Three Types of Systems Engineering Implementation

Sarah A. Sheard
Software Productivity Consortium
2214 Rock Hill Rd. Herndon, Virginia 21070
sheard@software.org

Abstract. INCOSE has been bedeviled by arguments about the definition of systems engineering. Many definitions have appeared, but the only one that is widely accepted is so broad as to be almost a tautology. As a result, INCOSE has been unable to answer many questions in a way that most members can accept.

This paper claims that systems engineering can be defined in a way that leads to clean answers to many questions. This definition claims that what people have been calling “systems engineering” can be split into three basic implementations or types of systems engineering: Discovery, a discipline or specialist type that involves significant analysis, particularly of the problem space; Program Systems Engineering, a coordination or generalist type that emphasizes the solution space and technical and human interfaces; and Approach, a process type that can (and should) be performed by any engineer. Such a breakout resolves controversies and has implications on systems engineering training, research, processes, standards, and promulgation.

INTRODUCTION: THE PROBLEM

“But that’s not systems engineering!” This phrase has been stated frequently on the INCOSE discussion list server. One member mentions what he or she clearly thinks is systems engineering, usually an aspect of coordination that has little to do with analysis. Another member then retorts that what is described may be program controls, but it is not systems engineering. Could it be that INCOSE members do not share a common definition of systems engineering?

“Why can’t we impose systems engineering on the commercial world?” “How can we justify the expense of performing systems engineering?” These issues, too, arise periodically on the discussion list.

Other questions that are often raised but not yet answered fall into the following categories:

Definition. Isn’t systems engineering just solving problems according to the scientific method? Isn’t systems engineering turning a complex problem into a statement of objective function? Isn’t concurrent engineering just good systems engineering? How does systems engineering relate to integrated product teams? Can you do systems engineering outside the field of systems engineering (for example, in law, education, or social science)? How does systems engineering fit with program management or disciplines such as software? Is it a discipline, a process, an approach, a program phase?

People. Who are systems engineers? Is systems engineering a job title, or does it describe anyone who wants to think about the larger system that a product fits into, or only people with “Systems Engineering” degrees, or something certifiable by Texas?

Value. What is the value of performing systems engineering? What are the benefits of systems engineering? How do you convince a manager to fund systems engineering?

Training. How should a systems engineer be educated? What on-the-job training is important?

Tools. What tools do systems engineers use? What tools can provide support for everything systems engineering does, in an integrated manner?

Measurement and assessment. How do you measure systems engineering? How do you assess a research and development organization, a maintenance organization, or an order fulfillment organization against a systems engineering model?

Standards. Who should use systems engineering standards, and how should they use them? Do the various standards apply differently to different implementations of systems engineering? How do systems engineering standards apply to a small company making piece parts or consumer goods?

Research. What kind of research is needed to make systems engineering a discipline on par with electrical or mechanical engineering?

Future. How is systems engineering going to change in the future?

These questions, when answered, will define systems engineering.
engineering in a manner that will help INCOSE turn systems engineering into more of a science—predictable, repeatable, trainable, and usable for improvements to the products of the world.

Many books and papers have attempted to define systems engineering (Kasser 1996, Bahill 1996, Blanchard 1998, Lake 1996, Senglaub 1996, Harwell 1996, Sheard 1996a, and many others). But they have not provided answers to these questions.

INCOSE needs a broad yet detailed description of systems engineering. This description should apply to most implementations and encompass most of what has been claimed about systems engineering, without being confusing or chaotic. It should also suggest a way to define products and directions for INCOSE. A definition meeting these criteria is suggested here, in the form of three types of implementations: Discovery, Program Systems Engineering, and Approach.

**THREE TYPES**

**Type 1. Discovery.** This is a specialist-type implementation of systems engineering in the form of (large) groups of “systems engineers” performing primarily systems analysis. This kind of systems engineering was one of the first recognized kind types. Several cases are very well characterized in (Hughes 1998), which describes the SAGE computer project, Intercontinental Ballistic Missile development, the Boston Central Artery/Tunnel project, and Arpanet/Internet. Air traffic control is another example. In this type of implementation, the problem space is incredibly complex. Modeling the problem space, and really understanding it, is most of the goal in Discovery implementations.

An unambiguous and unarguable statement of objective function, along with the models and analyses behind it, provides most of the value of Type 1 systems engineering.

The models and analyses created in Discovery implementations include requirements modeling, system behavior modeling, environmental and system simulation, reliability or survivability analysis, orbital analyses (for space systems), and contingency scenarios, to name a few. The output of Discovery systems engineering includes relationships between functionality and design implementation, cost and risk to achieve performance, technical feasibility, and new ways to allocate functionality. Analysis results include models from which a solution can be designed in a subsequent phase.

**Type 2. Program Systems Engineering.** The Defense Systems Management Guide (DSMC 1990) describes a generalist type of systems engineering in a way that sounds a lot like program management. A check of recent INCOSE proceedings books showed about two thirds of the papers were talking about Program Systems Engineering. In this implementation, specific people are designated “the systems engineers” and made responsible for the top-level design of the system, including system budgets, for specification of the pieces, and for coordinating the build and integration of the pieces. Large programs using Program Systems Engineering tend to designate a group of people as “systems engineers,” while smaller programs often call out a “chief engineer” or a sole “systems engineer.”

Program Systems Engineering contains less emphasis on systems analysis than Discovery because the problem space is more precedented (commercial communication satellites, for example), and the unprecedented aspect is how the pieces fit together to provide a new variation on a type of service (new communication frequency, higher power, different stabilization mechanisms, etc.) The systems engineering types called “discovery” and “reduction to practice” in (Dahlberg 1999) match these first two types.

In some cases, Program Systems Engineering is the follow-on to Discovery, which defined the overall problem and showed that a solution is feasible. Program Systems Engineering then creates that system.

The emphasis in Program Systems Engineering is on producing cost-effective solutions that meet quality and schedule criteria. Organizational processes can be defined and improved to standardize this type of systems engineering. Process standards and assessment models such as EIA 632 and EIA/IS 731, while they may apply somewhat to the other two implementations of systems engineering, apply particularly to Program Systems Engineering.

**Type 3. Approach.** The previous two types of systems engineering apply to large, complex systems. Can it truly be said that “systems engineering” is performed on smaller or less complex systems, such as the sub-systems that are integrated by the “systems engineers” of Program Systems Engineering? Yes, and in this case (Approach), systems engineering is implemented as a process or approach.

Approach is the systems engineering that every engineer must perform on the product, including understanding risks, understanding the operational need, clarifying requirements before jumping to a solution, doing at least informal trade studies to make decisions, and so on through the process or focus areas of the chosen capability model. This is the kind of systems engineering that translates to toasters and cell phones. It is different from coordination or management as in Program Systems Engineering—Approach is what the individual engineer does. Approach is the set of lifecycle roles as applied on a small project or as implemented on a small, commercial development project.

Although systems engineering capability models were created with Program Systems Engineering in
mind, all their process areas can be used also in a much smaller way by an individual engineer, whether writing a symposium paper or building a receiver power supply. Systems life cycles can be adapted to apply to any type of engineering that is performed using a systems engineering approach.

This type of systems engineering is most readily implemented in the commercial arena, although large commercial endeavors might implement Program Systems Engineering as well. This is also the type that you can first convince subsystems engineers, such as software engineers, to start using. Eventually, when this type proves meritorious (by increasing communication with customers and customer satisfaction and decreasing interface problems, for example), some of the larger software engineering efforts may institutionalize the implementation of systems engineering for software as Program Systems Engineering. See also (Roe 1996).

Many people call much of Approach just good management or just good engineering. But the systems engineering community is beginning to develop a body of knowledge to help with managing or engineering well, and this body of knowledge goes by the name of “systems engineering.”

Table 1 summarizes these three types and also notes how they affect many of the aspects of systems engineering implied in the questions in the introduction. Note that what seemed insoluble three pages ago can now be scoped out in a few words.

<table>
<thead>
<tr>
<th>Type 1</th>
<th>Type 2</th>
<th>Type 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Discovery</td>
<td>Program Systems Engineering</td>
</tr>
<tr>
<td>Also known as</td>
<td>Specialist systems engineering</td>
<td>Coordinator or generalist systems engineering</td>
</tr>
<tr>
<td></td>
<td>Concept exploration “SYSTEMS engineering”</td>
<td>Technical side of program management “systems ENGINEERING” “Reduction to practice” (Dahlberg 1999)</td>
</tr>
<tr>
<td></td>
<td>“The engineering discipline applied to the science of complexity” (Senglaub 1996)</td>
<td>Concurrent engineering</td>
</tr>
<tr>
<td>Timing</td>
<td>Can be the initial phase of creating a large system</td>
<td>Can follow Discovery in building a large system</td>
</tr>
<tr>
<td>Consists of</td>
<td>Systems analysis and modeling; mission analysis, threat analysis. Unprecedented problems.</td>
<td>Unprecedented ways of putting together largely preceded components to meet a new variation of a known need.</td>
</tr>
<tr>
<td>Examples</td>
<td>Postwar large projects like Atlas, SAGE, even Boston Central Artery/Tunnel, air traffic control, space missions, war fighting capability</td>
<td>Realization of satellites, airplanes, avionics systems, control systems, large information systems, especially those including hardware</td>
</tr>
<tr>
<td>Discovery</td>
<td>Program Systems Engineering</td>
<td>Approach</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>Customers</td>
<td>Generally government, including military</td>
<td>Government, or sometimes commercial</td>
</tr>
<tr>
<td>Precedence</td>
<td>Unprecedented problems, completely new solutions</td>
<td>Precedent problems, largely preceded subsystems, new solutions</td>
</tr>
<tr>
<td>How measured</td>
<td>Progress toward technical problem. Somewhat research-like, so process measures are not very common or very predictive.</td>
<td>Precedent project measures such as in the combined Practical Systems/Software Measurement guidebook.</td>
</tr>
<tr>
<td>How assessed</td>
<td>No capability models address complex analysis well</td>
<td>EIA/IS 731 and predecessor models assume this as the typical type of systems engineering</td>
</tr>
<tr>
<td>Standards</td>
<td>None apply well</td>
<td>EIA 632 and IEEE 1220 (Velman 1999)</td>
</tr>
<tr>
<td>Benefits and Value</td>
<td>Solve problems not otherwise solvable. Generate innovations and revolutionary approaches. (Schoening 2000)</td>
<td>Prevents problems with interfaces and customer dissatisfaction downstream. Prevents test problems and rework. Helps organize work.</td>
</tr>
<tr>
<td>Funding</td>
<td>Systems engineering constitutes the bulk of the program</td>
<td>Often level of effort, sometimes overhead, but increasingly activity-based</td>
</tr>
<tr>
<td>Training</td>
<td>Directly relevant education is critical. Mathematics, electrical engineering, computer science, systems analysis, general engineering, astrophysics (for space systems), domain areas, training in the analysis tools. “System Technology” (Grange 1999) How to make sense out of uncertainty, give form to the formless, and create organizational structures that help promote understanding (Schoening 2000).</td>
<td>Broad variety of engineering courses as well as others (statistics, business, chemistry or physics, even liberal arts with technical minor), organizational dynamics, domain areas. Formerly specialist engineers (especially controls or software) who have broadened their expertise. Much on-the-job training. Organizational processes (budgeting process, change requests, marketing and business, engineering courses outside employee’s specialty), systems engineering processes and process assessment, process improvement. Project and program management and measurement. “Primary systems” (Grange 1999)</td>
</tr>
</tbody>
</table>
1999). Also (Plowman 1999).

<table>
<thead>
<tr>
<th>Tools/Methods</th>
<th>Discovery</th>
<th>Program Systems Engineering</th>
<th>Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Systems analysis and modeling tools and methods, such as behavior diagrams (Alford 1994), dynamic simulators, and complex nests of models. Simple models showing relationships rather than details, including analogies (Schoening 2000).</td>
<td>Office tools (word processing, spreadsheets, briefing tools, etc.) Particularly, those tools called out by the program processes (templates). Meeting tools (especially if development is geographically distributed).</td>
<td>None in particular, although requirements management tools are becoming more popular.</td>
</tr>
<tr>
<td>Primary Roles</td>
<td>System Analyst, Requirements Owner, Information Manager, Technical Manager</td>
<td>System Designer, Coordinator, Technical Manager, Customer Interface, Glue, Validation and Verification, Requirements Owner</td>
<td>Requirements Owner, System Designer, Validation and Verification, Logistics and Operations, also Customer Interface and Technical Manager</td>
</tr>
<tr>
<td>Evolution</td>
<td>Decreasing numbers of huge programs are available. Needs to evolve to more civilian work, but then a major aspect is politics (less important in aerospace/defense programs of the past). As software builds up its store of reusable objects and becomes “object oriented” (as hardware systems have been at least since Henry Ford’s assembly lines), then many of the current Discovery software system projects will become more like Program Systems Engineering.</td>
<td>Pyramids to present day. Probably the need will never change. However, when money gets tight, this type of systems engineering tends to be targeted as “non-value-added” and cut. Need to evolve clear statement of return on investment of systems engineering activities under given conditions.</td>
<td>Should be taught to all engineers. This type is “below the radar” of many managers, so practitioners can suffer lack of recognition. Very bottom-up, so it can be hard to institutionalize. Eventually, when this type proves its worth in new fields (by increasing communication with customer and customer satisfaction and decreasing engineering interface and test problems and rework), the systems engineering effort in some of these may become larger and more formalized, like Program Systems Engineering.</td>
</tr>
</tbody>
</table>

Table 1. Three Types of Systems Engineering Implementation (continued)


**INTERFACES AMONG TYPES OF IMPLEMENTATION**

All three types. 

**Approach** can be used with any kind of engineering and addresses the full life cycle, but when the complexity increases, people tend to split off a separate group called “systems engineers” or “systems engineering.” Two main types of complex situations stand out, the type where the complexity is due to unprecedented problems (Discovery) and the type when the problems are fairly well known but the solutions are very complex (Program Systems Engineering).

**Example.** (Cook, Lawson, and Allison 1999) show a hierarchy of military systems. Many of these “Systems of Systems” (military task forces, “Defence Organisations”) would require Discovery systems engineering, whereas the “Primary Systems” (man-in-the-loop weapon systems, communications networks) would probably require Program Systems Engineering. Lower level systems (buildings, rifles, thermostats) would use Approach systems engineering.

**Example.** (Grange 1999) also shows three types of systems engineering as a continuum.

**Type 1—Type 2.** Clearly Discovery and Program Systems Engineering overlap. The difference is primarily that the focus in the first type is on analysis and problem space, and the focus in the second type is on solution space and getting a product out. However, note that even Discovery implementations eventually end up creating a realized system. It is often obvious to the practitioners when the focus of the program shifts from analyzing the problem to getting something built. This is the boundary between Discovery and Program Systems Engineering in time. Additionally, Program Systems Engineering implementations also have a short Discovery phase when deciding what to propose. Sometimes an organization has “trappers” who specialize in this concept exploration phase and “skinners” who take over to get the system built.

**Example.** In one government agency, a group called Systems Engineering performs analysis-intensive, future-oriented, and big-picture-oriented (Discovery) tasks. Program Systems Engineering is not recognized as such, and is performed by integrated product teams (IPTs) for the various products. However, the engineers on IPTs recognize that they are performing systems engineering and wish the Systems Engineering group would be a bit more helpful to them.

**Example.** (Carlok et al. 1999) describes top level systems-of-systems as requiring political, economic, operational, and technical considerations, in that order. These are Discovery systems. For more typical Program Systems Engineering developments, the order is reversed. This paper also shows three levels of “systems-of-systems” engineering, of which the first two can be mapped to Discovery and the third can be mapped to Program Systems Engineering. The difference between the first two levels is that the first level is primarily political and economic, and the second level is primarily analysis that verifies the technical and operational characteristics.

**Type 2—Type 3.** Program Systems Engineering and Approach both concentrate on the realization of systems. Both types exist on most large programs. Figure 2 shows a typical large-program structure that has a Program Systems Engineering organization at the system level, and systems engineering tasks that are performed in an Approach manner at subsystem and element levels. (Adamsen 1996) shows similar tiers of integrated product teams.

**Example.** Most people who want to apply systems engineering to commercial companies or to software development are thinking Program Systems Engineering, but they would probably be better off starting...
with Approach. See (Jackson 1996).

Type 2—Program management. Program Systems Engineering and program management are closely related. (Kasser 1996) concludes that systems engineering is just management with an extra emphasis on requirements. In addition, the systems engineering capability models disagree whether “integration of disciplines” is in the Engineering (SE-CMM) or Management (EIA/ISO 731) category. INCOSE and the Project Management Institute intend to get together and work out the definitions and interfaces between these two disciplines. The author suspects that the overlap is great and both are solutions to the same need: solid management of interfaces among technical groups.

CONCLUSION

Many questions can be answered and problems can be resolved when systems engineering is viewed as consisting of more than one basic thing. Consider the three types discussed in this paper when you talk about the kind of systems engineering you do or should do. INCOSE technical working groups should address these three types separately, e.g., measurement of Discovery systems engineering, measurement of Program Systems Engineering, measurement of the systems engineering Approach, or assessment of the three types. All work products of INCOSE should address one or more types explicitly. For example, the Systems Engineering Handbook might specify it is for Program Systems Engineering, and/or include tailoring guidance for Approach and perhaps Discovery implementations.

If INCOSE adopts this definition, the future direction will be clarified, as will its relationship to other disciplines such as program management and software.

REFERENCES


Dahlberg, Sten, Message posted to INCOSE list server, October 1999.


BIOGRAPHY

Sarah A. Sheard has twenty years’ experience in systems engineering, including hardware systems such as satellites and software systems such as air traffic control. Currently at the Software Productivity Consortium, Ms. Sheard helps companies activate systems engineering process improvement efforts and integrate the software and systems engineering aspects of both development work and process improvement efforts.


Ms. Sheard is currently the chair of INCOSE’s Measurement Technical Committee and in the past has served INCOSE as chair of the Communications Committee and as Programs Chair of the Washington Metropolitan Area Chapter. Ms. Sheard received a bachelor’s degree in chemistry from the University of Rochester and a master’s degree in chemistry from the California Institute of Technology.