TESTING AS PART OF A SOUND ENGINEERING APPROACH

Presented at the 22nd Space Simulation Conference[1]

October 21 - 24, 2002

Thomas P. Sarafin
Instar Engineering and Consulting, Inc.
tom.sarafin@instarengineering.com

ABSTRACT

This paper is a reminder of the roles of testing in a sound and cost-effective process for developing space-mission products. Topics include development testing, qualification testing, acceptance testing, and analysis-validation testing. Emphasis is on why these tests are done, related to both mission success and cost-effectiveness. The purpose of this paper is to help refocus organizations and people in a time when many perceive tests as requirements (and thus do not do them when not "required") rather than tools for product development.

INTRODUCTION

In the recent push by space programs to do things faster, better, and cheaper, probably no activity has been under attack as much as testing. The common perception is that testing is expensive. Often it is. But there is a far greater cost—that of the mission itself—that testing can help us avoid. Even expensive testing can be cost-effective.

A disturbing extension to this perception is that we do tests only because our customers demand them. A tactic for reducing cost that has been popular recently is for the space-mission customer to relax requirements pertaining to verification. This strategy has produced mixed results. Once a customer drops the requirement for a test, the contractor typically decides not to do it—even when the test makes sense. There are two likely reasons:

1. The clear message from space-industry leaders over the past decade or so that reducing cost is more important than ensuring a successful mission. Despite banners that have remained posted proclaiming otherwise, this recent emphasis is undeniable. It has in turn driven companies to promote the best cost savers, rather than the best engineers, into decision-making positions. Of course, deleting a test reduces cost, at least in the short term.

2. Many years of customers dictating which tests must be done rather than expecting contractors to figure such things out for themselves.

These two reasons together have contributed to a big problem in our industry: Many engineers and decision makers do not remember—or never understood—the reasons why tests were done in engineering projects before they became "requirements." The goal of this paper is to remind people of these reasons. For additional guidance, see MIL-STD-1540D (ref. 1), MIL-HDBK-340A (ref. 2), and GEVS-SE (ref. 3).

UNDERSTANDING VERIFICATION

Verification means establishing confidence that a system will perform as needed. In a space program, we need a great deal of confidence before launch because afterwards we probably won't be able to fix any problems. Thus,

---

[1] An earlier version of this paper was presented at the 20th Aerospace Testing Seminar, March, 2002.

what we consider to be verification ends with the decision to launch. Much of the cost our program will incur up until we deliver the system for launch will be in verification.

Criteria and standards for verification are needed to remove the ambiguity associated with "establishing confidence." To feel ownership, which is essential for a quality product, the people developing the product should be responsible for developing plans, criteria, and standards for verification. However, because the customer typically has a lot at stake in a space mission, those plans, criteria, and standards must be acceptable to the customer.

Planning for verification must start early for several reasons:

- 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.
- It's needed to scope the program and provide adequate funding and schedule.
- 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.

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. 1). Verification follows a building-block process, rooting out problems when it's most affordable to do so.

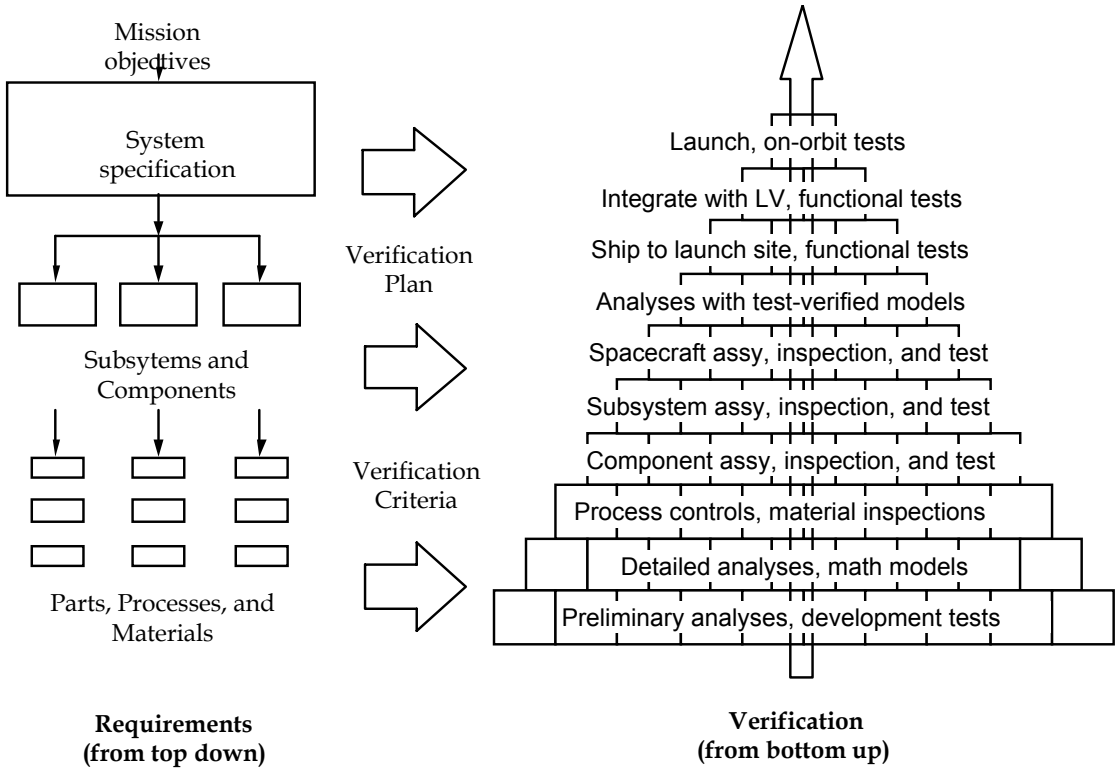

Figure 1. Requirements Flow from Top Down, and Verification is from Bottom Up. Verification is connected to requirements by the verification plan, criteria for design and verification, standards, and controlled processes.

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 your system and then instilling that confidence in your 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?

Thorough verification is simply good engineering. We should not 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.

Verification is essentially the same as quality assurance. Depending on how you look at it, either one could be considered part of the other. In the space industry, quality is reflected by the probability that the mission will succeed. For high quality, we must

- define the requirements accurately and completely,
- design the item to meet those requirements,
- build the item to its design specifications,
- and control the product's configuration up until launch.

Verification overlaps all of these activities. Mostly, though, what we think of as verification concerns the middle two activities in the above list: Have we properly designed and built the item? These are separate issues, and it's important that we understand the distinction. The key difference in verification is that, when the design is in doubt, we need to verify it only once, whereas, when the concern is with manufacturing, we should verify quality for each build.

VERIFICATION METHODS

Depending on the requirement, we can verify it by one or more of the following methods:

- Analysis
- Process control
- Inspection
- Test

Another method quoted is *demonstration*, but a demonstration is simply a form of test. Any test, including a demonstration, requires success (pass/fail) criteria.

Similarity is another method frequently used, but it is merely a combination of analysis and test: Our product is much like another we have tested, and we use analysis to show the differences are insignificant. Normally this means there are minor design differences, while the mission environments are the same (or less severe), but we can also use similarity for the same design under different environments, as long as sound analysis shows the environmental differences are inconsequential.

Be careful with similarity as a method of verification. We are often tempted to use it without justification. Without dependable analysis addressing the differences, claiming similarity as a method of verification is not sound engineering.

Proactive Versus Reactive Verification

Verification can be proactive or reactive (Table 1). With reactive verification, we try to find and fix defects after our product is built, and we wait for problems to surface before we address them. Proactive verification means anticipating problems and taking steps to avoid them.

Of the four basic methods of verification—analysis, process control, inspection, and test—the one we most associate with verification is test. Testing (or inspecting) the end product, though, is a reactive method of verification. In other words, although end-item inspection and testing are effective in detecting defects, they do not improve the product's quality. Our goal is to be more proactive in verification: build confidence with methods that will also build quality into the product. When used this way—as it should be—analysis is a proactive method of

verification. We use analysis to steer the design, to build confidence in the design before we commit to it. If we wait until after the design is released to do the analysis, which programs tend to do when they perceive analysis simply as a contractual requirement, then the analysis is reactive.

Table 1. Proactive and Reactive Methods of Verification and Quality Assurance. Proactive methods act to prevent problems, whereas reactive methods try to find them. Space programs tend to be reactive. If we want to reduce total cost and shorten the time to launch, we must become more proactive in verification. However, because of the economics of low-volume production, space programs will probably always need reactive methods as well.

Proactive Methods	Reactive Methods
Development testing Analysis (when done to steer the design) – supported by development testing Process control – supported by development testing and analysis	Inspection Analysis (when done after the design is released) End-item testing: – Qualification testing (one-time test of a flight-like product) – Acceptance testing (test each flight unit) – Analysis-validation testing (testing to obtain information used to confirm critical analysis)

Another proactive method of verification is *development testing*, which is testing to acquire information that will improve the design or the manufacturing process. We do development testing to reduce risk. The philosophy here is that early testing of representative hardware is (or should be) a low-cost way to detect and correct deficiencies when we don't have enough information to depend on our analysis. Unfortunately, many large organizations have grown internal bureaucracies that inhibit development testing. What is supposed to be a cost-effective tool for engineers has too often become something that is nearly impossible to get approved.

It's truly refreshing to find an organization at which such things are still simple. I'm presently helping the U.S. Air Force Academy design, build, and verify a small spacecraft for launch on the Shuttle. Because of the ease of building hardware in their own shop—and because of limited analysis capabilities—the team of cadets and staff decided to build and test a development unit for their spacecraft. To keep this unit simple and build it quickly, they decided not to install threaded inserts, instead planning to tap the holes and fasten into bare aluminum alloy. They did not know the pull-out strength for such tapped holes and didn't trust their analysis, so they didn't know how high to torque the bolts. I asked how difficult it would be to tap some holes in a spare piece of aluminum, install bolts, and then torque them until either the threads stripped or the bolts failed. "Not a problem," was the reply. We walked down the hall to the lab and, in two hours, we had the information we needed.

Controlling manufacturing processes is proactive, of course, but, to be a sole method of verification, our controls must be completely effective. Every manufacturing facility claims their processes are controlled. No one wants to admit otherwise. The important question is, "controlled relative to what?"

To control manufacturing processes well enough that we don't have to inspect or test the end item, we must have meaningful statistical data, obtained through testing, that demonstrates such control. We can't simply write a procedure specifying the process parameters, such as ranges of temperature and humidity. We must do development testing to demonstrate that those controls keep the products' critical characteristics—such as dimensions, surface finish, or strength—within acceptable ranges or at sufficient levels.

W. Edwards Deming, the man most credited with teaching quality to the Japanese after World War II, believed strongly in the importance of process control through statistical methods [ref. 4]. One of Deming's arguments was that it's wasteful and costly to churn out products with uncontrolled processes and then screen out defects through inspection. Another was that inspection as the means of quality assurance doesn't work. Experience bears this out. The next time you write a document, hand it to an office mate for an inspection, with the objective of finding typos

and misspellings. Then enlist a second inspector, and see if the first one found all the errors. Most likely, you'll see what Deming was telling us: Because of human error, inspection alone doesn't work.

Despite the desire to eliminate reactive methods of verification, the most cost-effective approach when production volume is low, such as in the space industry, is to mix proactive and reactive methods. Acquiring the information and data needed to make analysis dependable and to control processes completely is costly. We usually don't have the production volume needed to justify that cost. Thus, we try to find the best balance of analysis, process development, and product inspection and test.

End-Item Testing

As with inspection, testing products before use is warranted in the space industry, even when we've done thorough analysis supported by effective development testing. One reason is the high cost of failure combined with lack of accessibility after launch; another is that experience tells us certain characteristics, such as resistance of electronics to random vibration or shock, cannot be accurately predicted with analysis. A third reason is variation resulting from manufacturing processes that are not completely controlled. Inspection can find some defects, but many can be found only through testing.

We do three types of tests at the end-item level:

> ***Qualification test***—a test intended to demonstrate the adequacy of a design when analysis alone is not dependable. The *qualification article* is a unit dedicated for test but built to the same recipe (same design and same processes) as for the flight unit. Qualification environmental testing is done under conditions more severe than those we expect the flight unit to see in service, with the difference in environmental level called the *qualification margin*. This margin is intended to provide confidence that the follow-on products—the flight units—will be able to withstand the maximum expected (*limit*) flight environments, given the unavoidable build-to-build variation.
>
> ***Acceptance test***—a test intended to demonstrate the adequacy of a product when we are not certain the qualification margin is high enough to account for build-to-build variation. We often do not have meaningful statistical data that shows our manufacturing processes are controlled well enough to keep variation within the bounds of the qualification margin. In such a case, we structurally and environmentally test each flight unit to acceptance levels, which are the same as limit or slightly higher but considerably less than the qualification levels.
>
> ***Analysis-validation test***—a test done to acquire information needed to confirm critical analysis upon which verification is dependent. An example is a modal-survey test. Even if we test the vehicle structure to verify strength, the applied loads probably are based on analysis with a finite-element model. The model, though, doesn't perfectly represent the structure—its accuracy is based on the assumptions of the engineer—and thus the loads it predicts are in error. To minimize the error, we do a *modal-survey test*, in which we individually excite the structure's key modes of vibration and measure mode shape, frequency, and damping. We then correlate the finite-element model with the test results and repeat the loads analysis. *Thermal-balance testing* is an analogous way to validate the model used for predicting temperatures.

Many programs combine qualification and acceptance testing on the first-built flight unit with an approach called *protoflight*. Instead of building and testing a dedicated unit, we test the first-built flight unit to levels that are usually lower than qualification but higher than acceptance. Depending on the type of construction and on the environment, we then might test each follow-on flight unit to acceptance levels.

The protoflight approach can be risky, especially for the first-built unit. The reason we don't normally fly a qualification unit after testing is that the materials within the unit have incurred unknown fatigue damage. The same applies to a protoflight unit. Even though the protoflight environments (and duration of exposure) may be less severe than the qualification environments, without doing the qualification tests we have no assurance the design can withstand those environments. For all we know, the protoflight unit may have just barely passed its tests, and little fatigue life might remain for the upcoming mission. To reduce this risk and build confidence in fatigue life, we should plan to do more extensive development testing and fatigue analysis when using the protoflight approach.

Remember: Having sufficient fatigue life is a requirement for all hardware products, whether or not anyone remembered to specify it.

Although the above discussion is geared toward hardware, qualification, acceptance, and protoflight testing also apply to software. If we develop software a single time and then use it for several missions, with nothing different, we need test the software only once with a protoflight approach. If, for each mission, we must re-enter the code or the input, or if we reassemble software modules, acceptance testing becomes important to protect against human error.

PLANNING FOR VERIFICATION

A common question on a space program is, "How do we develop an effective verification plan?" We want to ensure a successful mission, of course, but we don't want to spend any more than we have to. Most of us have been on programs in which, in hindsight, many activities provided questionable benefit.

Planning for effective—yet concise—verification and quality assurance isn't difficult if we truly understand the problem, including

- the product's requirements (what the product must be able to do and under which constraints must do it)
- engineering principles (how things work)
- potential failure modes (what can go wrong)

To build understanding, let's look at the logic flow for verification. If you accept the above definition of verification as "establishing confidence," we certainly begin verification during conceptual design, so we'll start there with the logic flow. The logic described here applies to hardware development, but much of it pertains to software as well.

To assess alternatives during conceptual design, we must estimate their costs, much of which would arise from establishing confidence that they would work. Early planning for verification not only enables these estimates, it also is essential for us to adequately scope the program so we'll be able to do things right. There is no more important time to anticipate and avoid problems than when we are selecting a design concept.

For a proposed concept, the question we must address is, "How well can we predict the product's key characteristics?" The answer depends on our manufacturing processes, our engineering database, our tools, and the skill of our analysts. To predict a product's characteristics, we must be able to predict what the manufacturing processes can achieve at low levels of assembly and then extrapolate with analysis the characteristics at the subsystem and system levels. The key is to recognize when such extrapolation is not valid.

The X-33 program was cancelled in early 2001 after more than $1 billion had been spent on it. The event that contributed most to the program's demise, even though it happened about a year and a half before cancellation, was test failure of the composite propellant tank. Given the ultimate consequence of this failure—death of a billion-dollar program—we have to ask if this flaw, whether in the design or in the manufacturing process, could have been found in development testing, when failure would have had little impact. An article in the November 15, 1999, issue of *Aviation Week and Space Technology* raises this question by sharing the opinions of several engineers on the X-33 program that development testing had been inadequate. According to the article, the program manager defended the approach taken and criticized a statistical study that indicated the manufacturing process was insufficient. His argument was that such studies are not appropriate for experimental programs. This argument is flawed. Any expenditure of over $1 billion more than justifies a sound engineering approach.

If we can't afford the time or expense of the development testing warranted for our design concept, we should change the concept. This usually means finding a simpler design. For example, use of composite materials in a new (to us) structural configuration almost always warrants extensive development testing at incremental levels of assembly. When there is no time or money to develop and confirm processes, we should try to design the structure with a ductile metal alloy instead, which will be more tolerant of process variation.

Testing as Part of a Sound Engineering Approach

The goal of verification during conceptual design is to become confident enough to commit to the concept and proceed with full-scale development. Figure 2 shows the verification logic flow during conceptual design.

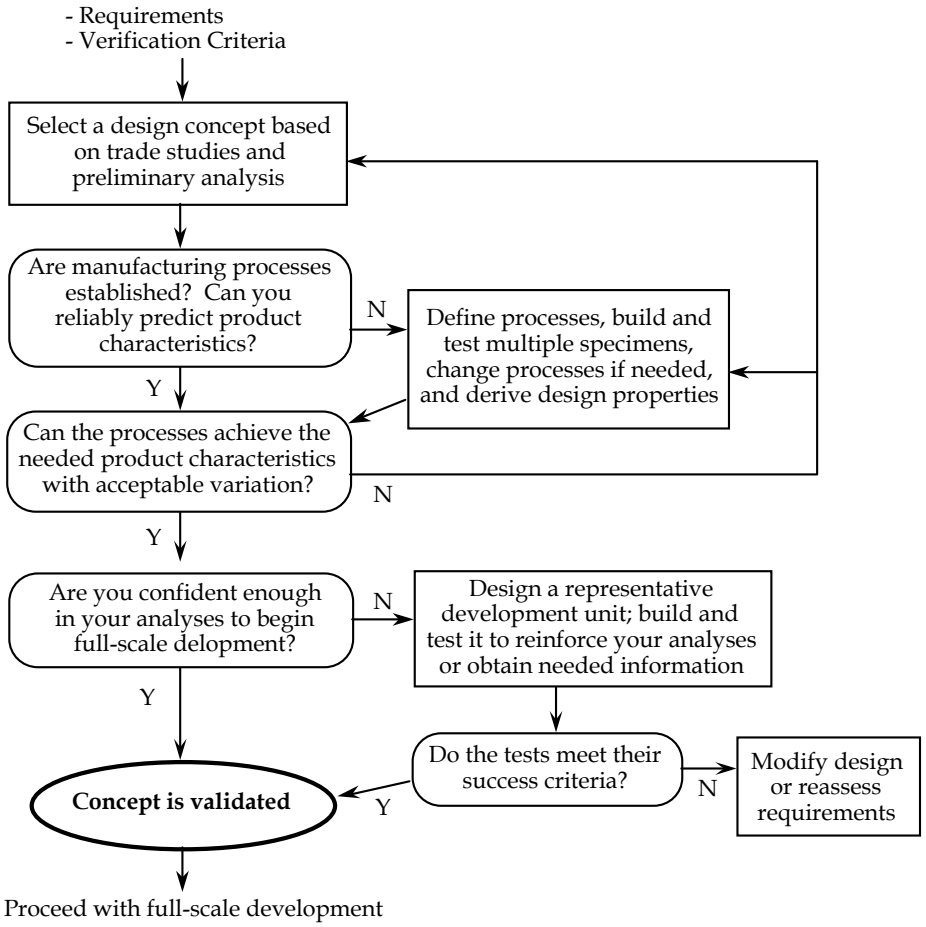

Figure 2. Verification Logic Flow during Conceptual Design. The design concept defines the appropriate extent of development testing. (Adapted from reference 5, Fig. 11.2.)

The design phase of full-scale development culminates with the Critical Design Review (CDR), for which the goal is to agree, contractor and customer, upon the design and commit to production. To obtain such a commitment, the contractor's engineers must instill confidence. In other words, they must, to the best of their ability, verify requirements by analysis, supported by any additional, needed development testing.

When verifying a requirement by analysis, it's particularly important to have well thought-out criteria for that analysis. Examples include thermal margins, weight-growth allowances, and structural factors of safety. Such criteria are intended to ensure analysis quality, given uncertainty and the human element, and also to ensure a high probability of success, given random variables.

We use these criteria to judge the acceptability of analysis as our basis for committing to production, but the decision is easy only if the criteria are met. If not, we still might decide to accept the design and build the product.

Criteria are not the same as requirements. A criterion relates to risk, whereas a true requirement is something the product must be able to do or a characteristic the product must have. Failure to meet our criteria simply means there is more risk than we originally intended to accept. But, the later we go in the program before uncovering the risk, the more time and money we have invested and the more costly it would be to start over. Before accepting this consequence, it makes sense to assess the risk implied by failing to meet the criteria.

Risk is quantified by two things: probability of failure and consequence (cost) of failure. If we can estimate these two things, we can multiply them by each other to determine the *expected cost of failure*, which is the amount we should be willing to spend to eliminate the risk. We can think of the expected cost of failure as a cost that is amortized over many risk decisions. Insurance premiums are based on this approach.

This is the key to the effective use of criteria. We use them to steer the design, but we can't get emotionally attached to them—and we can't let them become a "black box" that no one truly understands. If a criterion is not met (e.g., negative structural margin of safety), we need to strip away the jargon and strive to understand the risk.

Figure 3 portrays the verification logic flow during full-scale development. The conclusion that justifies proceeding to manufacturing—that the design meets requirements—is a preliminary conclusion, as discussed below.

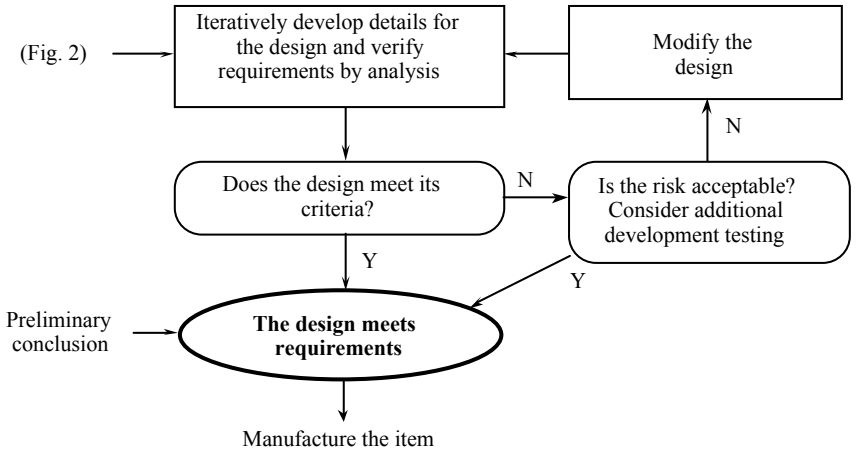

Figure 3. Verification Logic Flow during the Design Phase of Full-Scale Development. During design, confidence comes from analysis. Often that confidence is only to bridge the span until end-item testing. (Adapted from reference 5, Fig. 11.3.)

In manufacturing, the objective is to build something that conforms completely to the specified design. Because processes, raw materials, and people aren't perfect, things can go wrong. The first decision regarding verification of manufacturing quality concerns how well we'll be able to control the process and how much time and money we're willing to spend learning to do so better.

An inspection is similar to an analysis or a test in that it requires pass/fail criteria. Just as we don't discard a design based solely on failure to meet the criteria, we don't necessarily scrap hardware that fails inspection. Again, failure to meet the criteria simply means more work is required. We write a discrepancy report to document the problem and then get the right people to assess it. Sometimes their revised analysis will still satisfy its criteria, even though some of the product's features are not within specified tolerances. If this is not the case, we still might accept the product, depending on our assessment of risk. Figure 4 shows the flow of verification logic during manufacturing.

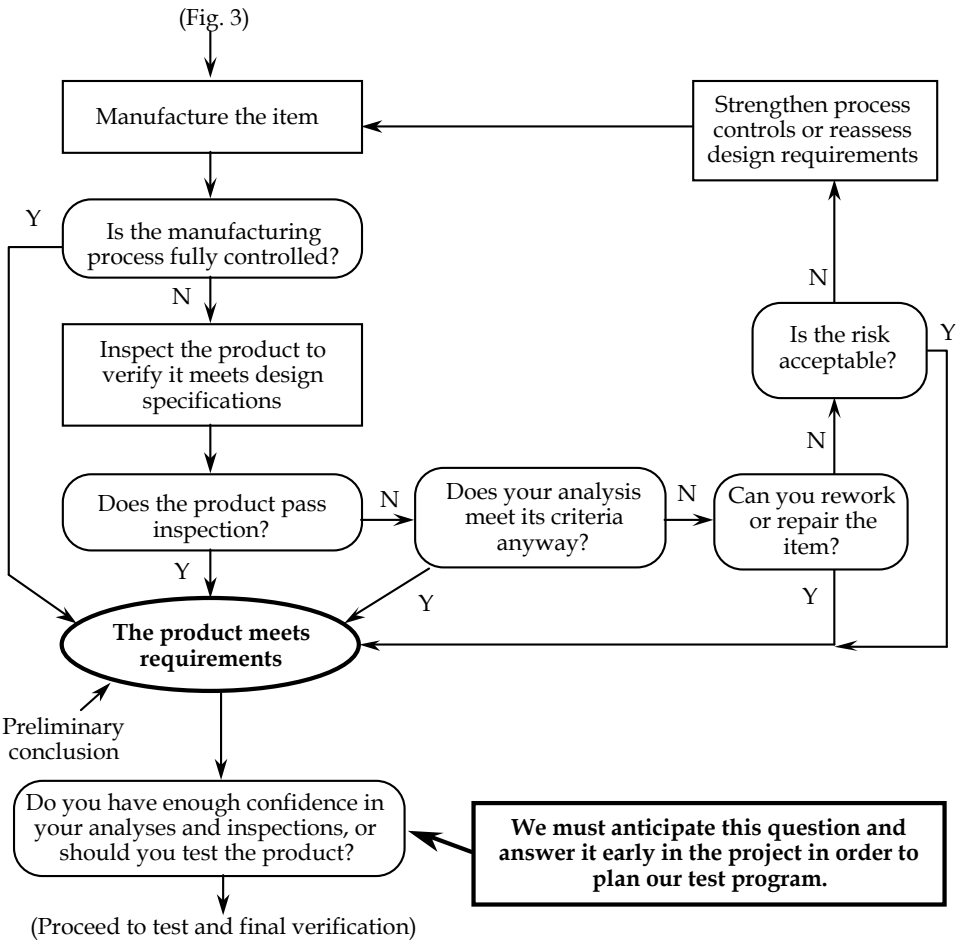

Figure 4. Verification Logic Flow during Manufacturing. Verification continues through manufacturing by use of inspection and process monitoring. (Adapted from reference 5, Fig. 11.4.)

Note the conclusion we are trying to reach after manufacturing: The product meets requirements. As noted in the flow chart, this is a preliminary conclusion, as was our decision going into manufacturing that the design meets requirements. Nagging doubts may persist. If, for some reason, we are not confident enough to rely solely on our analyses and inspections, we test the product.

Of course, we can't wait until this point in the program to decide whether to test the product. We must project ahead when planning for verification in order to ensure adequate time and budget. We go through the logic described above, based on our design concept, the planned method of construction, the experience of our staff, our organization's experience with this type of design, and the current or expected state of process controls.

As shown in Fig. 5, there are three possible reasons for testing an end-item product. When we don't trust analysis that is critical to our program, we do an analysis-validation test. If the analysis is in doubt, the design based on that analysis is usually in doubt also. In this case, we ideally would do a qualification test on a dedicated, nonflight unit. When the process controls are in doubt, we test each product for acceptance.

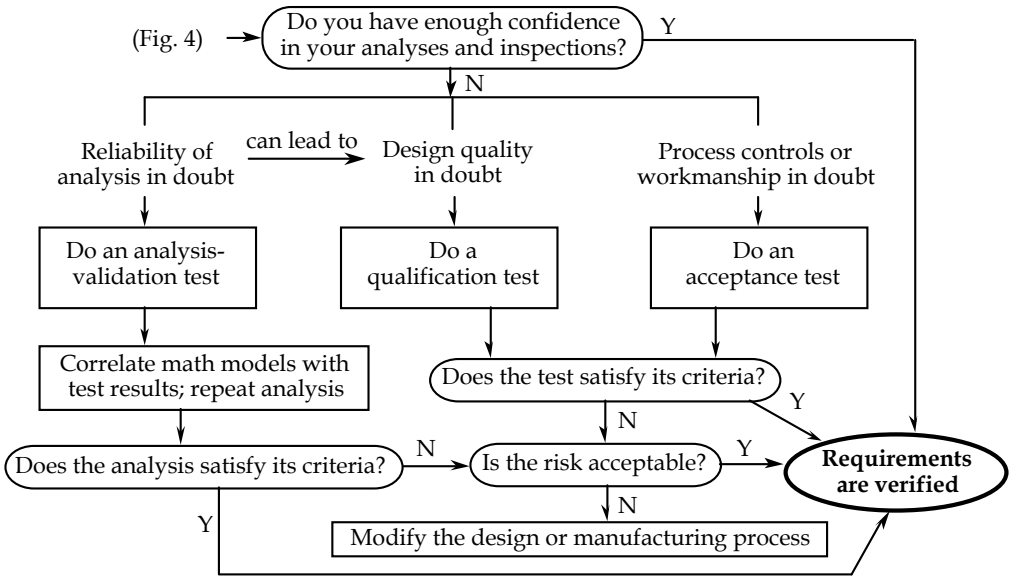

Figure 5. Logic Flow during Final Verification. End-item testing is warranted when the dependability of analysis and process controls is in doubt. (Adapted from reference 5, Fig. 11.5.)

We often refer to an acceptance test as a "workmanship" test, but let's stop and consider the message this terminology sends to the workers, or technicians. We essentially are saying we don't trust them to do a good job. W. Edwards Deming said that, if quality is poor, it's not the fault of the workers, it's the fault of the system or process. In other words, something outside the control of the worker, such as poor-quality raw materials or tools, is usually at fault. Workmanship is only part of the process, and we do acceptance testing when we don't fully trust the process.

It's quite likely that, for a given product, we'll need to do all three types of tests: analysis-validation, qualification, and acceptance. For each test, note the role of criteria shown in Fig. 5. In all cases, failure to meet the criteria means we must assess risk.

So, when budgets are getting squeezed, how do we go about reducing the cost of a test program? As stated earlier in this paper, it's a mistake to desert a sound engineering process, such as that shown in the above logic flows, in order to reduce cost. If end-item testing is too expensive, we invest up front in development testing and in finding a simpler design in which we can be confident with minimal testing.

Planning to Avoid the Need for Acceptance Testing

Increased build quantity is a situation that is becoming more common as more uses for satellite constellations are conceived. If, say, we were planning to build 100 spacecraft of identical design, we would like not to have to test all of them. A reasonable plan would be to test a dedicated qualification unit to confirm the design, test the first four or five flight spacecraft for acceptance to build confidence in the process controls, and then fly the rest without testing. Before adopting such a strategy, though, we should think it through by trying to answer the following questions:

- How will we control manufacturing processes?
- How will we know if processes are controlled well enough to support our plan?
- What happens if environmental testing finds problems in one or more of the vehicles tested?

As we consider such questions, we recognize our plan isn't as simple as it first appears. If failures do occur during environmental testing, as a result of design or process deficiencies, our program may be in trouble. If the

dedicated vehicle fails qualification testing, will we have time to change the design, or will we have already built several flight vehicles? If qualification testing is successful but one or more failures appear during acceptance testing, it would be clear our processes are not adequately controlled, but we won't have the money or time to improve them or to test the follow-on spacecraft. Budget and schedule would have been based on the plan of testing only the first few.

This scenario helps drive home the importance of extensive process development, supported by statistical data collected from development testing. Even if the answer is to build, say, an extra five spacecraft to make the constellation redundant, the same issue exists. Only with meaningful data can we decide whether our processes are controlled well enough to support our plan. Deciding to test only the first few, even with a redundant system, without supporting data is not sound engineering. It's irresponsible.

Similarly, any test that you are considering deleting can be deleted responsibly only through a sound understanding of its purpose and how it fits into the above logic flows. Said another way, if you don't understand why a test is normally done, you can't responsibly decide to do away with it.

SUMMARY

Someday we may learn how to eliminate the variation that is inherent in materials and manufacturing processes, to develop foolproof methods of analysis, and to breed humans who never make mistakes. Until then, testing space products before launch will be necessary. Our product may be defective, and, if it fails during the mission, we probably won't be able to fix it. Norm Augustine, former head of Martin Marietta, once compared launch to heart surgery: We have to get it right the first time.

Unquestionably, some tests are not worth doing. Some we do only because we're required to do them, without believing in their value. Others prove their worth by detecting flaws. Sometimes we discover that we went too long in the program before finding those flaws—we can no longer afford the time or money to fix them. To console ourselves, we resolve next time to reduce risk incrementally with a building-blocks approach, starting with development testing.

To plan the best test program, we should learn from the wisdom acquired in past programs but resist following "black box" processes and standards that no one understands. Understanding the problem is the key to establishing a cost-effective test program, and understanding a test's true objectives is the key to planning a good test.

REFERENCES

1. *Product Verification Requirements for Launch, Upper Stage, and Space Vehicles*. MIL-STD-1540D. Department of Defense. January, 1999.

2. *Test Requirements for Launch, Upper Stage & Space Vehicles*. MIL-HDBK-340A, Vols I & II. Department of Defense.

3. *General Environmental Verification Specification for STS and ELV Payloads, Subsystems, and Components*. GEVS-SE. Goddard Space Flight Center. 1990.

4. Deming, W. Edwards. *Out of the Crisis*. Massachusetts Institute of Technology. 1986.

5. Sarafin, Thomas P., ed. *Spacecraft Structures and Mechanisms: From Concept to Launch*. Microcosm, Inc., and Kluwer Academic Publishers. 1995.