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.

Article #3

Common Problems in Developing a Space or Launch System

Part 1

March, 2000

Tom Sarafin

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

We have a serious problem in the space industry: our products cost too much and they fail too often. Our attempts at reducing cost and padding quarterly profits have led to more mission failures, demoralized staff, and declining public confidence. Superficial fixes such as cutting budgets and reorganizing are not the answers. They have instead compounded the problem. To an outsider, our actions, stemming from a short-term focus, must make it appear that we are trying to run ourselves out of business.

If we are serious about saving the space industry, let alone our own organization, we must identify and attack the roots of the problem. In last month's article, I addressed some of the key reasons space missions are so difficult. Let's look now at some of the problems that appear repeatedly in space programs, problems that drive cost and risk without benefit. By so doing, we'll be more apt to identify and address the things that hinder or preclude success.

If you've spent much time on programs developing space or launch systems, chances are you've experienced most of the following problems:

- **Requirements that provide little or no value**

Example: After a class I was teaching, in which the subject was developing effective requirements, an engineer told me about a requirement levied by her program's customer that she was now going to challenge. She was designing an electronics module for a spacecraft prime contractor, and the requirement was to meet certain mass moments of inertia about three orthogonal axes. I responded that specified limits on moments of inertia were not uncommon. She said, "No, you don't understand. We're not supposed

to stay below those moments of inertia, our module has to have them." She went on to tell me about how they were adding ballast to the module and otherwise complicating its design to satisfy the requirement. She did not understand the purpose of the requirement, but she was now going to find out. She asked if I knew why, but all I could do was guess. One of her customer's engineers probably had a math model for controls analysis or loads analysis and had realized his job would be easier if he could simply assume certain mass properties for the module and then have the supplier build it that way. Otherwise, he might have to iterate his analysis.

If my guess was right—and I can think of no other reason—this was but one more example of an engineer who, by trying to simplify his or her job, causes ten or more times the work for someone else downstream.

Of course, we've all seen examples of a more common reason for questionable requirements: trying to gain some small benefit without regard for cost—and often without understanding how little benefit will be obtained. Many times, it simply comes down to a misunderstanding. I'm not sure which reason applied to the following example, which I found at the same company at which I got the above example, during the same class.

After class later in the week, one of my students, a stress analyst, asked if I could help him interpret strain-gage data acquired during a centrifuge test of an electronics module. The test had been done four months earlier, and, before their customer would approve it, they had to explain each plot of strain versus acceleration. They had used 280 strain gages and ran several tests. After four months, the stress analyst was at a loss to explain several nonlinearities in the data. He proceeded to show me some apparently nonsensical plots and ask if I knew what they might mean.

I stopped him then and said, "Time out. Weren't the success criteria that, after being spun up on the centrifuge arm to the required loads, the module still worked when it was powered up, and that there was no permanent deformation that violated any specified envelope? Where did the requirement come from to have a linear relationship between strain and load throughout the module?" I've been involved in many structural tests, and one thing I can almost always count on is having some strange data plots as a result of friction, localized nondetrimental yielding, or some other cause. The key question is whether the product functions as needed. Having to find an acceptable reason for every bit of anomalous data is a perceived requirement, not a true requirement, perceived in this case by both the contractor's engineer and the customer's engineer.

A bit of advice on the side: if you're planning a test, think through the instrumentation carefully so that you don't end up with useless data.

- **A key issue not being addressed because there is no requirement to do so**

Example: At a spacecraft design review, a customer representative asks the product team lead what will be done to confirm the assumptions used in the thermal model to predict temperatures. The team lead responds, "There's no requirement to validate the thermal model."

If you've spent much time working with some of your organization's contractors, you've probably run across this type of response. You've identified a valid technical issue during product development, and the response is that there is no requirement to address it. Clearly something is wrong here; the contractor has lost sight of the true requirement, that the product be able perform to some level during the mission. A thermal model is a tool used by the contractor to help show that the product will do just that. Making sure the conclusions drawn from that model are valid is simply sound

engineering. Since when does a customer have to specify that the engineering must be sound?

Refusal by your contractor to address a valid issue because "it is not a requirement" is a clear indication that you have overspecified the requirements. When we tell our contractors not only what their products must do but also how they must demonstrate to us that the products will do it (verification and quality assurance), we take ownership away from the contractors. The more we do this, the more we invite the contractor to treat the specification as a checklist. Specifying requirements and building contractor ownership will be the subjects of a future article, the writing of which I will enjoy immensely.

- **Customer demanding more analysis or testing than the contractor had intended**

Nearly every program has disagreements between customers and contractors regarding what verification must entail to be considered adequate. Does the thermal model require validation? Do all the strain readings from the centrifuge test need explanation?

Such disagreements occur most often late in the program, when changes are most expensive, and discussions can become quite heated. This is a lose-lose situation. If the customer wins the argument and forces the issue, the contractor loses ownership, and the engineers lose morale. This in turn degrades quality. If the contractor wins, with the customer in the end giving in under protest, the customer is unhappy and nervous about launch. Isn't quality supposed to be about pleasing the customer? So we try to negotiate an acceptable compromise, but this is not easy late in the program, especially for fixed-price contracts.

The solution that has gained recent popularity is for the customer to back off and leave verification to the contractor. Let's assume we're the customer paying a gajillion dollars for a space mission or some complex part of a space system. According to this popular philosophy, we are supposed to trust without oversight that the contractor will get everything right—despite the fact that the contractor has far less at stake than we do. We also know that, given the present environment, the contractor was probably forced to bid low in order to win the job. And remember: once we launch, this complex system has to work. We won't get a second chance.

I don't know about you, but this isn't an approach I would be willing to take as customer. Go back to my second article and read the closing paragraphs. Excluding the customer is not a viable approach to verification. We can't make the problem of disagreement over verification go away that easily. We have to get to the root.

Given how we normally start a space program, it's not surprising that this problem arises so frequently. We routinely award contracts without a clear definition of verification. We'll decide, perhaps, that a centrifuge test is needed, but nowhere do we document how strain gages will be used or what will be done with the data. As customer, we may assume the thermal model will be validated, but our contractor may not have recognized the need when making the bid.

Like it or not, verification in this business is subjective. It's also expensive. If we allow it to stay subjective throughout our program, disagreements that drive even more expense are unavoidable.

To remove subjectivity, we need a sound and thorough verification plan, complete with criteria, that is acceptable to both parties. And that plan—at least a good preliminary version of it, spelling out the intent—must be in place before the final contract is awarded. Of course, removing subjectivity is one of the main goals of our specifications, but the main difference here, for the purpose of clearly establishing

ownership, is that the verification plan must be the product of the contractor. I will explore this topic more in a later article.

Table 3-1 summarizes common viewpoints from customers and contractors regarding the above three problems.

Table 3-1. Why Do We Have so Many Problems with Requirements and Verification? The answer depends on perspective. The true reasons for these problems probably have more to do with poor communication, a lack of understanding, poor cooperation, and practices that take responsibility and ownership away from contractors.

Problems:	
• Requirements that provide little or no value • A key issue not being addressed because there is no requirement to do so • Customer demanding more analysis or testing than the contractor had intended	
Causes:	
Customer's point of view: – Our contractor didn't take time to understand the requirements. – Our contractors don't care about quality and mission success. – If we don't specify it as a requirement, they won't do it. – Our contractors are idiots. – We have to tell them how to do everything and watch them like a hawk. – We don't trust their analysis, so they'll have to do a test.	Contractor's point of view – Our customer is overspecifying, and the requirements are poorly worded. – If we had more information and freedom, we could provide a better product at lower cost. – But since we don't, who cares? – Just bid according to the specification, and then do the least amount possible to fulfill it. – The only way we won this job anyway was to bid low. – We don't need the test. If you want it, you'll have to give us more money.

Let's look at more common problems:

- **Costly design changes after drawings are released**
- **Designs that unnecessarily constrain or penalize other designs**

and

- **Products that are difficult and costly to build, integrate, and test**

These problems are closely related in that they result from releasing designs for production before we've adequately assessed them or considered downstream activities. Example: after manufacturing is well underway, an engineer suddenly identifies a design change is needed to enable the product to mate properly with another component.

For many space programs, faster-better-cheaper means reducing design time so much that the engineers can't finish their analysis (which is supposed to establish confidence) before releasing the design, let alone design the product to make it easier to build and test. In one of my recent classes at a space organization, I mentioned "concurrent engineering" and received a chorus of groans. I asked what was wrong with concurrent engineering, and an engineer replied, "Oh, that's when you have to sign out

the drawing before your analysis is done. You're supposed to do the analysis while the product (flight hardware) is being built."

Now, how exactly did we take a concept as good as concurrent engineering and bastardize it that way?

Releasing designs before they're ready (or even before we've fully defined the system's requirements) is not a new problem; we've been doing it long before the "faster-better-cheaper" mentality became popular. Consider the U.S. Space Shuttle Orbiter. Payloads attach to the Orbiter with trunnion pins inserted into latch bearings. The latches mount on bridges that span fuselage frames, on both side longerons and at the Orbiter keel. The basic mounting schemes consist of three-point (two longeron, one keel), four-point (three longeron, one keel), and five-point (four longeron, one keel) mountings.

At the time the design of the mid fuselage was released for manufacturing, the mechanical interface between payloads and the Orbiter was not defined. Longeron and keel bridge structures became the easiest solution. However, because each frame-to-frame segment of the mid fuselage is unique, a unique bridge is required. We lost the opportunity for common hardware.

The simplest way to attach a payload is to cantilever it from one longitudinal station with three-point mounting, but, for most payloads, this approach would lead to a longitudinal interface load at the keel that exceeds its capability. Because the design of the Orbiter structure was released before the payload interface was defined, the keel structure was not designed for a longitudinal load.[1] Thus, most payloads must attach to the Orbiter at two stations, usually with a five-trunnion support configuration. (See Fig. 3-1.) As a result, many payloads for the nearly 100 Shuttle missions have been penalized with additional mounting structure. And the Orbiter has carried the extra trunnion latches (between 28 lbs and 113 lbs each, depending on type and strength) and longeron bridges (between 50 lbs and 140 lbs each). Meanwhile, we have spent many millions of dollars scalloping weight out of the Shuttle's external tank.

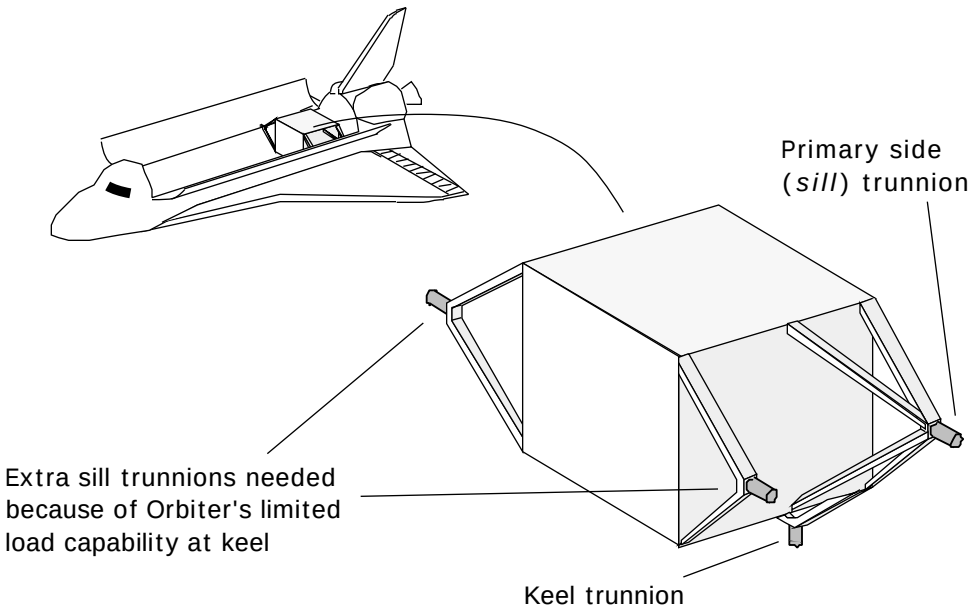

Fig. 3-1. Many Payloads Must Attach to the Shuttle Orbiter at Two Stations Rather than One to Keep from Overloading the Orbiter's Keel Structure.

[1] The Russians copied much from us, but they also learned from our mistakes. The Buran, their orbiter, was built with a keel spar, offering generous longitudinal load-carrying capability.

Problems arise for many programs during vehicle integration and test, handling and transportation, and integration with launch vehicles. Such problems include difficult assembly, expensive tooling, complex test procedures, and incompatible mechanical and electrical interfaces. After having supported several programs from preliminary design until launch, and after talking to hundreds of people from many other programs, I can safely say that most engineers on programs after design release spend the majority of their time dealing with problems. For many people, it's all of their time. And the most common reasons for these problems are inadequate design (or requirements) and inadequate planning.

Table 3-2 summarizes common viewpoints from engineers and managers regarding the above problems.

Table 3-2. Why Do We Keep Having Problems with Post-Design Activities such as Manufacturing, Vehicle Integration, and Test? These problems are symptoms of inadequate design, whether the result of inexperienced or unqualified engineers (perhaps too specialized?), too little time and money invested in the design, poor communication, or some combination.

Problems:

- Costly design changes after drawings are released
- Designs that unnecessarily constrain or penalize other designs
- Products that are difficult and costly to build, integrate, and test

Causes:

Engineer's point of view:	Management's point of view
– We don't have time to do things right. Priority is always on releasing the designs for production as soon as possible. – If we had more time, we'd design things that are less costly to build and test. – The manufacturing group is in the building across the street, and they're working to a different schedule. How are we supposed to work with them? – Management expects us to be more efficient, but how? Who will teach us? Where is the training budget? Where is the R&D budget?	– Our engineers have to get more efficient and start talking to each other. – If we gave them more time, they'd just put more elements in their finite-element models. – Cost is proportional to time on the program. If we compress the design schedule, the engineers will find a way to get their work done and we'll save money. – Analysts just cause problems. It's better to leave them off the team until after the drawings are released. – We have no choice but to rush the design. Our customer is demanding lower cost and shorter development time.

Well, I've found that if I make these articles too long, no one will read them. So I'll stop here and continue the discussion of common problems in my next article.

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.