

REPRINT

Title: THE COTS SOFTWARE ISSUE REVISITED/THE POWER OF POSITIVE COMPLAINING

Author: Strether Smith

Source: *Sound & Vibration*

Date: May 1997

A little over a year ago I discussed our early experiences with “Commercial-Off-The-Shelf” (COTS) software systems for data acquisition and analysis (S&V, April 1996). At that time, our group was doing the initial design on a new, large-scale, data acquisition system and, although we were committed to the COTS-software concept, we had not made a selection between the two final candidates. We had built several small software systems that were delivered to our in-house end users using each of the two prime candidates. They were relatively successful but the “products” of one of the COTS packages, National Instruments LabVIEW, proved to be consistently more robust, easier (and more fun) to program, faster, and easier to maintain. Thus, despite some reservations and knowledge of shortcomings, we chose it as our primary development system for all functions (test-definition data base, diagnostic systems, run/real-time operations, and post processing). In the ensuing eight months, a staff of four engineers has built a full-featured data acquisition and analysis system, reproducing and extending a system family that had taken ten man years to build with “conventional” (Fortran and C) programming tools.

At the completion of this task there is no doubt in our minds that the use of a “good” COTS processor results in superior end-user systems that are produced in record time. However, it is not a panacea!

The basic problem is that, when you decide to use a “COTS” processor, you put yourself at the mercy of the provider. If there are bugs and/or missing features, either work-arounds must be developed or the vendor must provide the required fixes or upgrades. Of course, the initial objective is to find a program that has a minimum of these land mines (one of the reason for our choice of LabVIEW) but all processors will have bugs and shortcomings, or will lack features that you think are critical. How do you get these holes/errors filled/fixes?

First, let’s look at the problem from the software vendor/developer’s side (I have been there too). The basic premise is that our vendors/developer is doing the best they can to produce a first-class product. They are soliciting feedback and implementing changes in order of priority primarily based on frequency of occurrence. If the problem is reported (and properly described) often

enough to get the attention of the development director it will be addressed. If complaints are not “persistently” presented, they will not be addressed.

For example, when we initially reported a long-standing LabVIEW logarithmic-format plotting problem, National reported that:

- It had only been reported a few times (strange, since we found that it was a widely-known problem).
- Nobody at National really understood the problem or its significance. No one had presented the problem to them in a way that demonstrated the anomaly properly. (To show them, I put together a small package showing how I thought that the log plotting should be handled. This small amount of effort produced instant interest).

The problem was that, although it was a well-known problem in the user community, the developers had not been adequately informed. Considerable complaining finally produced:

- Recognition that the problem is a genuine bug.
- A promise to fix it in a near-upcoming release (no commitment on time).

This is not a very satisfying result. The reason is that there have not been enough complainers. If you are unhappy with the logarithmic plotting (or anything else) in LabVIEW (or your favorite processor) call them. CAST YOUR VOTE!

We, as users, need to:

- Be more critical of the products: We have paid good money for them and made significant investments in learning how to use them. They should work and do what we want. Shortcomings must be reported.
- Be very descriptive of what the problems are and/or what we want. Send them examples. This takes effort on our part but it must be done. Other approaches are shortsighted. Show them that you care.
- Look at other products for comparison: If we did not know that some programs can print six graphs/minute, I would not be (as) dissatisfied with the one-plot/minute performance that some products have.
- Be creative: The vendors love new ideas that will make their product more attractive.

Smith, "The COTS Software Issue Revisited," *Sount & Vibration*, Page 2 of 2.

On the other hand, the vendors need to:

- Be sure that the "Help" lines are working and responsive. In many cases, the user is more knowledgeable about the problem than the front-line service staff. These instances must be elevated quickly. This is not a trivial problem, and the responsiveness is often clouded by differences in terminology. As a result, I don't know of any vendor that does this well.
- Differentiate between "Help/Technical Support" issues and bug-report/complaints. Technical support that requires work-arounds is a demonstration of a bug that must be fixed.
- Encourage their users to complain. Many vendors do this, in principle, but they should solicit and encourage problem reporting. How about a prize for the "bug and/or neat enhancement of the month"?
- They should institute a program of calls to the users by technical people to find out how things are going. What are you using the program for today? How well does it work? What would you like to see different?

The vendor should assume that, if there is no interaction (read complaints), that something is wrong. If this is wrong and all is fine, the vendor has made easy points by demonstrating that he or she cares.

The bottom line is: If we want good products (or products that fit our needs better) we must communicate with the developers and make it clear what we think is important. They are faced with a cacophony of requests from a variety of users that have a wide range of needs/desires. The squeaky wheel will be serviced first and if you want particular fixes, changes, or enhancements, you must let them know. The vendors are not clairvoyant: they need our help. Let's give it to them.