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 #7
Invest in Knowledge and Understanding

July, 2000

Revised October, 2002

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

There is no secret to having successful space missions at reduced cost. It doesn't require advanced materials, virtual prototyping, or more powerful computers. Those things may help, if used wisely, but they are not essential. Cost-effective, successful missions come from effective engineering, communication, cooperation, and management, which are natural products of having a sound understanding of the problem. You know as well as I that, if we get the right people on the program, they'll give us a successful mission we can afford. In this sense, the "right" people are simply the ones who understand things—technical, economical, and interpersonal—best.

It only stands to reason that developing our people (and ourselves) should be our highest priority. Rather than pursue the short-term benefits of having our people develop specialized skills that they can practice over and over, we need to aim at the long-term benefits of having engineers learn by staying on the program to implement the requirements they wrote or the designs they developed. To stay in business, we must invest our own money to educate our people rather than expect our customer to do it. And, to increase our contributions to our programs, as well as to ensure job security, we must be willing to invest our own time to become better educated.

College does not adequately prepare us for this industry. It never did, but new grads today are perhaps less prepared than ever before. Industry (not just ours) has been sending the message to academia that it wants graduates who know how to use the popular engineering software. As a result, we have sacrificed education for training. There is a difference, you know. *Training* teaches people how to <u>do</u> something, whereas *education* helps people <u>understand</u> something, which is a prerequisite to doing something well. Our colleges, our high schools—even our elementary schools (e.g., calculators)—have sacrificed education for training.

Engineers learn through a combination of theory and practice. The latter teaches not only how to modify theory to match real life but also how imperfection and the human element can overwhelm theory. Many college engineering programs have reduced time in the field, such as testing hardware or visiting factories, to free up time for sitting behind computers. In my civil engineering program at the Ohio State University, the activity that taught me most was making our own concrete cylinders and then testing them in compression to failure. Among other things, we learned the importance of process control (mixing, pouring, and curing the concrete): if our team didn't do things right, our cylinder broke before those of the other teams. The same applies for many materials and forms of construction we use for spacecraft. Such time on the floor—actually doing something—is being reduced at most colleges so that engineers can instead learn how to use finite-element software, CAD tools, or other software. I remember also taking a required course on engineering drafting that still helps me understand engineering drawings. Ask your new-grads if they took such a course.

Unfortunately, in this world of specialization, many engineers go directly from college to more computers. In the space industry today, it's not uncommon to find engineers with ten or more years experience who have never witnessed a test or seen hardware being built.

Most executives are unaware of the extent of this problem; many apparently believe there is no such problem at all. This is understandable for three reasons. First, the higher up the management ladder, the further removed from day-to-day operations that person typically is. High-level managers and executives often spend each day going from meeting to meeting; how often do they visit the trenches and take time to see what goes on there?[1] Second, this is a hard problem to accept. It's human nature to ignore problems to which there are no easy solutions. Third, many managers and executives have been promoted for other skills than technical ones, and many come from overspecialized fields

[1] At one company I recently tried to explain to an executive how the structural engineering process could be streamlined by integrating process steps that are normally done separately by specialists. He interrupted me, saying, "Tom, our engineers already do these things." Given the circumstances, I couldn't find a tactful way of saying, "No, you're wrong. I've had your engineers in my courses, and they are <u>not</u> doing those things."

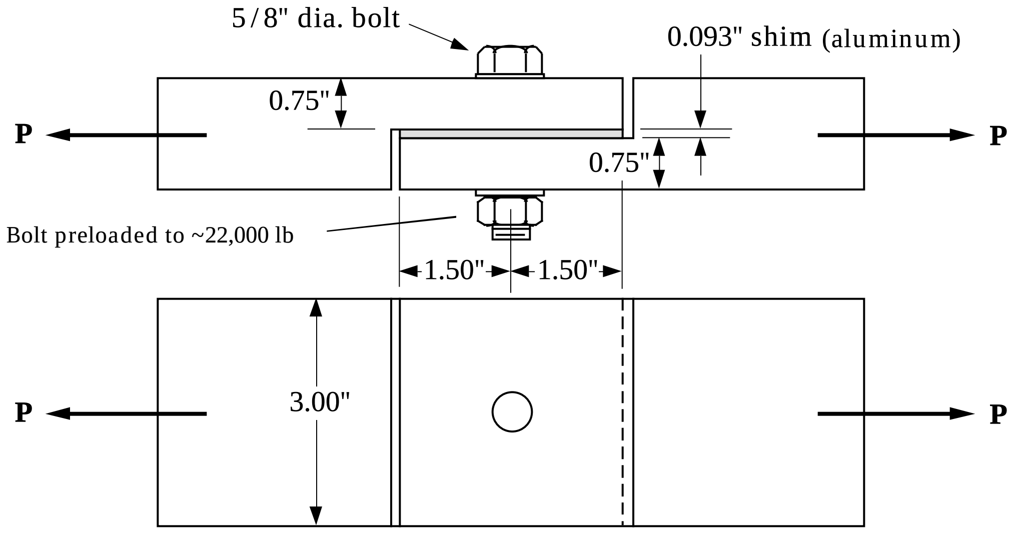
themselves. As a result, they may not understand the technical issues well enough to recognize inadequacies in how they are addressed. A common human failing is to think we understand something when we've only scratched the surface.

Whether or not we want to believe it, the space industry is in a crisis. Engineering understanding is declining, on the whole, while we are at the same time cutting budgets and relying more heavily on computers. To reduce cost and development time, we are foregoing traditional inspections, tests, and other activities meant to insure quality without a sound understanding of why those activities were there in the first place.

Over the past several years, I have been in a rare position of being able to evaluate, at least in part, the level of engineering understanding in the space industry. I have taught courses, mostly on structures, to over 1200 engineers and managers from many of the large aerospace companies and several Government organizations. Let me share some of what I've found.

Figure 7-1 shows the design of a simple structural joint that I've used in a class problem. I've asked over 500 engineers to calculate a design allowable yield load for this joint; in other words, the highest load *P* the joint could carry and still function properly. Nearly all of the participants were structural engineers in the space industry: design engineers, stress analysts, and structural dynamics specialists. The answers have varied from a high of 90,000 lbs all the way down to 2000 lbs—a difference of a factor of 45.

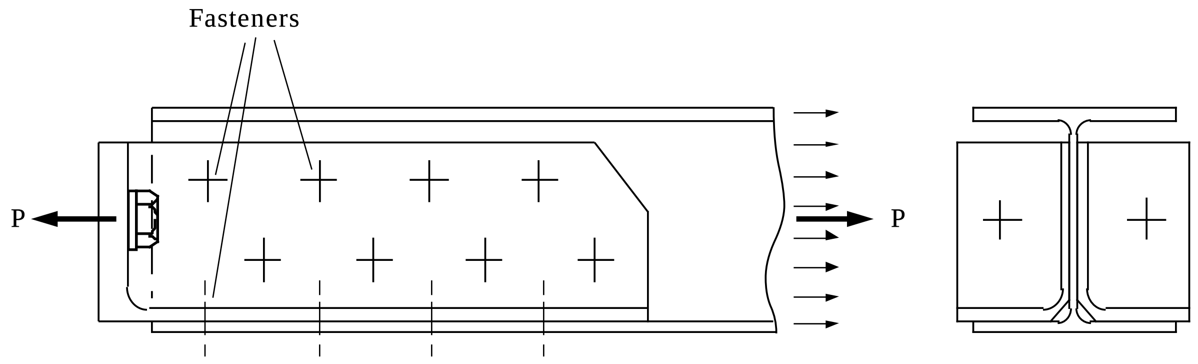
Of all the space programs opting to fly their structures without testing them, none I'm aware of is using a design factor of safety of 45.



Fastener material:	A-286 alloy	F_{ty} = 120 ksi	F_{su} = 95 ksi
		F_{tu} = 160 ksi	
Fitting material:	Aluminum alloy	F_{ty} = 49 ksi	F_{bry} = 94 ksi (for e/D = 2)
		F_{tu} = 62	F_{bru} = 123 ksi (for e/D = 2)

Fig. 7-1. What is the Design Allowable Yield Load for This Joint? Answer: somewhere between 2000 lbs and 90,000 lbs. Finite-element analysis would be of little help; how would you model the interface between the bolt and the fitting? And remember: we don't care about localized yielding, we just want to make sure there is no permanent detrimental deformation. The correct answer: If this is an alignment-critical joint, you had better test it. I did, and the results were surprising.

Figure 7-2 is representative of a structural joint that ruptured in test over twenty years ago at Martin Marietta, Denver, as part of a structural assembly. The mode of failure was not recognized then, and it would not be today. I know this because I've asked over 1000 engineers in my classes to identify potential modes of failure, based on the premise that you can't preclude a failure in design unless you first recognize it. Only one engineer out of 1000 included the actual mode of failure on his list. I'm not saying only one said this was how the joint would fail; only one engineer identified the failure as being worthy of assessment.



Materials: ductile metals

Fig. 7-2. This Joint Failed its Structural Test. Only one of over 1000 aerospace structural engineers polled included the actual mode of failure in the list of modes that he would assess.

I'm sharing these examples not to make our engineers out as stupid. Engineering is a difficult profession, and there's a lot to learn. Engineers have no chance of learning if they never leave their computers. I had the good fortune of being able to test the joint in Fig. 7-1, and I was able to learn from the people who tested the joint in Fig. 7-2.

A lack of understanding appears to be at the root of many of our problems, as evidenced by the following things I've run into over the past several years:

- Programs showing they did not understand the concepts of variation in materials, processes, and workmanship by planning to environmentally test only the first-built spacecraft or component without sufficient data demonstrating their processes were controlled well enough to provide repeatable ability to withstand those environments.
- A program showing they did not understand the basic engineering process by deciding not to structurally test their alignment-critical payload while also deciding not to document their analysis in a stress report.
- Other programs showing they also did not understand the basic engineering process by releasing designs for production before assessing, through analysis or test, whether they would work
- A stress analyst showing he did not understand how to relate analysis to reality by spending three months analyzing a bonded joint.

- Structural design engineers with over twenty years experience showing they could not solve a simple statics problem, let alone design structures with straightforward load paths.
- A manager showing he did not understand what he is managing by refusing to accept a subcontractor's stress analysis because it did not include color stress plots.

I could go on and on, and I'm sure you've thought of a few examples by now. Again, I'm not criticizing the people in our industry; I'm criticizing the system and the environment that have led to such a lack of understanding and to the costly problems and failures that have resulted.

As you may be able to tell, I feel quite strongly that overspecialization and lack of understanding are huge problems in the space industry. Let's look at a few truly effective ways to build understanding:

– **Rotate people through different specialized groups.** My favorite example, and of course the one that's closest to my roots, is the three groups contributing most to the design of structures: Stress Analysis, Dynamics and Loads, and Structural Design. Actually, I should add another group: Manufacturing. Most companies are organized with distinct groups in this manner. Perhaps it's unreasonable to suggest we combine these groups into one discipline, but let's resist the notion that someone should be, say, a stress analyst or a design engineer for life. Only through analysis can someone learn to understand structures well enough to design simple ones with efficient load paths; only through a sound understanding of manufacturing processes and tools can someone design things that are easy to build; and only through understanding design objectives and constraints, as well as other technical considerations, can someone do an efficient analysis.

  This is a problem not only in the structures area. Many companies have proposal-team specialists who go from one proposal activity to another without ever having to live with the constraints their proposals drive. We have specialists who do nothing more than write requirements—people with little or no understanding of the technical fields their requirements impact and who repeat the same mistakes over and over. There are countless other examples of how overspecialization drives problems.

– **Keep engineers on the program from cradle to grave, or at least until launch** (and, for some, until the system is deployed and operational in space). I know this isn't practical for everyone, but it becomes much more feasible with people who are willing to wear more than one hat. A good example is Lou Arthur, who is now retired after 22 years at JPL and 14 at Martin Marietta. In the sixties and seventies at JPL, Lou would start a program as a subsystem design engineer. Once his design was released, he became a manufacturing engineer, responsible for building, assembling, and testing the subsystem. When it came time to integrate and test the spacecraft, Lou put on his bunny suit and became an I&T engineer. You can imagine that Lou and his coworkers, after having to build, integrate, test, and fly their own designs, designed a much better spacecraft the next time! Lou never went to college, but he's one of the best engineers I've met.

- **Bring manufacturing back to your facility.** There is no denying the short-term benefits of subcontracting manufacturing to established shops rather than maintaining the facilities and skills in-house. Long-term, though, we pay an awfully high price. It used to be that design engineers had to walk only a hundred yards or so to see their designs being built. As a result, they were in the shops all the time, learning how to make future designs better. Now it's common for design engineers to have to board a plane and fly cross country to see their products built, and often they never do.

- **Bring development testing back to the design process.** I know there is a current push to replace hardware testing with virtual prototyping, or virtual testing. This may work with software—and even certain hardware characteristics—but it's a formula for disaster for most hardware aspects. Typically the people who have little hands-on test experience are the ones crusading to eliminate hardware tests. Those with experience are adamant about keeping them because they've learned to understand that certain things are not predictable, such as the effects of random variation and dimensional imperfections. Testing humbles us and gives us a healthy respect for uncertainty. The real world is far from the idealized world we create in computers.

  For the first five or six years of my career, I did stress analysis with no involvement in testing. I was highly rated by my superiors, and I considered myself a good analyst. Then we started testing things I had helped design, and my value as an engineer to my company immediately doubled. It doubled again when we did some low-cost development testing to support the design of new hardware. From my previous test experience, I was able to recognize the limitations of our analysis, and I convinced management to fund the development tests. The results were surprising, and we all learned from them.

  Development testing is not just for the design engineers and analysts; it also helps the manufacturing group iron out their processes. Computers can't predict the effects of process parameters, such as temperature, cleanliness, and workmanship.

  Development testing does not have to be expensive, and it should not be. We tend, in a large organization, to construct roadblocks that make it so. Rather than eliminate testing simply because it has gotten to be so expensive, we must get at the root of the problem and eliminate the things that make it expensive.

- **Enable and encourage knowledgeable engineers to teach the less experienced ones.** Our natural tendency is to ride our best engineers for all their worth, reaping the short-term benefits. Then, finally burned out, these people retire, and we are left with nothing. Many engineers don't want to teach; they want to do as much engineering as they can—but this isn't always the case. One of the most effective, knowledgeable engineers I know wanted to float from program to program, helping people out and "spreading pollen like a bee." It would have been a great use of his time, but it would have required funding from overhead, which of course made it impossible. Instead, a program put him in a managerial position, and he spent much of his time working raise plans and offload schedules. Not having fun anymore, he retired at the age of 55.

– **Develop or find effective courses that supplement rather than replace the above.** All organizations are limited in how much they can spend on education and training. If given a choice, choose education first. It makes little sense to send everyone in the department through training programs for the latest software, year after year, when many lack understanding of sound engineering.

If we want effective people, we must invest in them. We must be willing to take the lumps short term, such as increased overhead. We must have the goal of staying in business more than a couple years, and we must have confidence that our investments will pay off.

One of the common arguments against investing in the education of an employee is that the employee may leave and join a competitor, who then would benefit from your investment. But, if your organization follows the rest of the principles for doing things right, your people won't want to leave.

Having addressed most of this article to management, I need to take the other side for a moment. If you're an employee waiting for your company to hand you educational programs on a platter, wake up! There's no guarantee you'll be with that company a year from now. If you've spent fifteen years doing the same thing and never took the initiative to learn more and improve, and now you find yourself looking unsuccessfully for a new employer, you have only yourself to blame. Being unwilling to use personal time for education or to perform different roles at your company may be the path of least resistance, but it hurts yourself in the long run. It's kind of like cheating your way through college.

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.