Instar

Doing Things Right in Space Programs

Ten Principles for Doing Things Right in Space Programs

1. **Adopt the right attitude**
2. **Invest in knowledge and understanding**
3. **Instill ownership and responsibility**
4. **Constantly seek ways to improve teamwork**
5. **Follow a sound engineering approach**
6. **Reduce total cost through good engineering**
7. **Keep everything as simple as possible**
8. **Establish an effective quality system that involves everyone**
9. **Be willing to accept risks, but only those you truly understand**
10. **Make sure everyone has enough time, resources, and freedom to do things right**

Article #12
Follow a Sound Engineering Approach

Part 1: The Systems Engineering Process

January, 2001

Tom Sarafin
President, Instar Engineering and Consulting, Inc.
6901 S. Pierce St., Suite 384, Littleton, CO 80128 • (303) 973-2316 • instarengineering.com

Until now in this series of articles, although I've touched on technical aspects, I've dealt mostly with people issues and management philosophies. This will be the first of several articles solely addressing the topic of engineering, including process (Principle 5) and effectiveness (Principles 6 and 7). These subjects are not just for engineers, though.

You can't manage a space program effectively without a thorough understanding of the engineering process.

I plan to address the sixth principle in four parts, each of which will be a separate article:

1. The systems engineering process (this article)
2. Developing and specifying requirements
3. Planning verification
4. Sound processes for engineering activities

To understand what systems engineering is all about, we need to define a couple terms. First, *engineering*: "The application of scientific and mathematical principles to practical ends" (from The American Heritage Dictionary, Second College Edition, 1991, Houghton Mifflin Company). Next, *system*: a group of elements that together perform a function (my definition).

A system can be made of lower-level systems:

– Spacecraft, launch vehicle, ground segment, and operators together make a system for accomplishing a space mission
– A spacecraft is itself a system at a lower level
– A battery is a system within the spacecraft for storing power
– A bolt and a nut comprise a fastening system for attaching the battery

A system has one or more functions, interfaces with users or other systems, inputs, and outputs.

To me, *systems engineering* means coordinating or orchestrating the development of a system. In other words, it means coordinating the engineering process. Starting with an understanding of what someone wants, we set out to develop a product that will do it. Through systems engineering, we ensure that we efficiently end up with a product that does what it's supposed to do. Systems engineering often entails simplifying a complex system into manageable pieces, all the while making sure that those pieces will join and work together and that nothing falls in a crack.

Every engineer on the program is in some way involved in this process. The Systems Engineer allocates parts of the system, coordinates their interfaces, and then integrates those parts.

The Systems Engineer has a broad understanding of the system and of the engineering process, knowing something about everything, and has broad involvement in the process. Someone who specializes in writing requirements, and passes from program to program doing so, is not a true Systems Engineer; he or she is a specialist. Systems Engineers are the first ones on the program, finding concepts, doing trades, and making decisions that drive cost and heavily influence the program's success or failure.

The Structure as a System within a System

Many people having the title of "systems engineer" come from specialized fields such as electrical engineering or thermal analysis and then stay within the bounds of those disciplines. I am continuously surprised by how little most "systems engineers" understand about structures, which is typically considered just another specialty. Stress analysis, vibration, manufacturing—those are specialties, but not the structure itself. A space or launch vehicle's structure is itself a system, one that physically integrates a higher-level system. The structure influences temperatures, the vehicle's flight control system, the performance of sensors, and the vibration that all components will experience. Most of the manufacturing cost will be in building structures.

Everything attaches to the main (primary) structure. The primary structure is usually in the critical path of the program's schedule: until it is built, nothing else can be assembled. Depending on its concept, the structure's tooling may require a long lead time for development. Thus, the primary structure is often the first design released.

Yet few "systems engineers" consider the structure important enough to spend time on early. The structure is often an afterthought, designed in a rush by design engineers who are themselves specialists, often little more than drafters, supported by others equally specialized—well after the vehicle's configuration has been established. Such a process is backwards; after all, the structure defines the configuration. We find the right configuration only by considering load paths, materials, form of construction, mounting interfaces, and methods of manufacturing. Much of a vehicle's cost and weight often come from having to mount equipment to the structure in inefficient ways simply because the structure was not designed to accommodate that equipment. With foresight, we can design structures that are adaptable (the subject of a later article, perhaps) to the unknowns, which are often plentiful as a result of having to release the structure's design early. Such foresight is difficult to acquire, but it's a goal of the Systems Engineer.

The Systems Engineering Process

A space program is normally broken down into phases, as shown in Table 12-1. Government programs are usually funded incrementally to enable the customer to redirect or cancel the program if the apparent cost or risk gets too high. For DOD (Department of Defense) programs, full funding is not authorized until satisfactory completion of the Demonstration / Validation phase. Note the series of reviews, which typically entail the contractor presenting to the customer.

TABLE 12-1. Phases of a Space Program. (Adapted from Table 2.1 in *Spacecraft Structures and Mechanisms: From Concept to Launch* [Sarafin, ed., 1995], and Table 1-2 in *Space Mission Analysis and Design*, 3rd edition [Wertz and Larson, eds., 1999])

Phase, DOD Programs	0 Concept Exploration	I Demon-stration and Validation	II Engineering & Manufacturing Development	III Production and Deployment	Operations and Support
Milestone Goal	Select a concept	Commit to program go-ahead	Commit to production	Commit to deploy (launch)	Operate system
Reviews	△ SRR	△ SDR	△ PDR	△ CDR △ FRR	⇧ Launch
Analogous Phase, NASA	A Concept Explor.	B Definition	C Design	D Development	E Operations
SRR - System Requirements Review SDR - System Design Review PDR - Preliminary Design Review			CDR - Critical Design Review FRR - Flight Readiness Review		

A well publicized, common mistake in developing a system is failing to adequately define its requirements before committing to its design. Many programs have incurred high cost because requirements have changed midstream. The International Space Station is but one example of a program that changed direction many times at huge costs. But firming up the requirements too soon can be an even greater mistake. It's important to understand why most requirements should change early in the design process, as shown in Table 12-2.

TABLE 12-2. The Process for System Design and Development.

Step	A Closer Look
1. Define objectives	▪ Understand customer wants and needs
2. Identify driving requirements	▪ Identify performance drivers and key constraints
3. Develop concepts	▪ Develop concepts and requirements simultaneously
4. Derive requirements	▪ Identify life-cycle events and environments
5. Evaluate concepts	▪ Compare options and make selections
6. Nail down requirements	▪ Flow requirements down to system elements, and follow a similar process with each element
7. Finalize design	▪ Plan downstream activities (manufacturing, handling, verification, system integration and test, transportation, operation)
8. Build system	▪ Before committing to a design, show by analysis and development testing that it will meet requirements
9. Verify compliance	▪ Manufacture or procure parts, components, and assemblies; verify requirements for each before proceeding to a higher level of assembly
10. Implement system	▪ Integrate system elements and verify system requirements

Note that the process does not start with defining requirements. We certainly could do that, but we usually don't end up with a cost-effective system that way. Performance has a price, and, until we assess that price, we won't know whether that level of performance is a smart buy.

Sometimes a certain level of performance is required, or the mission is not worth doing. A spy satellite, for example, may require image quality of a certain resolution in order to tell a real parked fighter plane from a mockup. In such cases, we identify up front the needed performance as a driving requirement. What we must avoid in such a case is also establishing arbitrary limits on budget and schedule. We can either try to minimize cost and schedule for given performance requirements or try to maximize performance for given budget and development time. Specifying performance while arbitrarily constraining budget and schedule leads to the types of problems for which our industry is noted.

Most requirements can and should be flexible until we've evaluated concepts. If, for example, we are willing to back off, say, ten percent on our desired data rate, perhaps the spacecraft could be small enough to fit on a launch vehicle that costs half as much.

The process shown in Table 12-2 starts with objectives, not requirements: what we want (e.g., to conduct experiments in space) rather than how well the system must do it (able to house ten people for eight weeks without servicing). With a clear understanding of the objectives—along with any driving requirements, such as the constraints of budget, time, and interfaces—we can start brainstorming solutions, or system concepts. It's important to identify several alternatives for comparison. As we develop each concept, we uncover requirements needed to make it work, and we can begin to evaluate it: What level of performance could this option deliver? How much would it cost? How long would it take to develop? How sure are we that it would work?

While making this evaluation, we consider the full system over its complete life cycle. For example, before committing to a spacecraft concept, we look at how we will build and test it, how it will interface with the launch vehicle, how we will operate it in space, and how we will dispose of it. Remember, the launch and ground segments are part of the overall system. That low-cost concept for a satellite design we may have come up with may lose out in the system trades if fourteen costly ground stations are needed to operate it.

As this process proceeds, we begin to recognize concepts and levels of performance that either fit our budget and schedule constraints or provide the most value, depending on our objective. And, as they do, we start to whittle down the options and firm up the requirements. Fair trade studies are important, but eventually they must end—even if we're not convinced we've found the best solution. It's easy to fall in the trap of doing just one more trade over and over again until the program gets cancelled. We often must compromise with "good enough" rather than optimum.

Throughout the process thus far, we document and coordinate the evolving requirements in the form of a *system specification*. Such a document captures all the requirements in a single place so we all know what we are trying to build. A system specification keeps things from falling in cracks. As with everything else (Principle 7), we need to keep the specification simple or we will defeat its purpose.

In developing a system concept, we must consider the elements that make up that system. Here is where the process can become complicated. For the system to work, the elements must physically attach and work together, yet they will be designed and manufactured separately. Software also will be composed of elements that are developed separately yet must work together. Thus, a key part of systems engineering is breaking down the system-level requirements into requirements at the element level. Complex systems have several tiers of elements, such as vehicle, subsystem, component, part, and material. For any element to meet its requirements, its ingredients (lower-tier items) must meet theirs. Our challenge is to start at the system level and flow or allocate requirements down through the tiers without losing anything and without driving unnecessary cost. At each level, the process includes iteration between brainstorming concepts and defining requirements. Blindly allocating requirements down to the part level would drive us to designing our own bolts and nuts when the obvious solution is to make our design work with available bolts and nuts.

Whether you should generate specifications at the element levels depends on the complexity of your system and the extent to which you use off-the-shelf items. If your system is simple enough, a system specification alone may suffice, with element requirements captured in engineering drawings. Parts and assemblies that we build in-house are examples of items that usually don't warrant specifications other than engineering drawings.[1] If your system is comprised of elements that are themselves complex systems, you'll need to generate lower-level specifications for each in order to manage the process. Figure 12-1 shows a representative hierarchy of system elements for a space vehicle and the documents typically used to capture their requirements.

[1] Make sure your engineering drawings conform to ANSI Y14.5, a standard from the American National Standards Institute, to ensure everyone (engineers, contractors, machine shops, etc.) understands them.

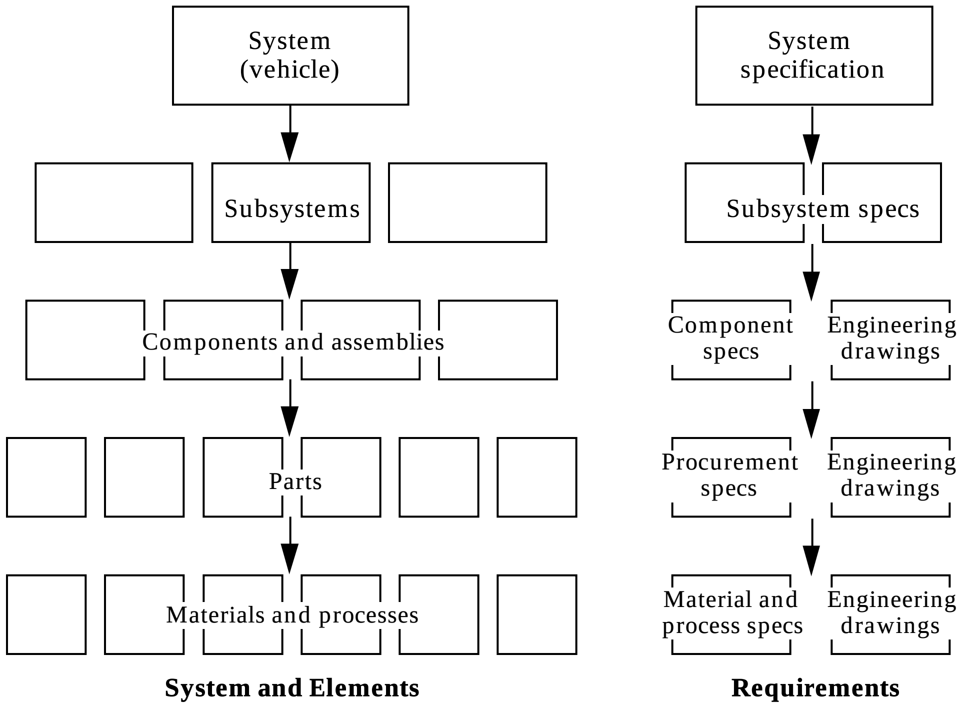

Fig. 12-1. Requirements Hierarchy. The systems engineering process includes deriving requirements for the elements of the system. This normally entails flowing requirements all the way down to the raw materials and manufacturing processes in order to ensure the entire system will work properly.

At the same time we are developing concepts and requirements, we also must begin planning the verification program. Recall that *verification* means establishing confidence that the system will perform as needed. In a space program, we need a great deal of confidence before launch because afterwards we won't be able to fix many of the problems that could occur. Thus, what we consider to be verification ends with the decision to launch. We do verification through analyses, inspections, tests, and demonstrations. Much of the cost our program will incur up until we deliver the system for launch will be in verification.

Planning for verification must start early for three reasons:

1. It's needed to evaluate design concepts. If we can't find an affordable, effective way to verify requirements, then we haven't found a good design.
2. It's needed to scope the program and provide adequate funding and schedule.
3. It will be the source of design requirements. For example, test environments bound mission environments, often with a designated margin, and we'll want to design our products to withstand those test environments.

Many people see verification as simply another requirement levied by the customer. It is this attitude perhaps more than any other that gets us in trouble. We spend a great deal of money often with little true benefit, and we overlook key issues. As soon as our customer stops telling us what to do—or we get a new customer who doesn't know what to tell us—we're tempted to cut back on verification in the spirit of being willing to accept risk in order to reduce cost.

At such times, we forget that verification is integral to the engineering process. It's not about plodding through a series of required analyses and tests—it's about becoming confident in our system and then instilling that confidence in our customer. If you had no customer and were, in fact, spending your own money on a space mission, wouldn't you follow a systematic process of becoming confident before launch that the mission will be successful?

With a sound engineering process, we flow requirements down to the system's elements and then make sure those elements meet their requirements before assembling the system. Whereas we develop requirements from the top down, we verify them from the bottom up (Fig. 12-2). Verification follows a building-block process, rooting out problems when it's still affordable to do so.

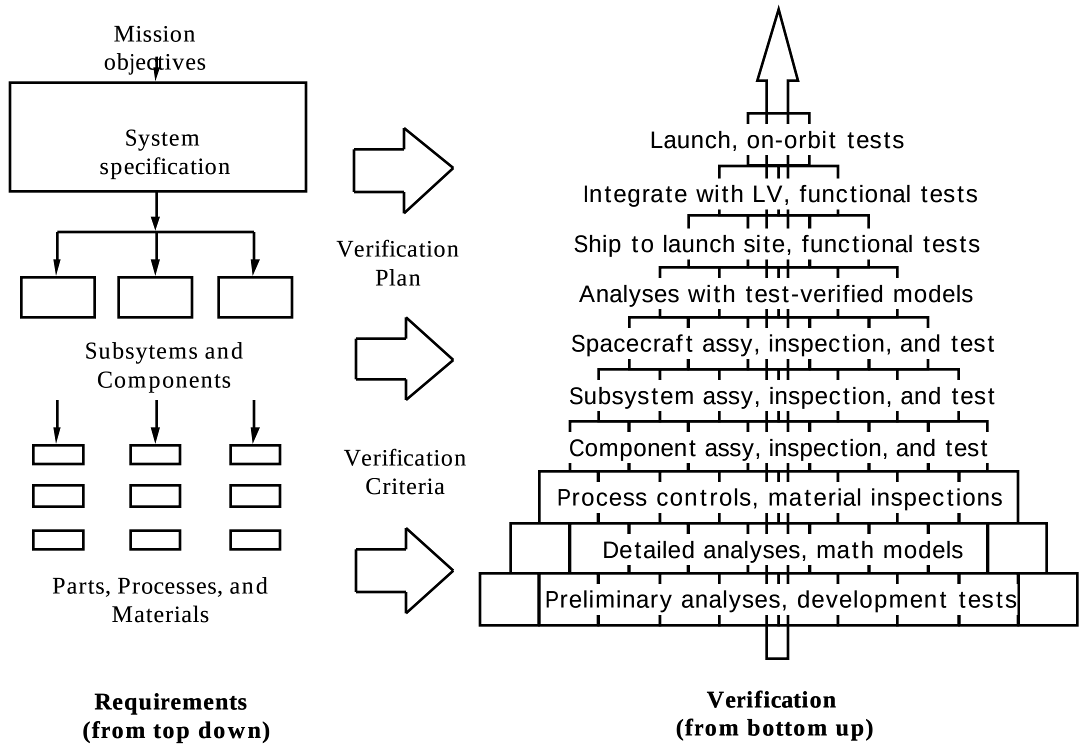
Mission objectives
System specification
Subsytems and Components
Parts, Processes, and Materials
Requirements (from top down)
Verification Plan
Verification Criteria
Launch, on-orbit tests
Integrate with LV, functional tests
Ship to launch site, functional tests
Analyses with test-verified models
Spacecraft assy, inspection, and test
Subsystem assy, inspection, and test
Component assy, inspection, and test
Process controls, material inspections
Detailed analyses, math models
Preliminary analyses, development tests
Verification (from bottom up)

Fig. 12-2. Requirements Flow from Top Down, and Verification is from Bottom Up. Verification is connected to requirements by the verification plan, criteria for design and verification, standards, and other documents.

Analysis is our first tool for rooting out problems. In a sound process, analysis steers the design. I've been involved in several programs in which managers chose to release designs without analysis in order to meet aggressive schedules, with the plan of finishing the analysis while manufacturing is underway. Those managers apparently saw analysis solely as something required by the customer rather than as an essential ingredient to a good design. Either that or they felt their hands were tied by an unrealistic schedule.

Another valuable tool during design is *development testing*, which is testing low-cost, representative hardware with the goal of learning how to design the expensive item so that it will work. We do development testing when we don't have enough information to do dependable analysis. With new, risky technologies, it doesn't make sense to put too much faith in analysis and wait to test for the first time at an expensive level of assembly.

Verification also encompasses what we think of as quality assurance. Making sure that raw materials meet their requirements, that hardware is built to specified dimensions, and that tests are done as planned are all for the same purpose: becoming confident that the product will work or that the mission will be successful.

Thorough verification is simply good engineering. We don't try to reduce cost by compromising the engineering. If someone has raised an issue and we don't address it—or if we don't take time to turn over all the rocks—then we are not following a sound engineering process. We are irresponsibly using someone's money.

Attention to detail has always been the key to success. Note the words of Glenn L. Martin, one of the true pioneers of the aircraft industry:

"The way to build aircraft or do anything else worthwhile is to think out quietly every detail, analyze every situation that may possibly occur, and, when you have it all worked out in practical sequence in your mind, raise heaven and earth and never stop until you have produced the thing you started to make."[2]

As perhaps you are starting to realize, verification has always been so deeply entrenched in engineering, or product development, that few industries even separate it into a distinct word as our industry does. The term "verification," in this sense, grew from Government purchases of complex systems. When customers hire us to develop products instead of simply purchasing new products from us—risking their money instead of ours—the need arises to expand the sound engineering process to include convincing the customer throughout development that the product will work. Verification is no more than what we'd normally do in a sound engineering process other than satisfying our customer that the process we are following is indeed sound.

Goals of the Systems Engineering Process

Throughout the process of defining and deriving requirements, and planning and conducting verification, our goals are to

- Find a cost-effective design for satisfying mission objectives
  - This is why we iterate the design along with the requirements.
- Identify every requirement (anything needed to ensure success) and make sure we address it.
  - We do this through diligence, effective documentation, and reviews.
- Have no unanswered questions at the time of launch
  - We do this by constantly asking ourselves, "What could go wrong?" We then pursue each possibility until we are satisfied it will not occur.
- Ensure failure is unlikely, with acceptable risk, rather than impossible

[2] From Harwood, William B., 1993, *Raise Heaven and Earth: The Story of Martin Marietta People and their Pioneering Achievements,"* Simon & Schuster, New York.

– Trying to prove failure is impossible is a common trap in which we keep spending money with continuously lower returns. It's not possible to prove a space mission will be successful; there are too many random variables outside our control. If we want to do more space missions, we must keep them affordable, which implies we must accept some risk of failure. Because the contractor and customer each have a say regarding verification, most likely with different opinions regarding what constitutes "acceptable risk," a space program needs to establish *verification criteria*, which are ground rules for determining the acceptability of a verification activity (e.g., a test). It is the contractor's responsibility to derive requirements, identify things that could go wrong, investigate them enough to understand the problem, and then propose a verification plan, complete with criteria, to the customer for approval.

• Keep everything as simple as possible

We do this several ways. First, we keep our designs simple, with few unknowns that could affect performance. Second, as noted earlier in this article, we think through the verification plan while defining requirements to ensure verification is affordable. Finally, also as noted above, we define criteria that enable us not to have to seek perfection.

Documentation

Effective documentation is key to managing the process of system development. Table 12-3 describes some of the key documents. Space programs are continuously finding, with computers, better ways to distribute, maintain, and interrelate their documentation.

Table 12-3. Key Documents Generated when Developing a System. Many other documents may serve equally important roles.

Document	Description	Purpose	When Initiated
System specification	Assemblage of requirements at the system level	Make sure everyone knows what they are designing	Concept Exploration
Configuration drawing	Scale drawing defining the system configuration, including key dimensions and locations of key items	Enables people to visualize the system and ensures everyone works on the same thing	Concept Exploration
Interface control document	Defines everything needed to ensure two parts of a system developed separately will interface properly	Coordinate an interface; provides information; does not specify requirements	Varies

Table 12-3. Key Documents Generated when Developing a System. (Continued)

Document	Description	Purpose	When Initiated
Verification and quality assurance plan	Definition of activities planned to ensure (or establish confidence) before launch that the system will meet its requirements	Build confidence, enable the program to be scoped properly, identify design drivers, focus efforts, and plan the program	Concept Exploration
Design criteria and verification criteria	Standards for design and analysis and ground rules for verification activities	Ensure quality; make sure planned verification activities will provide adequate confidence	Concept Exploration
Engineering drawings	Documents defining all physical features for parts and assemblies and requirements for materials, handling, storage, etc.	Communicate require-ments to Manufacturing and ensure engineers know what is (or will be) built	Concept Exploration (layouts) and Engineering (released engineering)
Drawing tree	Hierarchy of engineering drawings	Control configuration and coordinate drawings	Varies
Parts, materials, and processes (PMP) control plan	Requirements for procurement and processing; lists of acceptable materials and parts	Ensure parts, materials, and processes are compat-ible with mission environ-ments and otherwise meet requirements	Dem / Val
Manufacturing plans, test plans, and procedures	Plans capture engineering requirements for planned activities; procedures specify detailed implementation	Communicate requirements and ensure quality	Varies
Analysis reports, test reports	Organized reports documenting key activities	Build confidence, answer questions, and assist in future activities	Varies
System compliance report	Top-level report summarizing results of the verification program and pointing to lower-level documents	Build confidence and ensure traceability	Production and Deployment

In my next article, I'll give a closer look at developing requirements. After that, the subject will be verification planning.

About the Author

Tom Sarafin has been involved in the space industry full time since 1979, at which time he graduated from The Ohio State University with a BS in civil engineering and took a job as a stress analyst at Martin Marietta Astronautics in Denver, Colorado. While at Martin, he was involved with design, analysis, verification planning, and testing on several spacecraft and launch vehicle programs. After contributing to the book *Space Mission Analysis and Design* [Larson and Wertz, editors, first edition published in 1991], he obtained management's support and funding at Martin Marietta for the development of a book on the interdisciplinary development of structures for space missions, and served as principal author and editor for 23 other authors. He left Martin Marietta in 1993 to complete this book, under the guidance of Dr. Wiley Larson at the U.S. Air Force Academy. The result of nearly four years work—*Spacecraft Structures and Mechanisms: From Concept to Launch*—was published in 1995 jointly by Microcosm, Inc., and Kluwer Academic Publishers.

In 1993, Mr. Sarafin formed his own company, Instar Engineering and Consulting, Inc. Once he finished his book, he began providing review and advice as a consultant to space programs. He also developed a short course based on his book and began teaching it throughout the industry. The course has been quite popular, and the business has grown. Now Instar offers a curriculum of courses taught by experienced engineers and continues to add to that curriculum.

Instar's Core Courses

- **DTR—Doing Things Right in Space Programs: A course for managers**
- **SDV—Doing Things Right in System Development and Verification**
- **USS—Understanding Spacecraft Systems**
- **SMS—Space-Mission Structures: From Concept to Launch**

Additional Instar Courses

- DASS—Design and Analysis of Space-Mission Structures
- USRV—Understanding Structural Requirements and Verification
- SPAD—Space Propulsion Analysis and Design
- OSPS—Overview of Space Propulsion Systems
- DAFJ—Design and Analysis of Fastened Joints
- APSIT—Avoiding Problems in Spacecraft Integration and Test
- GDT—Geometric Dimensioning and Tolerancing

Additional courses in work; customized versions available

For information on these courses, visit our website at instarengineering.com