A Complexity Typology for Systems Engineering

Sarah A. Sheard,[1] and Dr. Ali Mostashari
Stevens Institute of Technology
Castle Point on Hudson
Hoboken NJ 07030
sheard@3MilSys.com / ali.mostashari@gmail.com

Abstract. This paper shows how the literature on complexity is related to complex systems and to systems engineering. A framework for types of complexity is proposed that includes three types of structural complexity (size, connectivity, and architecture), two types of dynamic complexity (short-term and long-term), and one additional type, socio-political complexity. These types cover most of the characteristics of complexity mentioned in the reviewed literature; omissions are noted.

It would be advantageous to identify specific measures of complexity so that the complexity of systems or development programs could be compared and tracked, and risks could be identified and mitigated. To this end an overview of systems engineering measurement is provided, followed by a discussion of how complexity types might be able to be measured. At this point, however, a composite measurement of the six types of complexity is not recommended.

1. Introduction

Complexity is blamed for problems in system development ranging from computer programming failures (McCabe 1976) to acquisition program failure (Murdock 2008; United States Government Accountability Office 2008). Systems engineering complexity measurement, which could in theory lead to an understanding and possibly control of complexity, is problematic: "Even students and scholars in complexity...use the word 'complexity' to describe different ideas and perceptions." (Suh 2005)

Sometimes complexity implies a large number of components. More often the problem lies in how those components are interconnected, because interconnections, and the architectures derived from them, can create emergent patterns and unintended consequences. Sometimes the system itself may be well-understood, but the problem of developing the system is complex because of the number of contractors or number of tasks in a development schedule, dependencies among these nodes, or socio-political aspects of the development effort. Often interconnectedness of tasks in a development schedule can create a cascading effect if one task runs into problems, the same way a crack can propagate through a material.

To identify useful and robust measures of complexity, one must first clarify what complexity consists of and what its effects are. To that end, this paper proposes a typology of complexity, consisting of six main subtypes. Thoughts about how these types could be measured are included at the end.

2. Systems engineering and complex systems science

Historical systems engineering

In the early 20th century, large and unprecedented technological systems of many kinds were created. Military aircraft in World War 1 followed the first airplane flight in by less than twenty years. An unprecedented alliance of European cryptographers broke the code on the German Enigma machine long before World War II started. (Schwager 1998) Should these technological developments be called engineering efforts, or are they perhaps early systems engineering efforts?

[1] Author to whom correspondence should be addressed

Hughes (1998) argues that systems engineering as a profession developed after World War II, with the specific purpose of managing complexity on large projects using new technology. From these projects evolved the field referred to as systems engineering, a new field combining systems analysis, operations research, management science, and systems integration. The field was documented in a textbook by Hall (1962), followed by dozens of other textbooks by the 1990s.

Even so, Friedman (1994) lamented: "In the future, we will either manage complexity, or complexity will manage us, and we will be overwhelmed by ineffective, costly and late systems which are unbalanced, incomplete, incoherent, irrational, divergent, undocumented, polluting, and generate unnecessary risk at every turn." When this was published, there was much disagreement as to the scope of systems engineering, its principles or even the name (is it "system" or "systems" engineering?).

Within the next five to ten years, systems engineering standards and capability models (Sheard and Lake 1998) were written, released, harmonized with other standards and models, and re-released.. The vocabulary defined in such documents has become well-understood by systems engineers, software engineers, and to a lesser extent, project managers. Several versions of the International Council on Systems Engineering (INCOSE) Systems Engineering Handbook (Haskins, Forsberg et al. 2008) have been released as well. The most recent, rationalized to an international systems engineering standard, serves as the basis for a systems engineering certification program. Currently many aspects of the practice of systems engineering have become standardized across the industry.

This is not to suggest, however, that this practice is based on an overarching and complete systems engineering theory. For one thing, standardized practices accumulated from the practical and industrial side, with programs reusing that which worked on previous programs. Industrial practitioners have also dominated INCOSE symposia, to the extent that researchers found a need for a focused "Academic Forum." Programmatically, incorporation of theoretically-suggested improvements has taken a back seat compared to reducing cost and complying with process standards, particularly since capability models became required in the mid-1990s. Those practicing systems engineers who have taken on larger workloads with the consolidation of the defense and aerospace industry have little time to read up on recent research. And finally, the dominance of software in new development has led some managers to turn to software engineers to do the systems engineering, even those who have little to no knowledge of systems engineering history or best practices. Thus theory has found too little traction in improving systems engineering practice.

Complex systems theory and systems engineering

Historically, systems engineering theory has consolidated wisdom from a variety of fields. Figure 1 shows a very sketchy view of this history; a wall-sized poster can also be drawn.(Mdd 2006) Systems theory beginning as early as cybernetics (See Ashby (1956) for a good history), general system theory (Bertalanffy 1969), and systems thinking, continuing into the 1970s (Checkland 1993), and beyond (Weinberg 2001). System dynamics, in the form of structured analysis and business analysis, has been used in systems engineering for several decades.(LaPlante 2004) These have been applied to the evolving fields of software engineering and systems process discipline.

Theory derived from more recent complex systems sciences has not been explicitly incorporated into systems engineering standards or into most practice. While studies of fractals (Addison 1997), chaos (Gleick 1987) and complexity (Waldrop 1992) were enabled by widespread availability of digital computers beginning in the 1970s, topics such as the numerical modeling of complex adaptive systems (Miller and Page 2007) and theoretical fields such as nonlinear dynamics and networks have been folded into systems engineering only much more recently. Since 2000, a recommended new kind of systems engineering has been identified. This field is variously called Engineering Systems (Moses 2004), Enterprise Systems Engineering (White 2006) and Complex Systems Engineering (Sheard and Mostashari 2009). "Systems of Systems engineering" is closely related (Office of the Undersecretary of Defense

2008), although that particular guide intentionally only discusses extensions to current systems engineering practice rather than anything totally new.

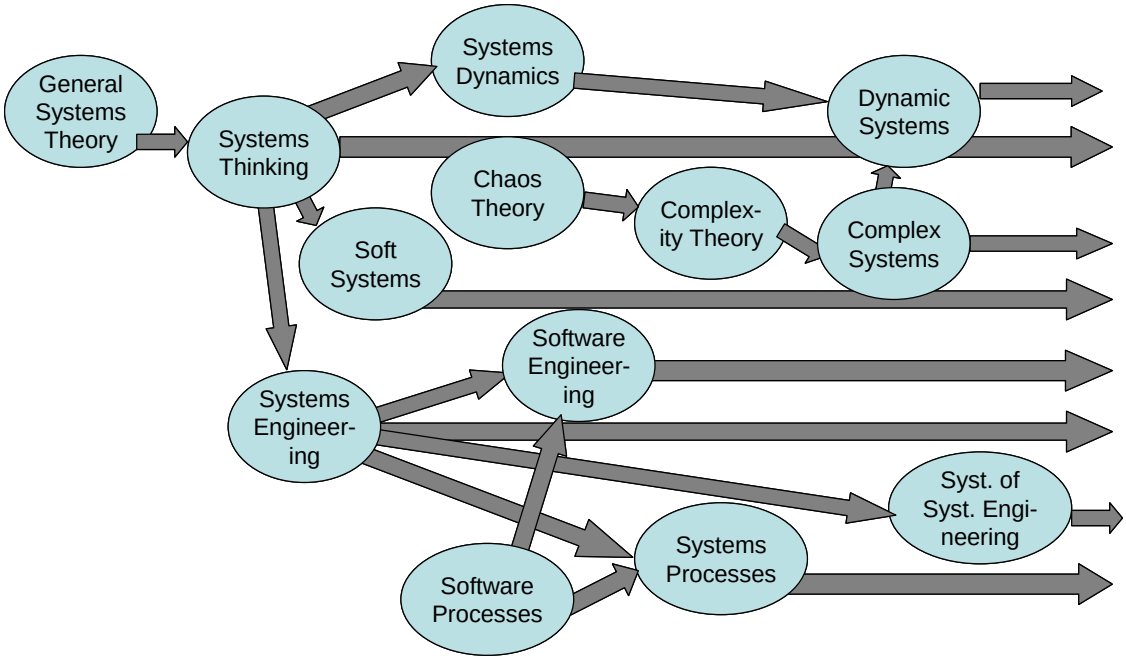

Figure 1. History of systems fields

This new field is still expressed in terms of broad descriptions of recommended improvements, rather than a filled-in body of knowledge. "Systems engineering" and "Complex systems" are both very broad fields. In addition, the intersection of the two fields is not smooth but patchy: a particular principle connects one small systems engineering task to one specific kind of complex analysis, under specific conditions. The problem is exacerbated by the fact that today's standard systems engineering, centered around system life cycles and standard processes, is but a subset of the early endeavors labeled "systems engineering." The broader type documented, for example, in (Hall 1962) addressed not only process but also systems engineering patterns; environmental research; economic, psychological, and "casuistic" theories of value; statistical decision making; and even "psychological aspects of synthesis."

Today's systems engineering challenges

Many of today's systems engineering challenges are not technical problems such as those addressed in engineering education. Murdock (2008) lays out the causes of "DoD's Systemic Acquisition Failures." Acquisition problems include decisions of what to acquire, funding issues, requirements creep, loss of internal technical competency, and incentive structures; in summary: "The defense acquisition system is incredibly complex, process-centric and risk-averse." In MITRE's Enterprise Systems Engineering Profiler tool (Stevens 2006), which quantifies program complexity, six of the eight octants have nothing to do with the technical system; rather they deal with stakeholders, scope, and acquisition context. The DoD's recent Guide to Systems Engineering of Systems of Systems (Office of the Undersecretary of Defense 2008), which shows adaptations of standard systems engineering as it applies to larger and more complex efforts, notes challenges such as "Coopetition" (development teams of multiple contractors who also compete against each other), spatially distributed development, and compliance with laws of multiple nations. These problems fall outside the standard scope of engineering; yet if they are not addressed, technical solutions will fail.

Most of the specifically technical challenges have been the focus of improved technology efforts for some time. These include a preponderance of increasingly intricate software, interoperability of systems that were originally designed to be independent, and evolving systems-of-systems whose inner workings are continuously exchanged. Contributing to complexity of this improved technology is more compact hardware (which is more tightly integrated) and the fact that today's software must be protected from unknown and increasingly sophisticated threats.

3. Defining complexity in the context of systems engineering

The goal of systems engineering is to create systems that work together and perform their intended function well in the operational environment. Before the last few decades, engineered systems were generally stand-alone systems with a single purpose that was usually limited in time and space. Assuming there was a feasible design in the first place, control of the development process usually sufficed to ensure the systems were built.

Consequences of recent complexity increases include: (Ameri, Summers et al. 2008)

- Increases in product life cycle costs (costly)
- Difficulty of getting engineering changes made (unmaintainable)
- Difficulty in servicing, leading to many failure modes (unrepairable)
- A complex supply chain, resulting in management and logistical problems (takes so long to build that it is practically obsolete upon delivery)
- The need for a complex, and therefore costly, design process

Figure 2 shows these as some of the effects of complexity on the right. Characteristics that create complexity are shown on the left.

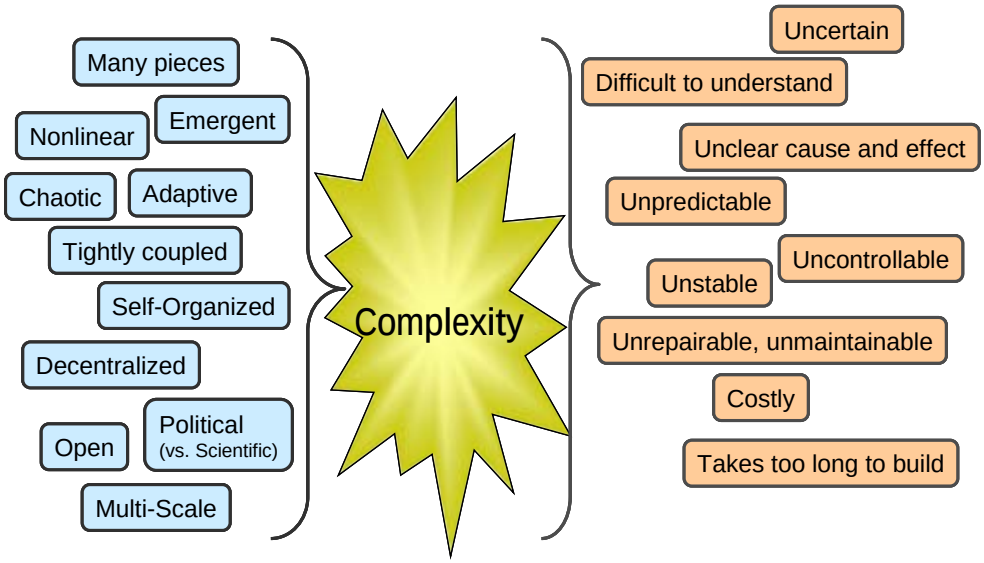

Figure 2. Contributors to and Effects of Complexity

Establishing a practice of improved systems engineering for the future will require addressing complexity explicitly. Systems engineers need to know "*when* a systems problem, and/or its solution, is complex*; how* complex it is (in relation to other systems, as a minimum)*;* [and] *how to manage* the complexity to permit us to answer...questions such as: 'When have we done enough? Is our confidence at an acceptable level?' "(Calvano and John 2004) Systems engineers will need to understand what complexity is and how to measure it. Summarized definitions from the literature are therefore provided in this section. The literature reviewed came both from the complex systems sciences and from systems engineering.

Things and Relationships

Systems engineering generally talks about "elements" and "interfaces". This dual concept of "things" and "relationships among the things" is common across many fields, although the vocabulary varies from field to field, as suggested by the terms in Table 1. No difference has been found between describing a system as consisting of "nodes" and "links" and describing it as having "entities" and "relationships," so for the purpose of this paper, all the terms in a column of Table 1 are considered equivalent.

Table 1. Equivalent Things and Relationships

Thing	Relationship	Source Field
Elements, components, systems, or subsystems	Interfaces	Systems engineering
Stocks	Flows	Systems dynamics
Nodes	Links	Network science
Vertices	Edges	Cyclomatic complexity
Modules	Messages	Software development
Entities	Relationships	Systems analysis
People	Connections	Social networking
Tasks	Dependencies	Project management, PERT
Process activities	Sequence of activities	Process engineering

Literature Overview and discussion: Complexity types

Sheard and Mostashari (2009) discuss the literature in detail, showing how the concepts addressed by Sussman (2000), Bolton (2007), Moses (2002; 2004), Bar-Yam (1997), Miller and Page (2007), Rouse (2007), Calvano and John (2004), Holland (1995), and Mostashari and Sussman (2009) correspond to the framework here. It is shown that the six types and subtypes of complexity (Table 2) adequately address most of the types in the literature, given that artifact and development process complexities have some differences.

The exception (those concepts from literature that are not well addressed by the typology) can be summarized as follows:

- Theoretical and mathematical complexity, as opposed to experiential and tangible complexity
- Causes and effects of various types of nonlinearity and emergence
- Environmental complexity, and a full set of subtypes of sociopolitical complexity (nested and evaluative subtypes are mentioned above)
- Risk, concerns and worries
- Uncertainty and measurement noise.

The concepts in the first bullet are only excluded from the typology to the extent that they are so theoretical as to be not usable on real programs (the types structural:size and structural:connectivity do address much in these concepts). Causes and effects of nonlinearity and emergence are interesting and possibly useful, but are not related to how complexity is measured, which is the focus of this work. Socio-political complexity deserves its own treatment, most likely in collaboration with sociologists. For now it is recognized that this type is poorly defined. Risks, concerns, worries, and uncertainty are more effects than types of complexity. Measurement noise is addressed briefly below.

4. Framework of types of complexity

Figure 3 shows how the various types of complexity interrelate. Green items (dashed border) are items whose complexity is at issue: systems (artifacts), the processes used to develop them, and the environment. Both systems and development processes exhibit both structural and dynamic complexity. Structural complexity (orange, dotted border) has three subtypes, described in more detail below Table 2. Dynamic complexity (pink, solid border) is split into short- and long-term. Socio-political complexity (blue, dot-dash border) applies primarily to the environment and development processes rather than to things, although this also plays a role in the function of the system itself, particularly in systems-of-systems.

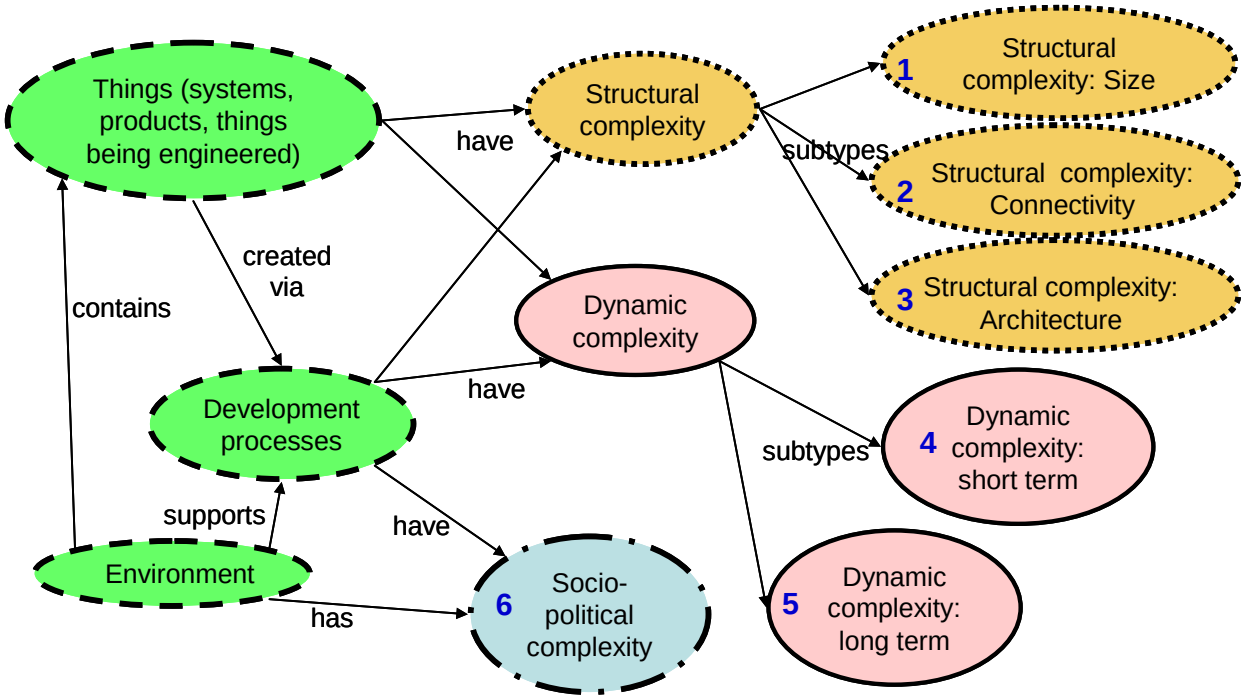

Figure 3. Relationship of complexity types

Table 2 lists these six types of complexity, including examples, why the factor contributes to complexity, and what kinds of problems can occur if a system (or, with a dash preceding it, a development process) has such complexity. Each type of complexity (row in the table) is discussed in more detail below the table.

Table 2. Framework of Types of Complexity

Type	Subtype	Example	Why complex?	Problem if have it
1 Structural complexity	**Size** (# elements, # instances, # types of elements) -of development process	People, groups, units, computer nodes -SE tasks	Many items are harder to track. (Some consider this "complicated" not complex)	Long list of elements, some won't be on time - Costly to staff all tasks

Type	Subtype	Example	Why complex?	Problem if have it
2 Structural complexity	**Connectivity** (# connections, types, strength of connections) -of development process	Number or density of connections, varieties (data/control/physical), (strong/medium/weak/ none) "Static emergence" -dependency links	Everything connects to everything else. Difficulty decomposing and isolating causes/effects. Determines network connectivity, path length, feedback loops	Emergent properties Cascading failures. Difficult to observe, describe, control, predict breakage -Program may not converge
3 Structural complexity	**Architecture** (Patterns, chunkiness of connections, inhomogeneity, boundaries)	Trees, layers, networks, teams Chunks, fragmentation, "non-holonomic constraints"	Chunks of order have different properties from other chunks or from the stew of complexity	Different patterns have different failures or strengths. Averaging across different types gives incorrect conclusions
4 Dynamic complexity	**Short term** (sudden rapid change in system behavior) -development system behavior	Homeostasis, Time patterns, Feedback. -Execution of development system (building of the product) is inherently dynamic	Difficult to predict, control, understand, communicate. Rapid problem exacerbation, oscillation due to time delays. -tasks may be out of sequence	Things can change suddenly and unpredictably. Hard to estimate. Typical linear and gaussian-based assumptions fail.
5 Dynamic complexity	**Long term** (changes in # and types of things and relationships)	Origin of complex systems, growth, evolution to a new system, self-organization, adaptation, learning, co-evolution causing changes in fitness landscape.	Evolution is long-term emergence. Almost impossible to predict how a system will evolve. Learning changes short-term and long-term behavior.	Adaptation is hard to predict. Difficult to engineer system qualities ("ilities") at end of life because both system and context change
6 Socio-political complexity	**Social and Political** (Human cognitive limitations , multiple stakeholders, global context, environmental sustainability, economics) -"Coopetition," supplier chain depth, distributed development	Objectives multiple, soft, value-laden. Multiple perspectives, multiple approvals required. Pluralism. Sociological aspects of teams and organizations, Diverse operational environments, diffuse boundaries	Individuals ambivalent and change their desires over time. Team dynamics and organizational dynamics can scuttle technical concerns. Politics can determine whether a system is built at all. Policies are resisted.	Goals change, acceptability of developed system changes, system requirements change, funding is cut, system may not be developable in time required.

Row 1 (Structural complexity: Size) groups all considerations about number of elements (nodes). How many nodes are there? How many types of nodes are there? How many instances total? The more nodes there are, the greater the amount of resources required to understand and manage a system, but greater numbers do not in themselves require a qualitatively different kind of management. Some consider the number of types of elements to be the only really complex part of Type 1, while others (Stevens 2009) consider this "diversity" rather than true complexity; in any case, no discussion of complexity can omit size altogether.

Row 2 (Structural complexity: Connectivity) includes all considerations having to do with number or types of connections (links, relationships, etc.). Note that number and density are essentially the same quantity, density being just the number divided by the maximum possible number of connections (which is (n)(n-1)/2 for bidirectional connections, considering only connections between two elements). Types of connections differ from representation to representation (Ameri, Summers et al. 2008) and include (for

example) signal, energy, and material in a "function structure" representation, and up to eight types for a "parametric associativity graph."

Combined with number of nodes (Structural complexity: Size), connectivity determines many network properties, including average network path length, degree centrality, group cohesion, and robustness to failure (Cross and Parker 2004). McCabe (1976) showed that the number of vertices (nodes) and links (edges) was sufficient to calculate the number of fundamental loops in a single software program.

When the issue is complexity of the development process, nodes are the tasks and connections are dependency links among them, including feedback that causes revisiting of a previous task due to outputs of a current task. If the connectivity is high enough, so much feedback can happen that the program may not converge at all (Mihm and Loch 2006).

Generally, emergent properties arise when connectivity gets high. Therefore emergence that is not explicitly time-dependent is included in this type. Dense networks (high connectivity) also make a system hard to decompose.

Connectivity may also include considerations of varying strengths of connections, whether simple bins of "weak, medium, or strong" or continuous variations in neuron-neuron strength as the basis for memory in artificial neural networks (Stergiou and Siganos 1996), for example.

Row 3 (Structural complexity: Architecture) was necessary to deal with issues such as architectural patterns (tree, layers, teams, etc.) and concepts of boundaries and "chunkiness" (e.g. islands of order within a complex or chaotic background). These imply a certain level of connectivity, but the concept is different because connectivity refers to a total or average number, while architecture specifically looks at inhomogeneity. This type of complexity arises from things not being the same across their expanse, including the effect of distinct spatial areas on populations that grow in an environment (Sayama, Kaufman et al. 2000) and boundary issues. "Texture" is a slightly broader term than "chunkiness" and may apply if the chunks get small enough.

Row 4 (Dynamic complexity: short term) is one of two types of dynamic complexity. Dynamics occur at all time scales, but in general there are two different kinds of effects. One effect is seen at an operational time scale, best exemplified by the concern that in high complexity systems, negative events may grow quickly, surprising operators and requiring emergency measures. Whether considered a nonlinear or "butterfly" effect, or analyzed in terms of feedback, homeostasis, and Gaussian distribution "outliers," these effects are difficult to understand, predict, control, or prevent.

It should be noted that development processes are inherently dynamic (in contrast to artifacts, which have at least some static properties). A development process by nature implies beginning with early steps, completing them, and over time progressing through the remaining steps.

Row 5 (Dynamic complexity: long term) addresses the longer time scale, the time scale over which systems evolve as the agents adapt, or perhaps die and are replaced by another generation (as a minimum; the time scale can also be much longer than a single generation). Complexity arising from the growth of complex systems, self-organization, or learning is considered long-term, as are many concepts related to the fitness landscape changing (although that can also occur on an operational time scale, particularly in war-like situations).

Row 6 (Socio-political complexity) is acknowledged as covering too much. Into this "type" has been squeezed anything having to do with humans, from cognitive limitations that prevent designers from being able to consider much more than 7 plus or minus 2 items at once, to sociological phenomena such as fads and marketing, to fields of economics, environmental sustainability, and politics. These are grouped together primarily because most engineers have neither education nor aptitude to deal with them. Ideally, systems engineers can bring specialists in these fields onto programs at appropriate points, but

this ideal is not often enacted. This type has not been divided into sub-types, partly because differences among these fields are not evident to many engineers.

Socio-political complexity includes concepts of multiple stakeholders, viewpoints, and views; of soft, value-laden, and conflicting objectives; of pluralism (the social counterpart to "chunkiness"); management science and principles of teams; and competition among suppliers in the contractor marketplace. This latter mostly applies when considering the complexity of the development process rather than that of a product or artifact, but in fact this whole category is more about process than artifacts.

5. What is systems engineering measurement?

Systems engineering measurement was performed long before the discipline of systems engineering coalesced, as documented, for example, in standards and capability models in the 1990s (Sheard 2001). Individual programs measured progress using earned value and other project management techniques (Rosenau 1981) beginning in the 1960s or 1970s. From the earliest time in the modern era of systems engineering, programs tracked key technical parameters, a skill crystallized into Technical Performance Measurement, which was described in a 1974 Military Standard (U.S. Department of Defense 1974).

Improved and more broadly-accepted systems engineering measurement information came from four different directions, as follows:

- INCOSE, and in particular its Measurement working group, published a technical measurement guidebook (Roedler and Jones 2005) that refined and expanded an earlier Measurement primer (Measurement Working Group 1998).
- Practical Software Measurement (McGarry 2002) began as a software-only collection of common program issues and suggestions for their measurement. PSM's well-connected authors brought the same ideas into the CMMI (Chrissis, Konrad et al. 2003) and into international measurement standards (ISO/IEC 2007). Beginning in 2004, PSM explicitly added systems engineering. PSM has a well-developed method, process, lexicon, set of templates, examples, and recommended measures.
- USC's Center for Software and Systems Engineering developed first the CoCoMo (Constructive Cost Model) cost estimation model for software (USC Center for Software and Systems Engineering 2002) and then in 2006, a similar model for estimation of systems properties called COSYSMO (Valerdi 2006). These provide an industry-wide set of cost-drivers, and a baseline calculation capability that can be tailored to a particular organization.
- Leading Indicators. A 2007 report called Systems Engineering Leading Indicators (Roedler and Rhodes 2007) involved cooperation among most of the groups represented above. A complexity measure was one of the most requested additions for the 2009 update presently in work, but will probably not appear due to inability to capture all the different requested ideas in one measure (Sheard and Mostashari 2009).

6. Complexity measurement related to systems engineering

None of the INCOSE systems engineering measurement documentation addresses complexity. Sosa, Browning and Mihm (2007) have investigated software systems complexity and its relationship to architecture. While very promising, the concepts only apply to actual coded software, are not applicable to larger systems, and are not available early enough in the program to be considered "leading." PSM includes "cyclomatic complexity" (McCabe 1976). While this has been important in the software community, this does not translate easily to systems engineering. Only COSYSMO from USC addresses complexity, via the driver "architecture complexity", but this is estimated by experts, not calculated.

Architecture Complexity: The relative difficulty of determining and managing the system architecture in terms of IP platforms, standards, components (COTS/GOTS/NDI/new), connectors (protocols), and constraints. This includes systems analysis, tradeoff analysis, modeling, simulation, case studies, etc. (Boehm, Riefer et al. 2003)

At this time there is not a standard, frequently-used measure of complexity that is applied to systems engineering, and there is significant interest in such a measure (Sheard and Mostashari 2009; Roedler and Rhodes 2010).

The goal of systems engineering complexity measurement would be first, to be able to track changes in the complexity of a given system over time, and potentially second, to be able to compare the complexity of two different systems. Table 3 shows the types and sub-types of complexity from Table 2, followed by potential measures that could be used to evaluate the complexity of an artifact or development process according to that type of complexity. These basic measures could then be combined into a complexity indicator as shown in Roedler and Rhodes (2007) for other systems engineering measures. These are preliminary ideas that need significant study before implementing.

Table 3. Possible Measures of Various Types of Complexity

Type	Subtype	Possible measures
1 Structural complexity	**Size** (# elements, instances, # types elements) -of development process	# items (instances) # types of unique items # development tasks
2 Structural complexity	**Connectivity** (# connections, types, strengths), -of development process	# connections, density of connections (binned by strength?) # loops or threads per McCabe software complexity.
3 Structural complexity	**Architecture** (Patterns, chunkiness of connections, boundaries)	Measures similar to texture measures, such as size distribution function (Moskowitz and Jacobs 1975); (Kaye, Junkala et al. 1998) or Boundary fractal dimension (Wettimuny and Penumadu 2003)
4 Dynamic complexity	**Short term** (Nonlinearity, dynamic emergence, sudden rapid change in system behavior—butterfly effect) -development system behavior	In a system dynamics or agent-based model, predict characteristics of changes such as frequency and size (or consequence), and derive "change exposure" measures? - # Paths, deviations from waterfall, dependencies? PERT C measures
5 Dynamic complexity	**Long term** (changes in # and types of things and relationships)	ESE Profiler tool results Possibly: measures of resilience
6 Socio-political complexity	**Social and Political** (Cognitive limits, multiple stakeholders, global context, environmental sustainability, economics) -"Coop-etition," supplier chain depth, distributed development	ESE Profiler, sociological measures of group and organizational coherence, team measures, political measures such as size of constituency. Scale of effort in terms of number of users and user types.

These measures are all hypothetical, and very few relate to the practical measures documented in (PSM 2007). Many questions would have to be resolved, including those brought up by Ameri, Summers et al. (2008) such as whether the measure relates to the model of the product (the product representation) or to the product itself. Clearly models are easier to use theoretically, but if different models of the same product would give different measures, it is not clear that such measures would be useful to systems engineers or managers.

The Systems Engineering Leading Indicators revision project has been asked to propose a practical Complexity indicator for systems and/or programs (Roedler and Rhodes 2010). The requests were grouped together because they all mentioned "complexity," but some ask about size of the product while others refer to number of teammates or the risk of not meeting the development schedule.

Because of this lack of clarity it is not prudent at this time to recommend a composite measure that rolls up all six of these types into one number. It is unknown which of the six types is most likely to lead to problems, in general or for any specific system. Different systems will likely have different mixes of complexity, and a single number that was 95% social complexity might have very different implications to a program than a single number that was 95% short-term dynamic complexity. There would also be little credibility to any algorithm that would suggest how to combine the six types. Most likely the initial work on these complexity measurements will need to capture aspects of all six types and study correlations to see what aspects do lead to problems and what aspects are more benign.

However, once a candidate measure is proposed, it will likely be well-tested in the field, since there is great interest in being able to determine the level of complexity on a program. This is a fertile area for research, which is expected to provide data about the difficulty of making theoretical work usable on real programs.

7. Conclusion

This paper has reviewed the literature on complexity, as related to complex systems and to systems engineering. Table 2 proposes a framework of complexity types that includes three types of structural complexity (size, connectivity, and architecture), two types of dynamic complexity (short-term and long-term), and socio-political complexity. For each of these types, the complexity of the development process is discussed separately from the complexity of artifacts, if appropriate. The paper shows that these six types address nearly all of the types of complexity mentioned in the reviewed literature.

The recent history of systems engineering measurement features a growing body of work starting with a definition of the measurement process, growing to an industry-wide database of cost drivers, and including, in the more recent documents, suggested leading indicators and measurement and indicator templates. To continue improving measurement of systems engineering toward a solid complexity measure, a number of potential measurements are discussed that relate to the six types of complexity. Because such theoretically based measurements are so different from the practical measures in use across the industry, it is premature at this time to define a composite complexity measurement.

Future work is proposed that will fill in holes in the understanding and definition of various aspects of complexity as they relate to systems engineering.

8. References

Addison, P. S. (1997). Fractals and chaos: an illustrated course. Philadelphia, PA, Institute of Physics Publishing.

Ameri, F., J. D. Summers, et al. (2008). "Engineering design complexity: an investigation of methods and measures." Res Eng. Design **19**: 161-170.

Ashby, W. R. (1956). Introduction to Cybernetics Chapman & Hall.

Bar-Yam, Y. (1997). Dynamics of Complex Systems. Cambridge, Massachusetts, Westview Press.

Bertalanffy, L. v. (1969). General system theory; foundations, development, applications. New York,, G. Braziller.

Boehm, B. W., D. J. Riefer, et al. (2003). COSYSMO, A Systems Engineering Cost Model. International Council on Systems Engineering. Crystal City, Virginia, INCOSE.

Bolton, D. P. W. (2007). Some Thoughts on Systems Engineering, Engineering Systems & Complexity Symposium on Complex Systems Engineering. Santa Monica CA.

Calvano, C. N. and P. John (2004). "Systems engineering in an age of complexity." Syst. Eng. **7**(1): 25-34.

Checkland, P. (1993). Systems Thinking, Systems Practice. Chichester, West Sussex, England, John Wiley & Sons.

Chrissis, M. B., M. Konrad, et al. (2003). CMMI: Guidelines for Process Integration and Product Improvement. Boston, Addison-Wesley.

Cross, R. and A. Parker (2004). The Hidden Power of Social Networks: Understanding how work really gets done in organizations. Boston, Massachusetts, Harvard Business School Press.

Friedman, G. J. (1994). "Systems Engineering's Crucial Juncture." The Journal of the National Council on Systems Engineering **1**(1): 1-6.

Gleick, J. (1987). Chaos: Making a New Science. New York, Penguin Books.

Hall, A. D. (1962). A Methodology for Systems Engineering, Van Nostrand Reinhold.
Haskins, C., K. Forsberg, et al., Eds. (2008). Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, v 3.1, International Council on Systems Engineering
Holland, J. H. (1995). Hidden Order: How Adaptation Builds Complexity. Cambridge, Massachusetts, Perseus Books.
Hughes, T. P. (1998). Rescuing Prometheus. New York, Pantheon Books.
ISO/IEC (2007). ISO/IEC 15939:2008E Systems and software engineering -- Measurement process. Switzerland.
Kaye, B. H., J. Junkala, et al. (1998). "Domain Plotting as a Technique for Summarizing Fineparticle Shape, Texture and Size Information." Particle and Particle Systems Characterization **15**(4): 180-190.
LaPlante, P. A. (2004). Real-time systems design and analysis. Hoboken NJ, IEEE Press Wiley-Interscience.
McCabe, T. J. (1976). "A Complexity Measure." IEEE Transactions on Software Engineering **SE-2**(4): 308-320.
McGarry, J. (2002). Practical software measurement: objective information for decision makers. Boston, MA, Addison-Wesley.
Mdd (2006). "User:MDD/History of systems theory." Retrieved April 1, 2010, 2010, from http://en.wikipedia.org/wiki/User:Mdd/History_of_systems_theory.
Measurement Working Group (1998). Systems Engineering Measurement Primer. Seattle WA, INCOSE, the International Council on Systems Engineering.
Mihm, J. and C. H. Loch (2006). Spiraling out of control: problem-solving dynamics in complex distributed engineering projects. Complex Engineered Systems: Science Meets Technology. D. Braha, A. A. Minai and Y. Bar-Yam. Cambridge, Massachusetts, Springer**:** 384.
Miller, J. H. and S. E. Page (2007). Complex adaptive systems: An introduction to computational models of social life. Princeton, NJ, Princeton University Press.
Moses, J. (2002). The Anatomy of Large-Scale Systems. ESD Internal Symposium, Massachusetts Institute of Technology Engineering Systems Division.
Moses, J. (2004). Foundational Issues in Engineering Systems: A Framing Paper. Engineering Systems Monograph, MIT esd.
Moskowitz, H. R. and B. E. Jacobs (1975). "The texture profile: its foundation and outlook." J. Texture Stud. **1**(6): 157.
Mostashari, A. and J. M. Sussman (2009). "A Framework for Analysis, Design, and Management of Complex, Large-Scale, Interconnected, Open Sociotechnological Systems." International Journal for Decision Support Systems and Technologies **1**(2): 52-68.
Murdock, C. A. (2008). Assessing DOD's Systemic Acquisition Failures. Washington, DC., Center for Strategic & International Studies**:** 8.
Office of the Undersecretary of Defense (2008). Systems Engineering Guide for Systems of Systems Version 1.0. Washington DC, Office of the Undersecretary of Defense (A,T and L), Systems and Software Engineering,.
PSM (2007). Practical Software & Systems Measurement: Objective Information for Decision Makers.
Roedler, G. J. and C. Jones (2005). Technical Measurement. INCOSE. Seattle, WA, INCOSE. **INCOSE-TP-2003-020-01**.
Roedler, G. J. and D. H. Rhodes (2007). Systems Engineering Leading Indicators Guide. Seattle WA.
Roedler, G. J. and D. H. Rhodes (2010). System Engineering Leading Indicators. Cambridge MA, Lean Aerospace Initiative.
Rosenau, M. D. (1981). Successful project management: a step-by-step approach with practical examples. Belmont, Calif., Lifetime Learning Publications.
Rouse, W. B. (2007). Complex Engineered, Organizational & Natural Systems: Issues Underlying the Complexity of Systems and Fundamental Research Needed to Address These Issues. Washington DC, Engineering Directorate, National Science Foundation**:** 31.
Sayama, H., L. Kaufman, et al. (2000). The role of spontaneous pattern formation in the creation and maintenance of biological diversity. International Conference on Complex Systems. Y. Bar-Yam. Nashua New Hampshire, NECSI**:** 8.
Schwager, R. (1998). "History of the Enigma Machine." Retrieved April 1, 2010, 2010.
Sheard, S. A. (2001). Evolution of the frameworks quagmire. Computer, IEEE. **34:** 96-98.
Sheard, S. A. and D. J. G. Lake (1998). Systems Engineering Standards and Models Compared. Eight Annual International Symposium of the International Council on Systems Engineering. Vancouver, British Columbia, Canada, INCOSE**:** 589-596.
Sheard, S. A. and A. Mostashari (2009). "A Complexity Typology for Systems Engineering." Syst. Eng. **(Submitted)**.

Sheard, S. A. and A. Mostashari (2009). "Principles of complex systems for systems engineering." Systems Engineering **12**(4).
Sosa, M. E., T. R. Browning, et al. (2007). Studying the Dynamics of the Architecture of Software Products. ASME 2007 International Design Engineering Technical Conferences. Las Vegas NV.
Stergiou, C. and D. Siganos (1996). "Neural Networks." Neural Engineering **4**(11).
Stevens, R. (2006). Engineering Enterprise Systems: Challenges and Prospects McLean VA, The MITRE Corporation**:** 19.
Stevens, R. (2009). Origin of ESE profiler. S. A. Sheard. McLean VA.
Suh, N. P. (2005). Complexity : theory and applications. New York, Oxford University Press.
Sussman, J. M. (2000). Ideas on Complexity in Systems--Twenty Views. M.I.T.
U.S. Department of Defense (1974). MIL-STD-499A Engineering Management Washington DC, Department of Defense.
United States Government Accountability Office (2008). Defense Acquisitions: Assessments of Selected Weapon Programs. G. A. Office. Washington DC.
USC Center for Software and Systems Engineering (2002, 9/23/2002). "COCOMO, Constructive Cost Model." Retrieved May 9, 2009, from http://sunset.usc.edu/cse/pub/research/COCOMOII/cocomo_main.html.
Valerdi, R. (2006, 12/6/2006). "Welcome to the home of COSYSMO." from http://valerdi.com/cosysmo/.
Waldrop, M. M. (1992). Complexity: the emerging science at the edge of order and chaos. New York, Simon & Schuster.
Weinberg, G. M. (2001). An introduction to general systems thinking / Gerald M. Weinberg. New York, Dorset House.
Wettimuny, R. and D. Penumadu (2003). "Automated Digital Image Based Measurement of Boundary Fractal Dimension for Complex Nanoparticles." Particle & Particle Systems Characterization **20**(1): 18-24.
White, B. E. (2006). On the Pursuit of Enterprise Systems Engineering Ideas. Bedford, MA, The MITRE Corporation.