# I'm Sorry, Dave – I Can't Certify That

Come on, you know the reference I am making in the above title, right? Remember back in 1968 when the movie "2001: A Space Odyssey" came out? Based on the book by Arthur C. Clarke, this Stanley Kubrick movie was truly awesome! The HAL 9000 [Heuristically programmed ALgorithmic] computer was superb! (Did you ever notice that HAL is one letter below IBM in the alphabet?)

The HAL computer, voiced by the actor Douglas Rain, was particularly interesting. The computer, which developed paranoia, eventually killed four[1] crew members, and then tried to kill Dave Bowman, who was trapped outside of the ship in a smaller space pod without his space helmet. He asks the HAL computer to open the pod bay doors to let him back inside the ship, and HAL says "I'm sorry, Dave – I can't do that!" (As a side note, imagine the home life of Douglas Rain. His wife would ask him to do something simple, like "Honey, could you carry the trash out?" He would reply "I'm sorry …" and his wife would probably run screaming from the room.)

In a later novel (and movie), "2010: Odyssey Two," it was explained that HAL was given conflicting orders that drove the poor computer into a psychotic mania, forcing him to try to kill all of the crew. Imagine that – conflicting orders drove the computer crazy. Gee – wonder what that would do to a developer?

Back in the 1940s, John von Neumann originated the idea of what we consider modern computer architecture[2]. It included basic concepts such as data/instruction store, a central processing unit, input/output, etc. The nice thing about hardware was that visualization techniques (blueprints, circuit diagrams) allowed developers to see what they were building.

By the mid-60s, hardware technology had advanced so rapidly that software, not hardware, was becoming the limiting factor. Various methods were used to help developers visualize the software. One of the early tools – still used today – was a thing called a HIPO (Hierarchical Input Process Output) chart, which displayed a top-down visualization of the major components in a system.

I remember when I first learned how to use the HIPO process. In fact, I still have my original IBM-supplied green HIPO template. It seemed so easy: consider the inputs to your system, develop processes to manipulate the data, and produce output. How hard could that be? As any seasoned developer will tell you, it can be very hard!

When I was learning the HIPO process, I was given toy problems such as "Given three sides, determine the area of a triangle." Eventually, you would realize that a better problem statement was, "Given three sides, determine if they are indeed a triangle, and calculate the area of the triangle." The point eventually driven home was that you cannot ever trust the input.

Then it was time for another new law: "Garbage In – Garbage Out" (GIGO), that says that given garbage as an input, you should expect garbage as an output.

However, GIGO has recently taken on a new meaning. I have seen it referred to as "Garbage In – Gospel Out." This meaning implies that we poor humans have a tendency to implicitly trust the output of a computer. In other words, it doesn't matter how good the input was or how incorrect the process was, the output is considered gospel truth.

And that leads us back to the title of this column. How do you really know that your output is good? By making sure that the process is correct, and that the input data is valid. You do this by the process of verification, validation, and accreditation (VV&A). And part of the VV&A process consists of looking at the quality of the process.

Now, I am not saying that being Capability Maturity Model® Integration (CMMI®) Level X or ISO 9001 guarantees that you are producing top-quality software. I'm also not saying that having an assessment of your processes will make things better. What I am saying is that being CMMI Level X or ISO 9001 gives me a great deal more confidence that at least you care, and having an assessment lets me know what my weak spots are. Have you ever bought a used car? Did you have a good mechanic check the car out? Did he guarantee that the motor would not fall out after 10 miles, or that the transmission would not fail? Nope. But he did guarantee that basic things appeared OK, which gave you a bit more faith in the reliability of the vehicle. You were more willing to spend your money. And, if the mechanic said, "The car is OK – but you need new front shocks," you knew what you needed to fix in the short term.

Software development is much the same. Don't you feel a bit more confident spending your money knowing that the basics are all covered? Having a few experts tell you where your process is deficient, and letting you know what to fix first saves money and lowers anxiety.

As software and system developers, we are often given conflicting orders, just like HAL. "Cut costs – but keep quality up!" "Add these requirements, but don't increase the delivery schedule!" "Keep full functionality, but cut 10 percent from your development budget!" Perhaps, instead of carrying bottles of Valium and Prozac to my next budget and schedule meeting, I can use assessments and certifications to help me make the best of what I have, and improve what I have left to work with.

Just like HAL, I sometimes have a tendency to get a bit paranoid. While I have yet to reach the point HAL reached, I *have* sometimes considered locking a few managers and co-workers[3] outside the building to improve productivity.

**— David A. Cook, Ph.D.**
Senior Research Scientist
The AEgis Technologies Group, Inc.
dcook@aegistg.com

1. Well, actually only three. In a later book by Clarke (3001: The Final Odyssey), Dr. Frank Poole was found floating in space about a thousand years later, and successfully revived.
2. John von Neumann, et. al. "Preliminary Discussion of the Logical Design of an Electronic Computing Instrument." 1946 <www.cs.unc.edu/~adyilie/comp265/vonNeumann.html>.
3. For all of my current and former co-workers and managers, were you expecting to see your name here?