Systems” on-line syllabus. Imperial College London. (http://www.doc.ic.ac.uk/teaching/coursedetails/320 accessed 12 May 2012)

Y. Bar-Yam. “Engineering Complex Systems: Multiscale Analysis and Evolutionary Engineering.” in D. Braha, A. A. Minai, Y. Bar-Yam (editors), *Complex Engineered Systems*. Springer Media. 2010.

L. P. Beckerman. “Application of Complex Systems Science to Systems Engineering.” *Systems Engineering*, Volume 3, Issue 2. 2000.

N. Behdad, M. A. Al-Joumayly, and M. Li. "Biologically Inspired Electrically Small Antenna Arrays With Enhanced Directional Sensitivity." *IEEE Antennas and Wireless Propagation Letters*. Volume 10. 2011.

J. D. Bjerknes and A. F. Winfield. “On Fault Tolerance and Scalability of Swarm Robotic Systems.” Distributed Autonomous Robotics Conference. Lausanne, Switzerland. 2010.

Body of Knowledge and Curriculum to Advance Systems Engineering (BKCASE) Charter, September 2009 (http://www.bkcase.org/about-bkcase/project-charter/ accessed Aug 2, 2011).

C. N. Calvano and P. John. “Systems Engineering in an Age of Complexity.” *Systems Engineering*, Volume 7, Number 1. 2004.

C. Chittister and Y.V. Haimes. “Harmonizing ABET Accreditation and the Certification of Systems Engineers.” *Systems Engineering*, Volume 14, Issue 3. 2011.

A. Clayton, A. Wiborg and A. Riva. “A Rationale and Framework for Establishing a Systems Engineering Community within the Department of the Army.” Acquisition Research Sponsored Report, NPS-SE-11-002. 2011.

E. T. Crawley, J. Malmqvist, W. A. Lucas, and D. R. Brodeur. “The CDIO Syllabus v2.0, An Updated Statement of Goals for Engineering Education,” Proceedings of the 7th International CDIO Conference, Technical University of Denmark, Copenhagen, June 20 – 23, 2011.

Duke University, Center for Nonlinear and Complex Systems. (http://gradschool.duke.edu/depts_progs/certificate/nlcs.php accessed May 5, 2102).

T. Ferris, A. Pyster, D. Olwell, A. Squires, N. Hutchison, S. Enck (Eds.). “Graduate Reference Curriculum for Systems Engineering. Version 0.25.” Stevens Institute of Technology, Hoboken, NJ, USA. Released for limited review, 2010.

J. W. Forrester. *Principles of Systems*. Pegasus Communications. 1968.

J. W. Forrester. “Designing the Future.” Transcript of a lecture at Universidad de Sevilla, 1998.

Florida Atlantic University. Center for Complex Systems and Brain Science. (http://www.ccs.fau.edu/phdprog.php accessed May 5, 2012).

INCOSE, *Systems Engineering Handbook: A Guide for Systems Life Cycle Processes and Activities*, Version 3.2, INCOSE-TP-2003-002-03.2, International Council on Systems Engineering, 2010.

Indiana University, Graduate Programs in Complex Systems. (http://www.soic.indiana.edu/graduate/programs/complex-systems/index.shtml accessed May 5, 2012).

ISO/IEC, Systems and Software Engineering - System Life Cycle Processes, ISO/IEC 15288: 2008(E), International Organization for Standardization, Geneva. February 2008.

R. Jain, A. Squires, D. Verma and A. Chandrasekaran. "A Reference Curriculum for a Graduate Program in Systems Engineering." *Insight*, Volume 10, Issue 3. July 2007.

J. R. A. Maier and G. M. Fadel. "Understanding the Complexity of Design." in D. Braha, A. A. Minai, Y. Bar-Yam (editors), *Complex Engineered Systems*. Springer Media. 2010.

Menrad and W. Larson. "Development of a NASA integrated technical workforce career development model—the ROCK." Proc 59th Int Astronaut Cong (IAC) 2008, September 29–October 3, 2008, Glasgow, Scotland, UK.

J. H. Miller and S. E. Page. *Complex Adaptive Systems*. Princeton University Press. 2007

A. A. Minai, D. Braha, and Y. Bar-Yam. "Complex Engineered Systems: A New Pardigm." in D. Braha, A. A. Minai, Y. Bar-Yam (editors), *Complex Engineered Systems*. Springer Media. 2010.

M. Mitchell. *Complexity: A Guided Tour*. Oxford University Press. 2011.

National Defense Industrial Association (NDIA) Systems Engineering Division Task Group Report. "Top Five Systems Engineering Issues within Department of Defense and Defense Industry." July 2006.

D. O. Norman and M. L. Kuras. "Engineering Complex Systems." in D. Braha, A. A. Minai, Y. Bar-Yam (editors), *Complex Engineered Systems*. Springer Media. 2010.

Office of the Under Secretary of Defense for Acquisition, Technology and Logistics (OUSD AT&L), *System Engineering Guide for System of Systems, v1.0*. June 2008.

S. E. Page. *Diversity and Complexity*. Princeton University Press. 2010

D. Palmer, M. Kirschenbaum, J. Murton, K. Zajac, M. Kovacina, R. Vaidyanathan. "Decentralized Cooperative Auction for Multiple Agent Task Allocation Using Synchronized Random Number Generators." Proceedings of the 2003 IEEU/RSJ Intl. Conference on Intelligent Robots. Las Vegas, NV. 2003.

G. P. Richardson. "Reflections on the foundations of system dynamics." *System Dynamics Review*. Volume 14, Number 3. 2011.

S. A. Sheard. "Practical applications of complexity theory for systems engineers." Proceedings of INCOSE 15th Annual International Symposium. 2005.

S.A. Sheard and A. Mostashari. "Principles of Complex Systems for Systems Engineers." Systems Engineering, Volume 12, Number 4. 2009.

A. Squires. "Developing a Systems Engineering Curriculum Framework." School of Systems and Enterprises White Paper, Stevens Institute of Technology. 2007.

A. Squires and R. Cloutier. “Evolving the INCOSE Reference Curriculum for a Graduate Program in Systems Engineering.” *Systems Engineering*, Volume 13, Issue 4. 2010.

A. Squires, T. Ferris, D. Olwell, N. Hutchison, S. Enck, A. Pyster, D. Gelosh. “Work In Process: A Body of Knowledge and Curriculum to Advance Systems Engineering (BKCASE™).” IEEE International Systems Conference, Montreal, Quebec, Canada, April 4-7, 2011.

S. H. Strogatz, *Nonlinear Dynamics And Chaos: With Applications To Physics, Biology, Chemistry, And Engineering*. Westview Press. 1994.

H. Todd and D. Parten. “A Systems Engineering Approach to Address Human Capital Management Issues in the Shipbuilding Industry.” Master’s Thesis, Naval Postgraduate School, Monterey, CA. 2008.

University of Michigan, Complex Systems Certificate (https://secure.rackham.umich.edu/academic_information/program_details/complex_systems/#Complex%20Systems accessed May 5, 2012).

A. F. Winfield, W. Liu, J. D. Bjerknes. “Functional and Reliability Modelling of Swarm Robotic Systems,” in P. Levi and S. Kernbach (editors), *Symbiotic Multi-Robot Organisms*, unpublished, 2009.