The Art and Science of Systems Engineering*

Michael Ryschkewitsch, *National Aeronautics and Space Administration*
Dawn Schaible, *National Aeronautics and Space Administration*
Wiley Larson, *Stevens Institute of Technology*

The Scope of Systems Engineering
The Personal Characteristics of Good Systems Engineer
Realities of Complex System Design
Processes for Systems Engineers
The Best Preparation
Summary

This work culminates years of experience in systems engineering and focused discussions among NASA leadership, systems engineers, and systems engineering trainers across the Agency. One consistent theme in these experiences and discussions is that NASA uses many definitions and descriptions of systems engineering. We use the terms and job titles of chief engineer, mission systems engineer, systems engineering and integration manager, system architect, vehicle integration, and so on for various pieces of the complete systems engineering function. We need to agree on a common understanding of systems engineering. In addition, no matter how we divide the roles and responsibilities among people, we must ensure that those roles and responsibilities are clear and executed as a functional whole. **Our objectives are to provide a clear definition of systems engineering, describe the highly-effective behavioral characteristics of our best systems engineers and make explicit the expectations of systems engineers at NASA.**

Systems engineering is both an art and a science. We can compare systems engineering to an orchestra and its ability to perform a symphony. Most people understand what music is, but not everyone can play an instrument. Each instrument requires a different level of expertise and skill. Some musicians spend their entire careers mastering a single instrument, which is good because each one needs to be played well. But sophisticated music involves many different instruments played in unison. Depending on how well they come together, they may produce beautiful music or a terrible cacophony.

* Systems engineering is a critical core competency for successful NASA missions. This monograph summarizes the collective wisdom of some of NASA's best technical minds on the subject. So here the word "we" represents all contributors to this effort: Michael Bay, Bill Gerstenmaier, Mike Griffin, Jack Knight, Wiley Larson, Ken Ledbetter, Gentry Lee, Michael Menzel, Brian Muirhead, John Muratore, Bob Ryan, Mike Ryschkewitsch, Dawn Schaible, Chris Scolese, and Chris Williams. Among them, they have more than 390 years—almost four centuries—of experience in aerospace and systems engineering.

We can think of a symphony as a system. The musicians apply the science of music: they follow the process of translating notes on a page to play their instruments. But an orchestra conductor, a maestro, must lead them to connect the process of playing to the art of creating great music. Maestros do a lot more than just keep time! They:

- Know and understand music—such matters as pitch, rhythm, dynamics, and sonic qualities—as well as the capabilities of various instruments and musicians
- Are necessary once the orchestra reaches a certain size and complexity
- Have typically mastered one or more musical instruments
- May be composers
- Select and shape the music that an orchestra plays
- Interpret a composer's music in light of the audience
- Strive to maintain the integrity of the composer's intentions
- Organize and lead the musicians
- Are responsible for the success of the performance

The systems engineer is like the maestro, who knows what the music should sound like (the look and function of a design) and has the skills to lead a team in achieving the desired sound (meeting the system requirements). Systems engineers:

- Understand the fundamentals of mathematics, physics, and other pertinent sciences, as well as the capabilities of various people and disciplines
- Have mastered a technical discipline and learned multiple disciplines
- Must understand the end game and overall objectives of the endeavor
- Create a vision and approach for attaining the objectives
- May be architects or designers
- Select and shape the technical issues to be addressed by multidisciplinary teams
- Must often interpret and communicate objectives, requirements, system architecture, and design
- Are responsible for the design's technical integrity
- Organize and lead multidisciplinary teams
- Are responsible for the successful delivery of a complex product or service

A great systems engineer completely understands and applies the art of **leadership** and has the experience and scar tissue from trying to **earn** the badge of **leader** from his or her team.

Harold Bell
NASA Headquarters

The similarities between maestros and systems engineers are useful in describing the latters' desired behavioral characteristics and capabilities.

Systems engineering is the art and science of developing an operable system that meets requirements within imposed constraints. This definition is independent of scale, but our discussion here focuses on developing complex systems, such as aircraft, spacecraft, power plants, and computer networks.

Systems engineering is holistic and integrative. It incorporates and balances the contributions of structural, mechanical, electrical, software, systems safety, and power engineers, plus many other, to produce a coherent whole. Systems engineering is about tradeoffs and compromises, about generalists rather than specialists.

Systems engineering is not only about the details of requirements and interfaces among subsystems. Such details are important, of course, in the same way that accurate accounting is important to an organization's chief financial officer. But accurate accounting does not distinguish between a good financial plan and a bad one, nor help to make a bad one better. Similarly, accurate control of interfaces and requirements is necessary to good systems engineering, but no amount of care in such matters can make a poor design concept better. **Systems engineering is first and foremost about getting the right design**—and then about maintaining and enhancing its technical integrity, as well as managing complexity with good processes to get the design right. We define interfaces in a system design to minimize unintended interactions and simplify development and operations—and then we document and control the design. Neither the world's greatest design, poorly implemented—nor a poor design, brilliantly implemented—is worth having.

The principles of systems engineering apply at all levels. For example, engineers who are developing an avionics system must practice creative design and interface definition to achieve their goals. Similar activities are essential to the architecture, design, and development of elements and subsystems across the broad spectrum of NASA developments. But for the remainder of this discussion, we use the term "systems engineering" in the context of complex, multidisciplinary system definition, development, and operation.

In his 2007 presentation, "Systems Engineering and the 'Two Cultures' of Engineering," Mike Griffin describes how the complexities of today's aerospace systems and the ways they fail have led to branching within the industry. For our purpose, we divide systems engineering into technical leadership and its ally, systems management.

- Technical leadership focuses on a system's technical design and technical integrity throughout its lifecycle
- Systems management focuses on managing the complexity associated with having many technical disciplines, multiple organizations, and hundreds or thousands of people engaged in a highly technical activity

Once a credible design and architecture are established, the systems engineer's job is to maintain technical integrity throughout the complex system's very rigorous and challenging lifecycle phases.

Robert Ryan, Marshall Space Flight Center

Technical leadership, the *art* of systems engineering, balances broad technical domain knowledge, engineering instinct, problem solving, creativity, leadership, and communication to develop new missions and systems. It is the system's complexity, and severity of its constraints—not just its size—that drives the need for systems engineering.

NASA systems are often large and complex, so they require systems engineers to work in teams and with technical and other professional experts to maintain and enhance the system's technical integrity. The creativity and knowledge of all of the people involved must be brought to bear to achieve success. Thus leadership and communications skills are often as important as technical acumen and creativity. This part of systems engineering is about doing the job right.

For large complex systems, there are literally millions of ways to fail to meet objectives, even after we have defined the "right system." It is crucial to work all the details completely and consistently and ensure that the designs and technical activities of all the people and organizations remain coordinated—art is not enough.

Systems management is the *science* of systems engineering. Its focus is on rigorously and efficiently managing the development and operation of complex systems. Effective systems management requires applying a systematic, disciplined engineering approach that is quantifiable, recursive, repeatable, and demonstrable. Here the emphasis is on organizational skills, processes, and persistence. Process definition and control are essential to effective, efficient, and consistent implementation. They demand a clear understanding and communication of the objectives, and vigilance in making sure that all tasks directly support the objectives.

Systems management provides a framework for problem solving...creative problem solving for complex systems.

Dinesh Verma,
Stevens Institute of Technology

One of the biggest challenges for a systems engineer of a large complex project is to "bring order from chaos."

Chris Hardcastle,
Systems Engineering and Integration Manager, NASA's Constellation Program, Johnson Space Center

Systems management applies to developing, operating, and maintaining integrated systems throughout a project or program's lifecycle, which may extend for decades. Since the lifecycle may exceed the memory of the individuals involved in the development, it is critical to document the essential information.

To succeed, we must blend technical leadership and systems management into complete systems engineering. Anything less results in systems not worth having or that fail to function or perform.

The Scope of Systems Engineering

Since the late 1980's, many aerospace-related government and industry organizations have moved from a hard-core, technical leadership culture (the art) to one of systems management (the science). History has shown that many projects dominated by only one of these cultures suffer significant ill consequences. Organizations that focus mainly on systems management often create products that fail to meet stakeholder objectives or are not cost effective. The process often becomes an end unto itself, and we experience "process paralysis." Organizations that focus solely on technical issues often create products or services that are inoperable, or suffer from lack of coordination and become too expensive or belated to be useful.

To achieve mission success, we must identify and develop systems engineers that are highly competent in both technical leadership and systems management.

That is why we focus on the complete systems engineer, who embodies the art **and** science of systems engineering across all phases of aerospace missions—a type reflected in Figure 1. In any project, it is critical that systems engineering be performed well during all lifecycle phases. The scope of systems engineering and the associated roles and responsibilities of a systems engineer on a project are often negotiated by the project manager and the systems engineer. The scope of systems engineering and the activities for which the systems engineer is both responsible and accountable should be understood and documented early in the project.

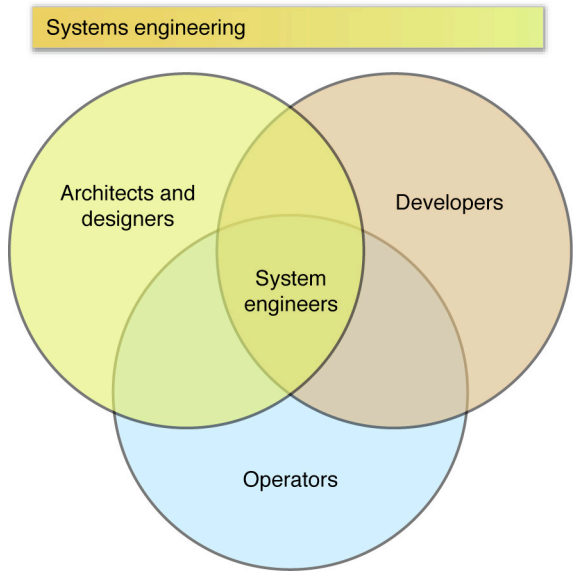

Figure 1. The Scope of Systems Engineering. Systems engineers often focus on one lifecycle phase like architecture and design versus development or operations, but good systems engineers have knowledge of and experience in all phases.

Here we describe the characteristics, some innate and others that we can develop, that enable select people to "systems engineer" complex aerospace missions and systems—to design, develop, and operate them. Then, we focus on how to further develop NASA's systems engineers to help them deal better with the complexities of sophisticated missions and systems.

The Personal Characteristics of Good Systems Engineers

Figure 2 depicts the personal behavioral characteristics of effective systems engineers.

Intellectual curiosity. Perhaps the most important personal characteristic of successful systems engineers is *intellectual curiosity*. People who prefer boundaries around their work to be comfortable, know what they know, and enjoy a focused domain may want to consider another occupation. Systems engineers continually try to understand the what, why, and how of their jobs, as well as other disciplines and situations that other people face. They are always encountering new technologies, ideas, and challenges, so they must feel comfortable with perpetual learning.

> People who have "systems engineer" in their title, regardless of the modifiers—project, program, flight system, and so on—are responsible for everything.
>
> *Gentry Lee, Jet Propulsion Laboratory*

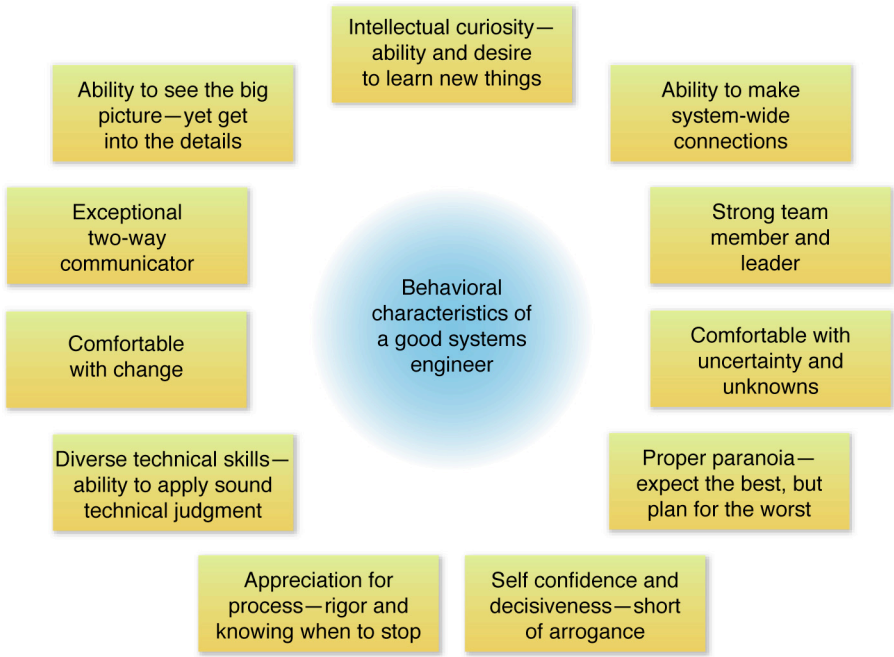

Figure 2. Characteristics of a Good Systems Engineer. The characteristics are shown in decreasing priority from top to bottom. Some of them are innate, whereas others can be learned and honed.

Ability to see the big picture. Good systems engineers maintain a *big-picture perspective*. They understand that their role, though always significant, changes throughout a project's lifecycle. At any point in the lifecycle the systems engineer must be fully cognizant of what has been done, what is necessary, and what remains to be done. Each phase has a different emphasis:

- Concept—mission and systems architecture, design, concept of operations, and trade studies
- Development—maintaining technical integrity throughout all lifecycle phases: preliminary design review, critical design review, verification, validation, and launch
- Operations—making sure that the project meets mission requirements and maintains technical integrity

Systems engineers pay particular attention to verification and validation. Verification answers the question: "Did we build our system right?" If we are successful, it proves our product meets the requirements. We emphasize the hard-earned lesson, "Test like you fly, fly like you test." Validation, on the other hand, answers the question: "Did we build the right system?" If we are successful, the system does what it is supposed to do, which often goes well beyond just meeting requirements!

Good systems engineers are able to "translate" for scientists, developers, operators, and other stakeholders. For example, "Discover and understand the relationship between newborn stars and cores of molecular clouds," is meaningful to a scientist. But developers and operators would better understand and use this version: "Observe 1,000 stars over two years, with a repeat cycle of once every five months, using each of the four payload instruments." The systems engineer that knows the project's objectives, helps determine how to meet them, and maintains the system's technical integrity throughout its lifecycle has a good chance of succeeding. A corollary is to check everyone's understanding of each other to make sure the team truly IS on the same page.

Ability to make system-wide connections. First-rate systems engineers understand the *connections* among all elements of a mission or system. They must often help individuals on the team see how their systems and related decisions connect to the bigger picture and affect mission success. The Chandra X-ray Observatory offers a practical example of these connections. The star tracker's designer must understand that the star tracker is part of an attitude control system—specifically, of an attitude estimator used to take precisely pointed observations—and that the star tracker's output determines whether or not the proper images are obtained. If the designer does not understand this, the project is in trouble. Good systems engineers can anticipate the impact of any change injected into the system or project, and describe the nature and magnitude of the impact throughout their system.

Exceptional two-way communicator. *Communications skills* are the great enabler. Systems engineers need to be able to get out of their offices and communicate well—listen, talk, and write. George Bernard Shaw once stated that England and America are "two countries separated by a common language," but engineers are separated by their **separate** languages—even more so since the advent of electronic communications. Systems engineering helps bridge the communication gaps among engineers and managers with consistent terms,

processes, and procedures. A key to success is the ability to see, understand, and communicate the big picture, and be effective in helping others develop a big-picture view.

Strong team member and leader. Here we distinguish between management and leadership, realizing that a systems engineer must be skilled in both.

So far, we have described the characteristics that good systems engineers share. Ideally, as they gain experience, they are able to deal with more complex systems through

- Breadth of technical knowledge and expertise, combined with execution excellence
- Passion for the mission and challenges, combined with force of personality and leadership ability
- Creativity and engineering instinct —ability to sense the right way to attack a problem while appreciating inherent risks and implications
- Ability to teach and influence others

While management and leadership are related and often treated as the same, their central functions are different. Managers clearly provide some leadership, and leaders obviously perform some management. However, there are unique functions performed by leaders that are not performed by managers. My observation over the past forty years...is that we develop a lot of good managers, but very few leaders. Let me explain the difference in functions they perform.

- A manager takes care of where you are; a leader takes you to a new place
- A manager is concerned with doing things right; a leader is concerned with doing the right things
- A manager deals with complexity; a leader deals with uncertainty
- A manager creates policies; a leader establishes principles
- A manager sees and hears what is going on; a leader hears when there is no sound and sees when there is no light
- A manager finds answers and solutions; a leader formulates the questions and identifies the problems

James E. Colvard

Comfortable with change. Systems engineers should be *comfortable with change*. They understand that change is inevitable. They anticipate change, are able to understand how it affects their systems, and deal with those effects properly, usually without losing sleep at night.

The number of changes must decrease with time. If projects continue to change, they will never get to the launch pad. This is particularly true with requirements. While it is undesirable to freeze them too early, it is much more likely that requirements will continue to change way too long. ...At some point, the design must be implemented, at which time "change" is the enemy.

Ken Ledbetter, NASA Headquarters

Comfortable with uncertainty. A companion characteristic is being *comfortable with uncertainty*—indeed, embracing uncertainty. We usually do not know when we will finish a task, or even a mission. We know requirements are not complete, so we have to interpret them. This is the simple side of uncertainty. But uncertainty has a more complex side, so a strong background in probability and statistics is important. A good systems engineer understands and encourages quantification of uncertainty. For example, if the mission objective is to land a probe on a comet, the location and severity of jets or debris may be unknown or the comet's albedo may be uncertain. The systems engineer must be able to work with a team to design a system that accommodates the uncertainties.

Proper paranoia. Another important characteristic is *proper paranoia*: expecting the best, but thinking about and planning for the worst. This suggests that the systems engineer is constantly checking and crosschecking selected details across the system to be sure that technical integrity is intact.

Diverse technical skills. A systems engineer must be able to apply sound technical principles across *diverse technical disciplines*. Good systems engineers know the theory and practice of many technical disciplines, respect expert input, and can credibly interact with most discipline experts. They also have enough demonstrated engineering maturity to delve into and learn new technical areas that should be integrated into the system. They must be strong *technical leaders*, in addition to having broad technical competence. Systems engineers must meet the special challenge of commanding diverse technical knowledge, plus managing, **and** leading effectively!

Self confidence and decisiveness. Systems engineers must have well-earned *self-confidence*. They know what they know and are aware of what they do not know, and are not afraid to own both. It does not mean systems engineers never make mistakes. We have all made mistakes...at least occasionally.

Commission, not omission. This should be written on the door of every systems engineer. There is no excuse for omission. A systems engineer does not need authority from anyone to investigate anything. The systems engineer's job is the whole space. You go out, you make decisions. If someone tells you to stop, you use your communication skills and listen.

Gentry Lee, Jet Propulsion Laboratory

Appreciate the value of process. Good systems engineers *appreciate process*. That does not mean systems engineering is just one process, plus another, plus another—like recipes in a cookbook. Let us look back at our metaphor. To create the music of a symphony, musicians use their instruments, musical scores, and notes. These tools provide them with a common frame of reference, help them keep proper time, and allow the orchestra to work together to create beautiful music. Processes serve the same purpose for the systems engineer. But just providing sheets of music to a group of musicians does not guarantee a great orchestra. While each orchestra uses the same tools and many have very skilled musicians, they do not all sound like the New York Philharmonic.

Herein lies the art—how well does the maestro lead the people and use the tools provided? Maestros know how to bring out the best in their musicians; they know how to vary the tempo and the right moment to cue the horn section to draw in the listeners. The same is true for systems engineers. We must all use processes to get the job done, but it is what we DO with the processes and talents of the team that matters.

A successful systems engineer knows how to balance the art of technical leadership with the science of systems management. Both are required for success! The behavioral characteristics described above are necessary to meet the many challenges facing NASA's systems engineers today and in the future.

Realities of Complex System Design

To this point, we have defined systems engineering as a combination of technical leadership and systems management. We have established that highly effective systems engineers share certain behavioral characteristics. These elements feed into successful mission and system design: the ability to get a system's design

right initially and to maintain its technical integrity throughout its lifecycle. There are numerous definitions of architecture and design. We use the following:

- Architecture encompasses the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution
- Design is creating a product or system, as well as a plan to develop and use it. For our purpose, architects provide the rules; designers create implementations using those rules.
- **Systems engineers do both**—help create the design and maintain its integrity throughout its lifecycle

Designing new aerospace missions and systems is a very creative, technical activity. Most engineers use a variation of a fundamental thought process—**D**efine the problem, **E**stablish selection criteria, **S**ynthesize alternatives, **A**nalyze alternatives, **C**ompare alternatives, **M**ake a decision, and **I**mplement (and Iterate, for that matter). Though not usually mandated, this process, or one very much like it, is applied because it produces good, useful results—it works! For shorthand, we will refer to this as DESACMI.

Tom Logsdon, a highly respected author of several books and thought-provoking papers in space systems, describes this activity in his book, *Six Simple, Creative Solutions That Shook the World*. He eloquently explains six steps that capture the creative approaches of highly creative people:

- Break the problem apart and put it back together again
- Take a fresh look at the interfaces
- Reformulate the problem
- Visualize a fruitful analogy
- Search for useful order-of-magnitude changes
- Be alert to happy serendipity

This book is worth its weight in gold—one reason it was renamed *The Midas Touch*!

The first credible design for a space mission and its associated systems is usually the product of a few individuals or a small design team. They

- Capture and validate stakeholders' needs and success criteria
- Identify critical top-level requirements (normally 3 to 7) and understand the acceptance criteria
- Create a mission concept as well as physical and functional architectures
- Develop a concept of operations and integrate it with the mission concept, architecture, and top-level requirements
- Design critical interfaces among the architecture's elements
- Develop clear and unambiguous requirements that derive from the mission concept, architecture, concept of operations, and defined interfaces

The result of this intense, highly iterative, and creative activity is a first credible design that is consistent with basic physics or engineering principles and meets top-level requirements. It is a baseline from which we apply systems management processes to do tradeoffs and more detailed quantitative analyses that focus on enhancing the design detail. We also continue to identify and mitigate technical, cost, and schedule risks.

Defining the interfaces is key. We have to keep the number of interfaces to an acceptable minimum; less is usually more provided the appropriate level of isolation of interaction is maintained. We should also keep them as simple as possible and, when confronted with a particularly difficult interface, try changing its characteristics. And of course, we have to watch out for Murphy's Law!

Designers and systems engineers engaged in this early activity, and indeed, throughout the lifecycle, follow several hard-learned principles: apply equal sweat, maintain healthy tension, manage margin, look for gaps and overlaps, produce a robust design, and study unintended consequences.

Eberhard Rechtin partly captures the essence of a first credible design of a space mission and its systems, as well as more refined or evolved designs, in System Architecting (which some would call system design):

- Relationships among the elements are what give systems their added value.
- Choosing the appropriate aggregation of functions is critical in the design of systems...In partitioning, choose the elements so that they are as independent as possible, that is, elements with low external complexity and high internal complexity...choose a configuration in which local activity is high speed and global activity is slow change...minimal communications between subsystems.
- The greatest leverage in system architecting and design is at the interfaces.

Apply equal sweat. The concept of *equal sweat* is to apportion the required performance or functional requirements, as much as possible, so that no single subsystem has an insurmountable problem. Figure 3 provides an example of the allocation and flow down of the top-level requirement for mapping error. If the mapping error is misallocated, it can easily drive the cost and complexity of one element up significantly, while allowing another element to launch with excess margin. Good engineering judgment and communication are required to allocate a requirement across subsystem boundaries in such a way that each element expends "equal sweat" in meeting the requirement. We must be prepared to reallocate when a problem becomes unexpectedly difficult in one area. To achieve this, team leaders must have established open communications and the expectation that the members can and should raise issues.

The concept of equal sweat applies to many aspects of space systems: spacecraft—pointing stability and knowledge; payload—line-of-sight, optical, thermal, and structural stability; operations—command complexity and recorder management; communications—total data volume and latency; data processing—artifact removal, data integrity verification, throughput, and reprocessing; and dissemination—metadata and archive management, to name a few.

Maintain healthy tension. A project must simultaneously meet its cost, schedule, and technical objectives. This often creates conflict. How much should we spend making the system better and how good is good enough? How much time and money must we spend in chasing down a problem? What risk will we take if we eliminate a test and how well do we understand that risk? Like the United States constitutional system, NASA's system of checks and balances is designed to ensure balancing of these objectives. If engineering becomes too focused on creating the perfect system, project management must push on cost and schedule. If project management becomes too focused on minimizing testing to reduce schedule, engineering or safety and mission assurance must push on the technical integrity. Discussions may become extremely complex and passionate but we need to keep the common goal of mission success at the forefront.

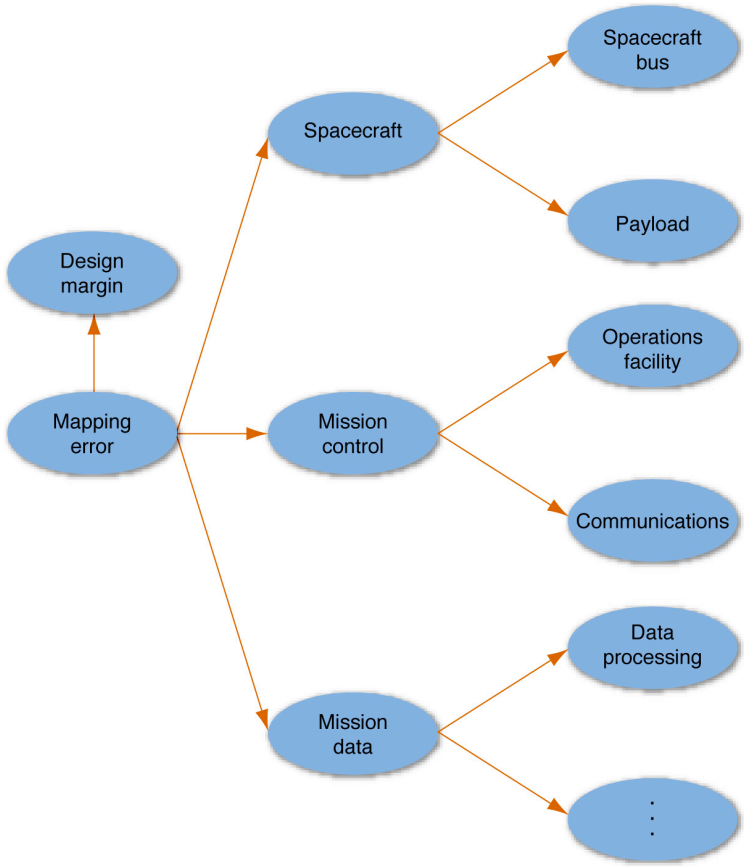

Figure 3. **An Example of Equal Sweat.** Here we see a potential mapping error allocation for a space system. Mapping error represents how well the system is expected to pinpoint the location of an image created by the system. Zero mapping error is perfection. The goal, after setting sufficient design margin aside, is to allocate the mapping error to elements of the system in such a way that no element has an insurmountable challenge and each element expends roughly equal effort (sweat) in meeting the mapping error requirement.

Constructive dialogue among respectful peers along with the timely elevation of impasses is critical. It is vital to allow sufficient time for productive discussion while making timely decisions to move forward. The interactions may be individually stressful and appear to be wasteful, but NASA has a long history of mission successes when we maintained healthy tensions among all of the parties and conversely, a number of major failures when we did not.

Similar healthy tension occurs in many areas—across organizations, across subsystems or elements, and between mission phase requirements. This tension plays out during the design phase, when the operators try to be sure that the system will be operable and maintainable while the designers work to balance significant nearer term constraints such as cost, schedule, or mass. It also plays out

during development and operations, when teams balance design changes and workarounds with ensuring safe, successful systems. Throughout the lifecycle this continual tension helps maintain the proper requirements, constraints, and testing. For example, we must strike a balance between too little and too much testing. Not enough testing adds risk to a program, whereas testing too much can be very costly and may add unnecessary run-time to the equipment. These healthy tensions are a key to creating and maintaining the environment that will produce the best-balanced system, and the systems engineer must embrace and foster them.

Manage margin. Good systems engineers maintain a running score of the product's resources: power, mass, delta-V, and many others. But more importantly, they **know the margins**. What exactly does margin mean? Margin is the difference between requirements and capability. If a spacecraft must do something, we allocate requirements. One way to add margin is to make the requirements a little tougher than absolutely necessary to meet the mission's level-one requirements, which some people call contingency. If we meet requirements, test effectively, and do the job correctly, we create a capability.

In Figure 4, the outer shape defines the capability, the inner shape represents the requirement, and the space between the two represents margin. The requirement comes very close to the capability (right side of the diagram), so we have a minimum margin.

> It is not sufficient, systems engineers, to simply know the requirements and say, "Look we met our requirements." We must go beyond and be able to understand and articulate how much margin we have available in any situation. This gets us back to knowing the partial derivative of everything with respect to everything!
>
> *Gentry Lee, Jet Propulsion Laboratory*

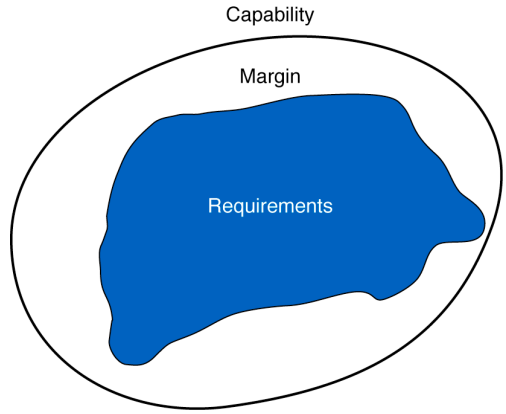

Figure 4. Requirements, Capability, and Margin. Where the requirements come close to the capability (as on the right side of the figure), we have little margin.

Look for gaps and overlaps. Once we begin feeling comfortable and confident about our design, looking for gaps and overlaps will help us recover from our comfort and confidence. What have we forgotten? Which requirements are incomplete? Where are the disconnects among our project's top-level requirements, architecture, design, and concept of operations? We also must carefully consider all

system interfaces and look on the other side of these interfaces to identify what could interfere with our system. When we do this we often find that our system of interest or the design's scope is not sufficiently defined.

Create a robust design. Robust design is a proven development philosophy focused on improving reliability of systems, products, and services. Other terms used to describe robust design include resilient, stable, flexible, and fault tolerant. Robust design is a key to successful missions and systems. To be useful, however, it must be an early and integral part of development. Our objective is to make our systems immune to factors that could harm performance and mission success. A robust design performs consistently as intended throughout its lifecycle, under a wide range of conditions and outside influences, and it resists unimaginable events. In other words, a robust design provides stability in the presence of ignorance!

Study unintended consequences. A key to our success in spaceflight is that we rigorously analyze failure modes and effects to determine how the system will perform when individual elements, subsystems, and, components fail. Good systems engineers study failures of complex systems, to gain insights into their root causes, ripple effects, and contributing factors. Hardware, software, interfaces, organizations, and people introduce complexity, so we study failures to avoid them. Henry Petroski, a professor at Duke University and author of *Success Through Failure*, points out that studying failures helps us better assess our design's unintended consequences. So the systems engineer should study as many failures as possible to develop good engineering judgment.

In *Apollo: The Race to the Moon*, Murray and Cox offer a stirring account of the Apollo 13 oxygen tank's explosion—a significant anomaly that resulted in mission failure. It shows how flight and ground crews creatively worked together to save the lives of the astronauts.

Two examples of robust design.

On Apollo 13, diverse systems in the lunar module (LM) allowed the crew to survive the transit to the Moon and back after the explosion. Then the manual attitude control capability, guided by the crew looking out the window and firing the command module reentry control system, allowed steering of the burns necessary to return to Earth. Using the LM as a lifeboat was considered during design, and contingency procedures were written before flight.

In contrast is the near loss of the space station when all of the triplex redundant computer systems necessary for attitude control failed. There was no simple backup mode on the ISS. Luckily, the Shuttle was docked at the time, providing a diverse attitude control capability that allowed time and opportunity for the crew to find the problem, bypass it with jumper cables, and restore nominal attitude control. Had the Shuttle not been there, it might have been necessary to abandon the station to allow the crew to escape via the Soyuz before the station tumbled out of control.

Mike Bay, Goddard Space Flight Center

By this point we have discussed the philosophy of mission and system design, reviewed hard-earned wisdom about design, and even applied what we have learned from previous failures to create our "first credible" design. We may hesitate to show the design to others until we have enhanced it a little more, a little more, and even a little more. In other words, it is hard to stop tweaking our design to make it better. Eventually, because of such realities as lack of money or time, we have to say: "Better is the enemy of good enough."

In universities, engineers learn to optimize designs, especially in the traditional electrical and mechanical disciplines. Typically, in a large, complex system design, competing requirements and constraints make optimized subsystems inappropriate. We need a balanced design that meets stakeholder

In this excerpt from *Apollo: The Race to the Moon*, Murray and Cox offer a stirring account of the Apollo 13 oxygen tank's explosion.

> ...In the end, NASA would find, this is what happened...In October 1968, when O_2 Tank 2 used in Apollo 13 was at North American, it was dropped. It was only a two-inch drop, and no one could detect any damage, but it seems likely that the jolt loosened the "fill tube" which put liquid oxygen into the tank.
>
> In March 1970, three weeks before the flight, Apollo 13 underwent its Countdown Demonstration Test...which involved loading all the cryos. When the test was over, O_2Tank 2 was still 92 percent full, and it wouldn't de-tank normally—probably because of the loose fill tube. Because a problem in the fill tube would have no effect on the tank's operation during flight, the malfunction was not thought to be relevant to flight safety.
>
> After three unsuccessful attempts to empty the tank, it was decided to boil off the oxygen by using the internal heater and fan. This was considered to be the best procedure because it reproduced the way the system would work during flight: heating the liquid oxygen, raising its pressure, converting it to a gas, and expelling it through the valves and pipes into the fuel cells where, in flight, it would react with the hydrogen. So they turned on the tank's heater.
>
> A technician working the night shift on Pad 39A was assigned to keep an eye on the tank temperature gauge and make sure that it did not go over 85 degrees Fahrenheit. It was not really necessary that a human serve this function, because a safety switch inside the tank would cut off the heaters if the temperature went beyond the safety limit. And, in reality, the safety margin built into the system meant that the temperatures could go considerably higher than 85 degrees without doing any damage. But the precautions were part of NASA's way of ensuring that nothing would go wrong.
>
> After some time, the technician noticed that the temperature had risen to 85 degrees, but all he had been told was that anything in excess of 85 degrees was a problem, so he let the heater run—about eight hours, in all. No one had told him that the gauge's limit was 85 degrees. That's as high as it could measure. Thus the technician could not tell that the temperatures inside the tank were actually rising toward a peak of approximately 1,000 degrees Fahrenheit, because the safety switch had failed.
>
> It had failed because of one small but crucial lapse in communication. Eight years earlier, in 1962, North American had awarded Beech Aircraft a subcontract to build the cryo tanks for the service module. The subcontract specified that the assembly was to use 28-volt D.C. power. Beech Aircraft in turn gave a small switch manufacturer a subcontract to supply the thermostatic safety switches, similarly specifying 28 volts. In 1965, North American instructed Beech to change the tank so that it could use a 65-volt D.C. power supply, the type that would be used at KSC during checkout. Beech did so, neglecting, however, to inform its subcontractor to change the power specification for their thermostatic safety switches. No one from Beech, North American, or NASA ever noticed this omission.
>
> On all the Apollo flights up through 12, the switches had not had to open. When the tanks were pressurized with cryogens hundreds of degrees below zero, the switches remained cool and closed. When, for the first time in the history of the cryo tanks, the temperature in the tanks rose high enough to trigger the switch—as O_2 Tank 2 emptied—the switch was instantaneously fused shut by the 65-volt surge of power that it had not been designed to handle. For the eight hours that the heaters remained on, the Teflon insulation on the wires inside the cryo tank baked and cracked open, exposing bare wires.
>
> On the evening of April 13...[when the cryo tank was stirred], some minute shift in the position of two of those bare wires resulted in an electrical short circuit, which in turn ignited the Teflon, heating the liquid oxygen. About sixteen seconds later, the pressure in the O_2 Tank 2 began to rise. The Teflon materials burned up toward the dome of the tank, where a larger amount of Teflon was concentrated, and the fire within the tank, fed by the liquid oxygen it was heating, grew fierce. In the final four seconds of this sequence, the pressure exceeded the limits of the tank in about eleven microseconds, slamming shut the reactant valves on Fuel Cell 1 and Fuel Cell 3. Then the Teflon insulation between the inner and outer shells of the tank caught fire, as did the Mylar lining in the interior of the service module. The resulting gases blew out one of the panels in the service module. That explosion also probably broke a small line that fed a pressure sensor on the outside of the O_2 Tank 1, opening a small leak.
>
> Once the service module panel blew out...

This true story is rich in content relevant to systems engineering. It includes examples of design, changes in requirements and their impact, technical data management, overall technical integrity, communications and decisions, transitions (from one contract to another, for example), interfaces, anomaly response, and processes, to name a few. It is worth taking a moment to think about the complexities of technical design, organization, and process that this single example demonstrates. Our job as systems engineers is to manage this complexity!

needs as well as top-level critical requirements and constraints. However, system constraints such as mass often require the overall system to be optimized.

Design is the creative act of managing constraints, organizing system complexity, developing effective interfaces, managing resources and margin, and injecting leading-edge technology when and where appropriate. Creating an architecture and design for a complex system often requires the use of proven processes to manage complexity.

Processes for Systems Engineers

Getting the design right is a key first step in creating, developing, and operating aerospace systems. This activity represents the 10-percent inspiration associated with an acceptable solution. Maintaining technical integrity and managing complexity using good solid processes represents the 90-percent perspiration necessary to deliver the needed products and services. No matter how brilliant the design, we must still understand and properly apply rigorous processes and procedures throughout the project's lifecycle. Otherwise, what seemed like the right design will have a high probability of failing to meet its intended mission, within cost and schedule. Systems engineers must be able to deal with broad technical issues and apply rigorous processes and procedures especially as projects get larger and more complex.

Clarence "Kelly" Johnson, the legendary aerospace engineer from Lockheed, recognized this fact regarding the second generation of the U-2. The celebrated SR-71 reconnaissance aircraft had replaced the U-2 in the late 1960's. Johnson understood that many hard lessons learned while developing the U-2 might be needed in the future. When the U-2 was initially taken out of service, Johnson persuaded his management to preserve and archive, at great expense, the extensive database of processes, procedures, and tooling used to build the original U-2.

That decision became prophetic. In 1979, many years after ending production of this outstanding aircraft, the mission was renewed and enhanced. First flown in 1955, the U-2 has had many variants performing a range of missions.

NASA has documented its systems-engineering policy in NPR-7123.1a, *NASA Systems Engineering*. This document was developed by researching the defense and commercial aerospace industry and building the best approach to bridging the communication gaps identified from many project reviews, anomaly reports, failure reports, and studies.

From experience we know that technical teams tend to ignore policy documents ("cookbooks") that dictate what they must do if the documents are not tailored to project circumstances and supported by complementary elements. Examples of these elements are "how-to" resources (such as SP-6105 (2007) and *Applied Space Systems Engineering*) education and training materials, on-the-job learning activities, and appropriate tools.

In summary, we need to preserve lessons learned and tried-and-true processes, as well as enhance communication and consistently apply processes and procedures throughout NASA. Solid processes enable good systems engineering of complex systems. The advice we offer systems engineers is to own the processes and tools, and know when and how to use them, but not to be owned by them. A lack of process rigor can easily lead to disaster, but too much process

rigor can lead to rigor mortis. So our challenge is to develop systems engineers with good engineering judgment who know how to take a balanced approach. The goal of a process is to provide the needed product or service!

Because systems engineering is both an art and a science, many of the skills and abilities needed to be highly effective in complex systems are not learned in school; they are gained through experience. Processes and tools are very important, but they can not substitute for capable people. Following processes and using tool sets will not result automatically in a good systems engineer or system design. Entering requirements into a database does not make them the right requirements. Having the spacecraft design in a computer-aided design (CAD) system does not make it the right design. Capable and well-prepared people make the difference between success and failure.

The Best Preparation

So how do aspiring systems engineers prepare themselves for such a challenge? Abraham Lincoln had it right when he said, "If I had eight hours to chop down a tree, I'd spend six hours sharpening my ax." But what is the best way to sharpen our axes?

We recommend that if you are an aspiring systems engineer, you embrace the personal behavioral characteristics described above and become proficient in APPEL's systems engineering competencies.* Then, decide what capabilities and experiences you have and what you still need to develop. Once you have assessed your capabilities, the best (some would say the ONLY) way to learn systems engineering is by doing it! It is a good idea to strive for assignments that allow one to develop, deliver, or operate hardware and software systems. Mike Griffin recommends participating in as many "hanger-flying" sessions as possible, spending time with experienced systems engineers and learning from their stories. Everyone is different, and there are many ways to proceed, but a career in systems engineering may unfold as shown in Figure 5.

Figure 6 demonstrates that, as systems engineers develop, they are better able to affect a project's success, even while dealing with increasing complexity. It also points out that the tried-and-true approach to cultivating systems engineering talent is through apprenticeship, mentoring, and on-the-job training—augmented by formal training and education. The best way to become a good systems engineer is to "get your hands dirty" with hands-on experience in multiple disciplines.

NASA has very robust education and training for program or project managers and systems engineers. With many courses and development opportunities, NASA employees have an excellent foundation from which to acquire the knowledge and information they need to improve these capabilities.

* URL's containing additional information for this section can be found in the bibliography under Links.

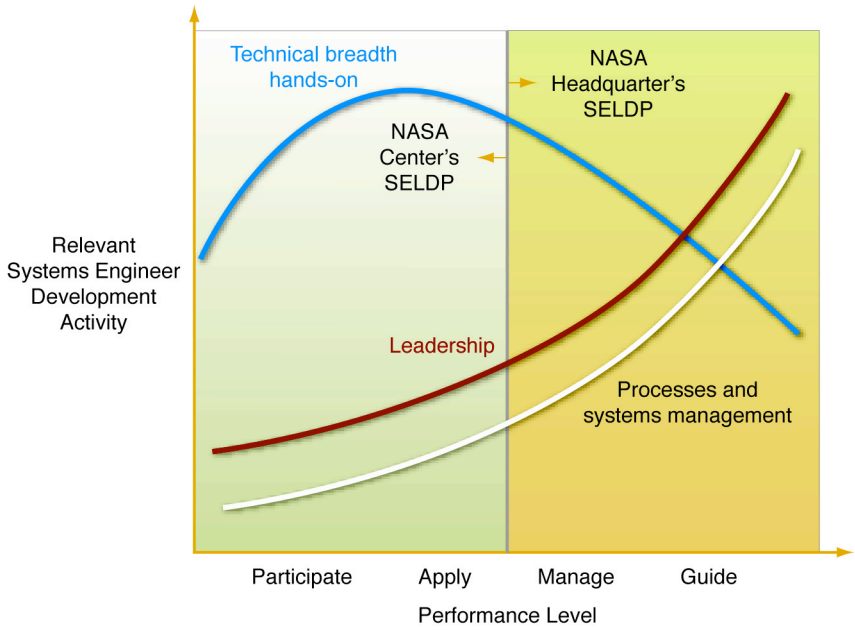

Figure 5. Development Activities for Systems Engineers by Performance Level. Systems engineers develop in three ways: technical breadth, systems management, and leadership. Early in a systems engineer's career—at the first two performance levels (participate and apply), the focus is mainly on developing technical breadth. That breadth develops through hands-on hardware and software experience, as well as growing abilities in systems management and leadership. Systems engineers usually need much greater leadership and systems-management skills later in their careers in the next two performance levels (manage and guide). SELDP is the Systems Engineering Leadership Development Program.

Who in NASA is responsible for a system engineer's development? The supervisor? The NASA Administrator? The Academy of Program, Project and Engineering Leadership? No. Each person is responsible for his or her own development. Figure 7 provides options and guidance. Many people are willing to be coaches and mentors, and to offer opportunities, but it is up to individuals to plan their own professional development.

The Systems Engineering Leadership Development Program (SELDP) provides development activities, training, and education through APPEL. The rationale for the "participate" and "apply" performance levels is provided in NASA's *Integrated Technical Workforce Career Development Model*—created by the Office of the Chief Engineer (OCE) under the leadership of Bob Menrad from Goddard Space Flight Center. The development activities, education, and training for the "manage" and "guide" performance levels derive from two sources: *Coaching Valuable Systems Engineering Behaviors* by M. E. Derro and P. A. Jansma from the Jet Propulsion Laboratory and *NASA's Systems Engineering Behavior*

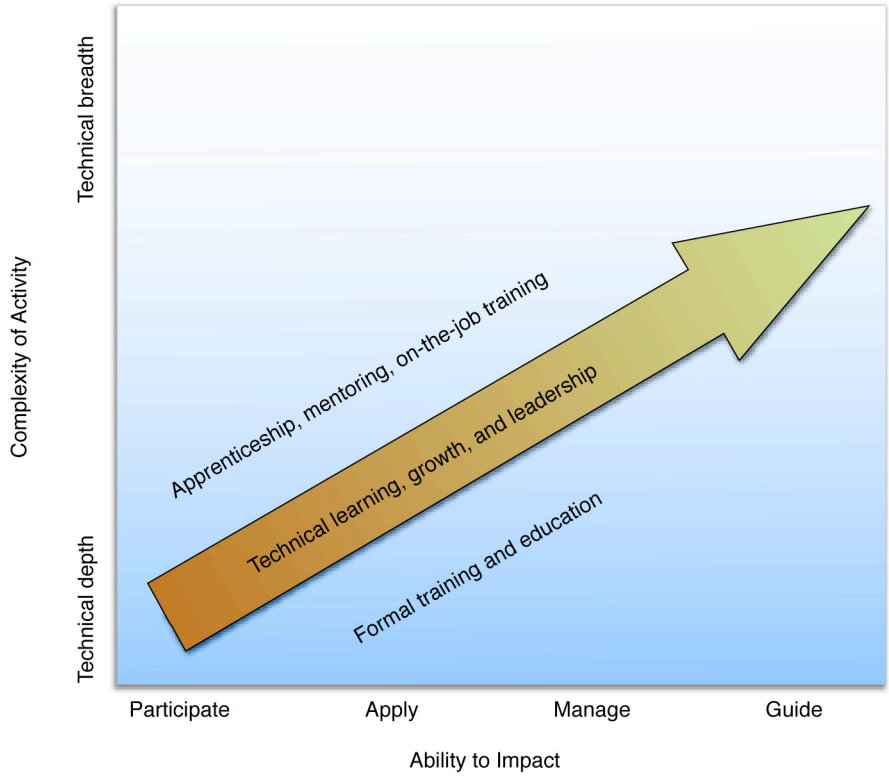

Figure 6. Another Perspective on Development Activities for Systems Engineers. As systems engineers progress from one performance level to the next, they undertake increasingly complex activities and learn that they can more strongly affect how these activities play out. The tried-and-true method of developing systems engineers is through apprenticeship, mentoring, and on-the-job training—augmented as necessary by formal training and education.

Study, by Christine Williams and Mary-Ellen Derro. NASA has identified five main behaviors for success that are assessed and coached as part of the Systems Engineering Leadership Development Program:

- Leadership skills—Demonstrates ability to influence others, work with a team, develop trusting relationships, communicate vision and technical approach, and mentor and coach less-experienced systems engineers
- Attitudes and attributes—Exhibits intellectual self-confidence, intellectual curiosity, and ability to manage change, remain objective, and maintain healthy skepticism

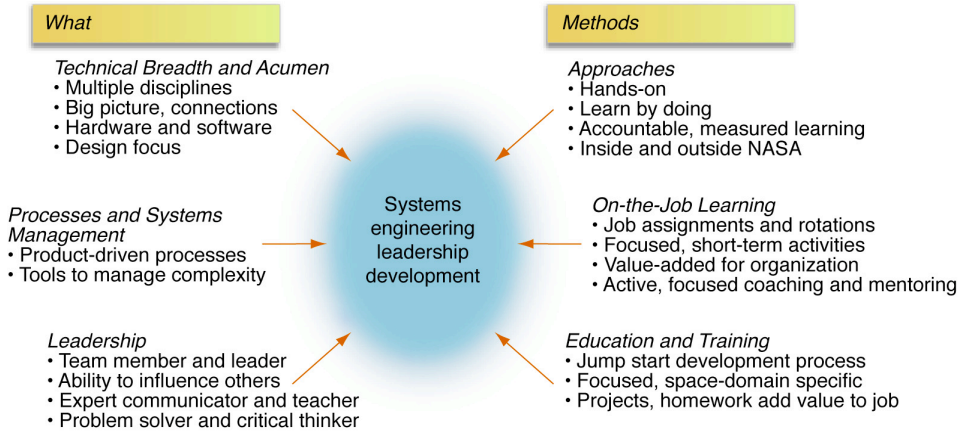

Figure 7. Options for Developing Leadership in Systems Engineering. This figure helps focus system-engineering development activities and identifies the most beneficial approaches.

- Communication—Successfully advances ideas and fosters two-way discussions, communicates through storytelling and analogies, listens and translates information
- Problem solving and critical thinking—Manages risk, thinks critically, and penetrates issues in a logical manner
- Technical acumen—Demonstrates technical breadth and competency, ability to learn new technology, applies experience and lessons learned from successes and failures

Summary

Systems engineering at NASA, and throughout the aerospace industry, is an art and a science, so anyone with the title "systems engineer" must be able to handle both technical leadership and systems management. In fact, both are critical to maintaining technical integrity throughout the development and operation of space systems. We place high value on technically competent systems engineers with diverse technical skills who are highly effective in leading teams and managing systems.

We discussed the personal characteristics of many of NASA's best systems engineers to help aspiring systems engineers better understand the nature of the profession and what is expected of them. We also described selected aspects of designing aerospace systems to demonstrate the importance of the architecture, design, and concept of operations to project success. We believe good systems engineers must understand and embrace the tenets of robust design and be able to participate in designing aerospace missions and systems.

We have documented the capabilities that a systems engineer at NASA must have at each performance level. NPR 7123.1a also includes fundamental processes that characterize systems engineering activities across NASA.

Systems engineering is a critical skill within NASA and the entire aerospace community. The Chief Engineer of NASA, working with counterparts at all Centers, has created a wealth of information and opportunities to help develop systems engineers at NASA. The Academy of Program, Project and Engineering Leadership has created a Systems Engineering Leadership Development Program to identify strong candidates for further development and provide opportunities for growth. Many NASA Centers have also created programs to develop their systems engineers. Anyone who is interested in honing his or her systems engineering skills is encouraged to approach the organization's leaders and persist in the quest to grow and develop as a systems engineer.

Bibliography

Collins, Michael. *Carrying the Fire: An Astronaut's Journeys*, New York: Cooper Square Press, June 25, 2001, PPBK, ISBN-10: 081541028X, ISBN-13: 978-0815410287.

Derro, Mary Ellen and P.A. Jansma. *Coaching Valuable Systems Engineering Behaviors*, IEEEAC Paper #1535, Version 5, December 17, 2007.

Ferguson, Eugene S., *Engineering and the Mind's Eye*, Cambridge, MA: MIT Press, 1992, PPBK, ISBN-10: 026256078X, ISBN-13: 978-0262560788.

Gladwell, Malcolm, *Blink: The Power of Thinking Without Thinking*, New York: Back Bay Books, April 3, 2007, PPBK, ISBN-10: 0316010669, ISBN-13: 978-0316010665.

Gleick, James. *Genius: The Life and Science of Richard Feynman*, New York: Vintage Publications, November 2, 1993, PPBK, ISBN-10: 0679747044, ISBN-13: 978-0679747048.

Griffin, Michael D., *System Engineering and the "Two Cultures" of Engineering*, NASA, The Boeing Lecture, 28 March 2007.

Griffin, Michal D. and James R. French, *Space Vehicle Design*, 2nd Ed., Reston, VA: AIAA Education Series, 2004, HDBK, ISBN-10:1-56347-539-1.

Johnson, Stephen B., *The Secret of Apollo: Systems Management in American and European Space Program* (New Series in NASA History), Baltimore, MD: Johns Hopkins University Press, September 20, 2006, PPBK, ISBN-10: 0801885426, ISBN-13: 978-0801885426.

Kidder, Tracy. *The Soul Of A New Machine*, New York: Back Bay Books, June 1, 2000, PPBK, ISBN-10: 0316491977, ISBN-13: 978-0316491976

Kirkpatrick, Doug, Wiley J. Larson, Dale Thomas, Jerry J. Sellers, and Dinesh Verma, *Applied Space System Engineering*, McGraw-Hill, to be published 2009. ISBN TBD.

Larson, Wiley J. and James R. Wertz. *Space Mission Analysis and Design*, 3rd Ed. Dordrecht, Netherlands: Kluwer Academic Publishers, 1999, PPBK, ISBN-13: 978-1881883-10-4.

Larson, Wiley J. and Linda Pranke. *Human Spaceflight: Design and Operations*, New York: McGraw-Hill Higher Education Publications, 2000, PPBK, ISBN-10: 0-07-236811-X.

Lee, Gentry. *So You Want to Be a System Engineer*, DVD, JPL, 2007.

Links:

Personal behavioral characteristics — http://nen.nasa.gov/portal/site/llis/community/SE/workforce/behaviors/

APPEL's SE competencies — http://www.nasa.gov/offices/oce/appel/pm-development/572.html

SE developmental planning matrix — http://www.nasa.gov/offices/oce/appel/pm-development/572.html

APPEL curriculum — http://appel.nasa.gov

NASA HQ SELDP — http://nen.nasa.gov/files/SELDP_Program_Plan_w-o_Funding_10062008.doc
NASA's SE behavioral study — http://nen.nasa.gov/files NASA_SE_BEHAVIOR_STUDY_FINAL.DOC

Logsdon, Thomas. *Breaking Through: Creative Problem Solving Using Six Successful Strategies. Reading, MA: Addison-Wesley*, 1993, PPBK, ISBN-10: 0-201-63321-3. Ph: 562-431-3334.

McCullough, David. *The Path Between the Seas: The Creation of the Panama Canal 1870-1914*, New York: Simon and Schuster, 1999, HDCVR, ISBN-10: 0743262131, ISBN-13: 978-0743262132.

Menrad, Robert J. and Wiley J. Larson. *Development of a NASA Integrated Technical Workforce Career Development Model*, International Astronautical Federation (IAC) Paper IAC-08-D1.3.7, September 2008.

Murray, Charles and Catherine Bly Cox, *Apollo: Race to the Moon*, PPBK, New York: Simon and Schuster, 1989, ISBN-10: 067170625X, ISBN-13: 978-0671706258.

Perrow, Charles. *Normal Accidents: Living with High-Risk Technologies*, Princeton, NJ: Princeton University Press, September 27, 1999, PPBK, ISBN-10: 0691004129, ISBN-13: 978-0691004129.

Personal Interviews, Emails and Discussions:

Michael Bay, Goddard Space Flight Center
Harold Bell, NASA Headquarters
Bill Gerstenmaier, NASA Headquarters
Mike Griffin, NASA Headquarters
Chris Hardcastle, Johnson Space Center
Jack Knight, Johnson Space Center
Ken Ledbetter, NASA Headquarters
Gentry Lee, Jet Propulsion Laboratory
Michael Menzel, Goddard Space Flight Center
Brian Muirhead, Jet Propulsion Laboratory and Johnson Space Center
Bob Ryan, Marshall Space Flight Center
Mike Ryschkewitsch, NASA Headquarters
Chris Scolese, NASA Headquarters
Dinesh Verma, Stevens Institute of Technology

Petroski, Henry. *Success through Failure: The Paradox of Design*, Princeton, NJ: Princeton University Press, 2006, PPBK, ISBN-10: 0691136424, ISBN-13: 978-0691136424.

Rechtin, Eberhard. System Architecting, Prentice Hall, Inc., 1991.

Ryan, Robert. *Lessons Learned—Apollo through Space Transportation System*, CD, MSFC, 2006.

Ryan, Robert, Wiley J. Larson, Vern Weyers, and Douglas Kirkpatrick. *Space Launch and Transportation Systems: Design and Operations.* 2005.

Sellers, Jerry Jon. *Understanding Space*, 3rd Ed. McGraw Hill, 2004, HDBK, ISBN-10: 0-07-294364-5.

Squibb, Gael, Wiley J. Larson, and Daryl Boden. *Cost-effective Space Mission Operations*, 2nd Ed. McGraw-Hill, 2004, ISBN-10: 0-07-331321-1, ISBN-13: 978-0-07-331321-4.

USGOV, Columbia Accident Investigation Board, Report V. 5, October 2003 (3 Books), PPBK, 2003, ISBN-10: 0160679044, ISBN-13: 978-0160679049

Williams, Christine, and Mary-Ellen Derro. NASA Systems Engineering Behavior Study, October, 2008. Publication TBD.