MEASURING MATURITY. The sophistication of measurement and experimentation is a key aspect of a discipline's maturity. When we first wanted to know about temperature, for example, it was enough to use our fingers to distinguish "hot enough" from "not hot enough." As we had more questions about temperature and its effects on other aspects of the world, we developed more sophisticated techniques and tools for quantifying temperature so we could refine our understanding. Similarly, software measurement and experimentation must become more sophisticated, building a body of knowledge that will let software engineering mature and grow. The difficulties of measurement and experimentation are no justification for relying solely on gut feeling (and perhaps taking counterproductive or even dangerous actions); rather, they are inspiration for forcing us to act like scientists and engineers, not witch doctors and astrologers.

Shari Lawrence Pfleeger, Systems/Software
Victor Basili, University of Maryland Institute for Advanced Computer Studies
Lionel Briand, Fraunhofer Institute for Experimental Software Engineering
Khaled El-Emam, Centre de Recherché Informatique de Montreal

Al Davis responds:
Thank you for taking the time to record your opposing viewpoint. I wrote the March 1996 editorial specifically to make readers more aware of their own actions. My goal was not to attack or defend empirical studies; nor was it to attack or defend rationalism. My intent was to make people more aware that both approaches are flawed—and flawed to the same extent.

BLIND ALLEYS. You write that "The software literature is full of examples where technology sounded good at the time but turned out to have many drawbacks." This is true. However, I believe that this is true in all areas in which humans endeavor. Life is full of blind alleys that seem initially promising. This is (and will continue to be) true whether the reason for making the decision is based on intuition or on experimental results. There are many examples of both.

You further comment on my assertion that "so many variables are uncontrollable that we should not even try to control them," and you say that this "pushes us backwards, giving us no scientific basis for our technology decisions." I agree with your final conclusion here; we do not have a scientific basis for our technology decisions. However, some experiments are so poorly controlled that we end up with what we think are scientific bases, when all we are doing is deceiving ourselves. The real problem is that there is no oracle available to validate our results. We have intuition (which may be flawed) and experiments (which may be flawed). If their results disagree, we do not know which one is valid.

I also agree that "a healthy skepticism about the quantitative nature of our analyses should lead us not to discard the analysis, but to build multiple models . . . ." One of those models should be our intuition.

RELEVANCE OF OPINION. You assert that "Continual investigation involves . . . experiments, case studies, surveys, observational studies, and . . . other investigative studies." And I would add "and people's opinions." Many experiments have shown that when subjects in an experiment believe that what they are doing is good, the results will be much more positive then when subjects do not believe what they are doing is good.

For example, we could perform an experiment to see if object orientation is good or not. We could assemble two equally productive and educated teams: one that already believes that anything that is OO must be good, and the other that already believes that OO is "just a fad." I would guess that there will be a high correlation between the beliefs and the resulting gains in productivity and quality. Strigini, in his January "Quality Time" column, sees such "self-fulfilling prophesies" as detriments to good experimentation. I see such preconceived notions as the very thing you want to measure. After all, if believing something will work is sufficient to achieve measurable results in experiments then, by golly, we have figured out the secret to productivity and quality improvement!

While it is true that "variable control applies also to other disciplines, including medicine," there are at least two differences between medical and software engineering experimentation. First, in many medical experiments—such as when validating a new pharmaceutical—the subjects do not know if they are in the control group or the experimental group; this is impossible (I think!) in software engineering. Second, there is a difference in the degree to which personal differences affect the outcome. You state your belief that "variability among people is at least as large in medical trials as in software engineering skill levels." I believe that personal differences are the most significant drivers of outcome in software engineering, not the inherent "goodness" of a method or tool. I do not believe the same to be true for medical trials, but that is pure conjecture on my part.

NO INCONSISTENCY. I do not see the results found by Belady/Lehman and Basili/Turner concerning software evolution to be inconsistent. Belady and Lehman studied data after a system evolved to a point of breakage. Their conclusion was that entropy increased. Their study said little about what could be expected if you tried to improve a program's maintainability during evolution. Basili and Turner concluded that maintainability could be improved if you tried to do so. There is no inconsistency here. There is no study of variables needed to see why they achieved "different" results.

Life is full of blind alleys that seem initially promising.

FOUR TENETS. Regarding my first tenet, clearly a more moderate position makes sense: You should use both experimental results and intuition. The medical examples you offered were indeed tragic. The software example you provided supports my revision; the results of the SEL experiment concerning Ada are intuitive. When experimental results differ from intuition, we don't know what to believe. When they agree, there is a general tendency to believe them.

You write that my second tenet is "misguided." I guess our difference of opinion stems from the amount of faith we have in the results of an experiment in which the most significant variables cannot be controlled. (I'm not referring here to the particular experiment you cite; I am not privy to its details.) If you believe in such results (or believe that the uncontrolled variables are less significant than I) then your point is valid. If not, then the point is invalid.

In many ways, this is like a religious argument: If you believe then you believe, and if you don't, you don't. Again, we lack an oracle. If an experiment shows that formal methods alone are ineffective, but formal methods plus unit testing are effective, how do we know the results are meaningful? You are essentially using the results of an experiment to refute my second tenet, but it is equally valid to use the tenet to refute the experiment.

You write that tenet three "suggests that only researchers do research." I apparently expressed my opinion poorly. I strongly believe in partnerships between researchers and practitioners. In fact, software engineering research without practitioner involvement is almost ludicrous (see my May 1996 "From the Editor" for a related discussion).

My advice to a researcher would be to find a practitioner to validate results by "practicing" with the practitioner. In this case, the practitioner's environment becomes a Petri dish. My third tenet was meant to imply that serving as a Petri dish is not the purpose of software development. My advice to the average practitioner is to find a researcher who already has the results of an experiment. It is unnecessary for every company to serve as a Petri dish for every new technology.

On tenet 4, you state that "expert judgment that is not grounded in careful investigation is highly suspect." I agree with you that Strigini's "Quality Time" column was one of the best discourses ever on this subject. Strigini says "that bolstering our subjective judgments with scientific analysis can increase their reliability, but as an aid in decision making the scientific method also has limits." This supports my position that intuition is at least as important to making intelligent technology-adoption decisions as experimental results.

IN AGREEMENT. I agree in spirit with much of what you presented in your letter. I believe that empirical studies in software engineering phenomena are very important. I believe that such research should and must continue. Any quantitative results we can generate will serve as a useful foundation for this generation and those to come. You represent some of the most talented researchers in software engineering today; if anybody can produce such a foundation, you can. However, as I mentioned in my original editorial, our inability to control extremely significant independent variables sheds some level of doubt on experimental results.

Industrial decision makers need to understand that both intuition and experimental results in software engineering are inherently imperfect. ◆

"Eras" and the "positive effect"

I THOUGHT AL DAVIS' "ERAS OF Software Technology Transfer" ("From the Editor," March 1996) was interesting and important. I'm a software practitioner in the trenches (whenever I stick my head up to see a little bit further, someone shoots at me) and reading Davis' column inspired a few random thoughts on "obvious positive effect."

The Mac fanatics call obvious positive effect "insanely great" and give it cult-like status. The Next fanatics have their own more modest code-phrase: "It just works." In any event, immediate apparent value is a big deal—not just because it helps people decide what tools to employ, but because it forces toolmakers to organize their products into discrete pieces, each with apparent value.

Intuition is at least as important as experimental results.

I read several "academic" professional magazines (*IEEE Software*, *Computer*, and *Communications of the ACM*) and in recent years they have only slightly improved the way they sell their value. For example, several magazines now offer article summaries, but the actual articles are too often dry and inaccessible.

Most of us have been to conferences where good speakers bring their dry, academic papers to life. Well, I want to read articles like those good lectures. I'd like the prose to be less academic and to see more examples and diagrams. I realize that this doesn't look as good in PhD circles—but, hell, I have only a master's in computer science. I want your articles to have "obvious positive effect"—the same topics, but presented more clearly.

I think Steve McConnell is the best thing to happen to software practitioners in the last few years. Not that you should have a magazine full of articles like his—you can only say so much about the details of writing code. But until recently, nobody said anything about it. He brings "obvious positive effect" to your magazine.

I wish Al Davis good luck at being a rationalist. Or rather, good luck moving the industry in that direction. People like being set in their ways, and "their ways" in our industry all too often means outmoded practices and tools, as well as the kind of auto-pilot management that provides endless fodder for the daily "Dilbert" comic strip. ◆

Mike Morton
mike@morton.com