Instar

Doing Things Right in Space Programs

This article is part of a series started in January, 2000. My intent is to share a philosophy and ideas for how to increase the chances of success in space missions while also reducing total cost. Once these articles are completed, I plan to assemble them into a book. Please send comments to me at Tom.Sarafin@instarengineering.com.

The articles in this series, as they are written, are posted on our website, instarengineering.com, and are available for free downloading. You are free to forward this article by e-mail, print it, copy it, and distribute it, but only in its complete, unmodified form. No form of mass publication is permitted. Small parts of the text may be quoted, but only with appropriate credit given. Otherwise, no parts of this article may be used in any other work without my written permission.

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, systems-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 #17

Reduce Total Cost through Good Engineering
(Not by Compromising Quality)

March, 2002

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

Throughout this series of articles, I've stressed the importance of not compromising quality with unknown risk in order to reduce cost. Instead, I've said the right ways to reduce cost are to improve our processes, teamwork, and communication; to invest in our people and build morale; and to improve our management and our engineering.

The topic of this article is good engineering. I plan to illustrate how, through effective engineering, we can reduce cost responsibly, and I hope to plant seeds for ideas. As is the goal of most of these articles, I'm trying to spur thought.

Following a sound engineering approach (Principle #5) is a necessary part of good engineering, as is keeping things simple (Principle #7). But good engineering takes more than this.

My definition of *good engineering* is anticipating downstream costs and problems when designing a product or planning an engineering activity and taking prudent steps to minimize or avoid them. This is, after all, why we follow a sound approach and strive to keep things simple. The goal is high quality at the lowest total cost. To obtain this goal, we must be willing to invest in the near term in order to avoid higher costs down the road.

To encourage and enable good engineering, management must stop emphasizing metrics for near-term cost savings, such as reduced engineering hours per sheet of drawing. What good is reducing engineering hours before drawings are released if many more hours are needed later a result of a poor design? Clearly, reducing design budgets not only discourages good engineering, it prevents it, even when the staff is capable.

Let's look at some examples of good engineering:

- **Scrubbing requirements**—especially when specifying them to other organizations. Our goals are to demand nothing that is not essential, word requirements such that they can be interpreted only one way, and not specify requirements for verification or quality assurance (see Article #8). Yes, defining requirements is part of engineering, and the rest of the engineering can't be good if the requirements are lousy.

  If you're an engineer, regardless of your position, go out of your way to contribute to requirements definition, if you can. If you leave the job of specifying requirements to a specialist or, worse yet, a bookkeeper, you will be doomed from the start. Look at requirements definition like a manufacturing engineer looks at design: Poor requirements can make your life a living hell, just as a poor design can do for a manufacturing engineer. Fight to get your inputs considered before requirements are released, the same as manufacturing folks do before designs are released.

- **Not letting a poor requirement drive cost**—Okay, you were too late. Requirements were specified before you got on the program, or you didn't have the opportunity to contribute for some other reason. Now you find yourself being expected to satisfy a requirement that makes no sense to you. What would a good engineer do in this situation?

  Speak up. Either you understand the problem better than your customer (or whoever wrote that requirement), or you're not seeing the whole picture yourself. For your engineering to be good, you have to believe in what you're doing. When you don't, challenge the requirement and either get a satisfactory explanation or try to bring sanity to the process. Too few engineers think this way; too many simply roll with the punches and do what they're told to do. Challenging requirements is often essential for a program to be cost-effective. Doing so tactfully is the key to a successful challenge.

- **Designing the space or launch vehicle from existing, qualified components**—In the 12th article in this series, I explained how requirements should be derived from the top down when developing a complex system. Although such a process is essential to some extent, we often go overboard and end up driving unique requirements and

custom requirements for components. Many times, off-the-shelf components would have worked with minor impact to the system.

As an example, consider how we typically derive vibration environments for the components and subsystems of a launch-vehicle payload: We configure the payload (spacecraft), predict how it will respond to the launch environment, and then specify environments as part of custom orders for the vehicle's components. Instead, is there a way we could use components that are already qualified and available? Can we control component environments through design of the structure and component mounting systems?

Similarly, rather than allocating performance requirements blindly from arbitrarily selected system requirements, can we start with a list of available components and see how much performance we can get for the system? Consider a reaction wheel as an example. We've flown reaction wheels on many spacecraft, so we know qualified designs are available. Any given design will produce a certain amount of torque, have a certain amount of wobble (which can cause jitter, or low-level vibration, to a spacecraft in orbit), and certain electrical and mechanical interfaces. A good engineer will find out the characteristics, interfaces, and demonstrated capabilities of available wheels—and batteries, and star trackers, and antennas—and will try to find a spacecraft design that will meet mission objectives with such components.

- **Planning out the complete product life cycle early, and considering all events before releasing designs**—Key events include manufacturing, handling, test, vehicle integration, transportation, launch-site operations, and mission operations. "Considering" is too soft of a word. The goal is to design something that can easily be built, tested, handled, transported, integrated, serviced, and operated. This is nothing knew; it's the basis of concurrent engineering and Integrated Product Development. The best design engineers understand how the parts will be built, assembled, and tested, and they design things that can be built, assembled, and tested easily.

  As an example of how understanding downstream activities can help you design something that will have reduced overall cost, consider manufacturing tolerances. Many designers specify tolerances (acceptable ranges for specified dimensions) arbitrarily, although it's only logical that tight tolerances are more costly to achieve than loose tolerances. Table 17-1 shows how the cost of machining increases as tolerance decreases.

Table 17-1. Relative Cost of Machining for Different Dimensional Tolerances. (From *Design for Manufacturability, Optimizing Cost, Quality, and Time-to-Market* [David M. Anderson, 1990, CIM Press].)

Tolerance	Process Required	Cost
0.030"	Rough machining	$
0.005	Standard machining	2 x $
0.001	Fine machining	3.5 x $
0.0005	Grinding	6 x $
0.0002	Honing	10 x $

Yes, many parts in a space or launch vehicle need tight tolerances to function properly, but a good engineer recognizes there should always be a reason for the tolerances specified: either based on functional requirements or based on what the planned manufacturing process can achieve.

- **Developing a physical configuration for the vehicle that is robust and adaptable**—The structure defines the configuration. The structural design usually must be released first because everything else builds off the structure. Despite our best efforts for concurrent engineering, the reality of a space program is that things will change after we've released the structural design. Components will be modified; they will be larger, heavier, and more numerous than we had planned. A good configuration engineer provides room between components, not only for potential growth but also for cable routing, and generous access. And a good configuration engineer finds a structural concept that is easy to attach to without costly (and heavy) adapter structures.
- **Coordinating interfaces**—to avoid costly miscommunication and ensure the elements of the system will come together seamlessly. This is important regardless of who is developing the system elements, but it's more difficult and thus requires more effort with physical distance between parties and with greater organizational or cultural differences. Units for shared data are especially subject to misinterpretation. Good engineers recognize the potential pitfalls and take no chances, not only by thoroughly defining things, but also by following up to ensure understanding.
- **Designing launch vehicles to accommodate payloads**—with simple interfaces, simple integration processes, and vibration during launch that is tolerable and predictable. As with scrubbing requirements, a moderate effort here will be paid back many times in future savings. The challenge to our industry is to make sure the folks making the initial investment will benefit from those savings. Unfortunately, this has not been the case so far. New launch-vehicle designs are procured, usually by the federal government, to lift payloads of a designated mass to desired orbits. Direct cost—development cost and recurring cost of the launch system—are considered heavily in procurement, but indirect costs—of complex interfaces and severe environments for payloads—seldom are. This is changing.

  In December, 2001, the Defense Advanced Research Projects Agency (DARPA) issued a solicitation for bids on the first of three phases of development for a new launch system, referred to as Responsive Access, Small Cargo, and Affordable Launch (RASCAL). The envisioned system uses a reusable aircraft for the first stage, with two expendable rocket stages for attaining final orbit. I contributed to the solicitation as a consultant to DARPA. The following appeared in the final solicitation; the referenced section 8.2 was a white paper I had written, which I may include in a future article.

  > "… a complete vision of affordable access to space … must acknowledge the indirect costs of launch to the payload organization: those costs driven by constraints, environments, and verification processes imposed by the launch system. We are challenging offerors to develop a transportation system that greatly reduces such indirect costs from those typical of existing launch systems. RASCAL should provide a "soft ride" (relatively free of potentially damaging vibration) that is predictable and dependable. Doing so will simplify payload testing and prelaunch analysis, enable lighter payload structures, and enable better chances that the satellite will be functional once it attains its orbit. The obvious way to provide a soft ride is to

decouple the launch vehicle's dynamics from those of the payload through use of a vibration-isolating mounting system, similar in concept to the suspension system for an automobile. We believe such a mounting system is feasible and affordable if incorporated early enough in RASCAL design (Details in section 8.2)."

- **Accounting for variation and uncertainty**—A good engineer recognizes variation in material properties, processes, and environments—and uncertainty in assumptions, math models, and collected data—and designs the product to work despite them. Development testing and analysis are often needed to get a handle on such variation and uncertainty.

  Figure 17-1 illustrates how using analysis to understand the problem can help us avoid costly problems downstream by making the design robust, or tolerant of uncertainty. By working with the launch-vehicle provider, we can derive a response spectrum such as this one for the mounting interface for our spacecraft. Such a spectrum would steer the structural design of our spacecraft away from certain natural frequencies that might prove troublesome.

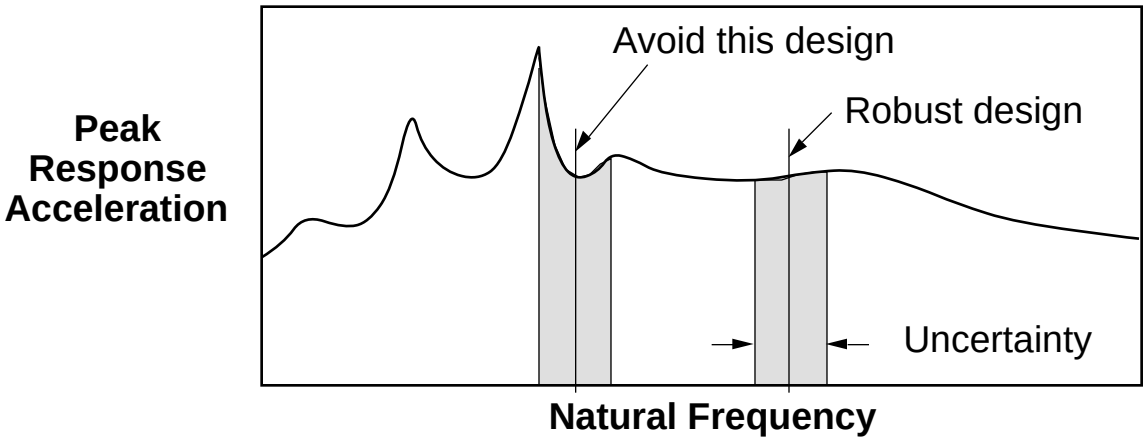

Fig. 17-1. Making the Design Robust for Launch Vibration. This plot is *a response spectrum*, showing the peak acceleration of a mass on a spring in response to known vibration of its mounting structure. Response is shown as a function of the natural frequency of the mass-and-spring system, for an arbitrarily assumed, constant damping. The plot, which we can easily derive by analysis, tells us how sensitive the response is to uncertainty in predicted natural frequency. For both designs shown, nominally predicted response is the same. But, over a reasonably expected uncertainty band (shown shaded), the design on the right sees much less variation in peak response, which makes it a better, more robust design. (Adapted from Fig. 1.3 in *Spacecraft Structures and Mechanisms: From Concept to Launch* [Thomas P. Sarafin, ed., 1995, Microcosm, Inc., and Kluwer Academic Publishers].)

  A good engineer does not stop to ask whether good engineering is required, does not say things like, "Accounting for uncertainty in my analysis is not a requirement." A good engineer recognizes his or her job is to develop a product that will do what it's supposed to do and understands that accounting for uncertainty is simply part of that job.

- **Prioritizing and planning analysis**—to recognize key problems early and to obtain no more accuracy than is needed. Accuracy is unfortunately the key objective of many analysts. But your analysis is of little value, despite its accuracy, if it's not done in time to influence the design. We must rid ourselves of the notion that analysis is for the

purpose of satisfying verification requirements. The main objective of analysis is to help us understand the problem so we can design an efficient product that works the first time. A good engineer recognizes this objective and coordinates his or her analysis to be compatible with design schedules.

- **Testing early, when it's inexpensive**—A good engineer recognizes early when there is not enough information to adequately predict the product's characteristics. In such situations, a good engineer educates the people who can authorize development testing before committing to the design. I've hammered plenty on this point in previous articles. Unfortunately, the benefits of development testing—as with many other proactive efforts towards avoiding problems—aren't very noticeable. We notice problems a lot more than their absence.
- **Planning tests thoroughly**—A good plan ensures the test will meet its true objectives rather than perceived requirements, coordinates activities so everyone will be ready for the test, and allows everyone to buy into the approach or object when an objection can still have influence.
- **Preparing for tests thoroughly**—by anticipating results and planning strategies for how to respond, so that armies of test personnel won't be left standing while you scratch your head. Prepare by projecting possible outcomes and planning what you would do in such events.
- **Designing a spacecraft that will meet mission objectives without a propulsion system, without attitude control, and without active thermal control**—There, I did it. I couldn't help it. I was trying to get through this list without a single reference to keeping things simple so I could leave everything on that subject for my discussion of Principle #7. But the temptation was too great. Keeping things simple is so essential to good engineering that I had to say something about it here!

As you read through the above list, hopefully you begin to see how many often possibilities there are for reducing cost without compromising quality—indeed, while improving quality. When we consider also the potential for better management, communication, and teamwork, the potential for improvement becomes mind boggling.

The ten principles for Doing Things Right are interactive and are prerequisites for each other. Good engineering requires knowledge, foresight, and initiative. It also requires a broader understanding of the system, the development process, and the mission than specialists are expected to have. Perhaps most important, good engineering requires the right attitude.

A friend of mine, whom I'll refer to as Frank, recently shared a problem at his company that illustrates much of what is wrong in the space industry—and also much of what is right in our best engineers. For a Government job, he had submitted an estimate of 1000 design hours. The estimate was arbitrarily cut in half after the customer representative responded with, “What? My secretary and I could get that drawing out overnight! I’ll give you 500 hours.” Frank's company gave in, and Frank was furious. He told his manager, “Don’t get upset when we overrun.”

I baited Frank by suggesting that the other option was to stick to the budget and release a poor design.

He said, “Not on my watch. Maybe my successor will, but I don’t work that way.”

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