(Response to reader letter in IEEE Software. Not acceptable to the publication as it is 62 words too long.)

Reach for standards — good ones

Bertrand Meyer, ETH Zurich and Eiffel Software

I am flattered by the attention, and the reproach that I didn't say all about testing — great compliment in its implicit suggestion that an 1800-word column could have contemplated such an aim. Its title was "Seven principles of software testing", not "*The* seven principles". Mr. Everett is right to advise readers to read seminal references.

I regretfully disagree with his three main points: that my definition of testing is too narrow; that it is too negative; and that the ISTQB document provides the answers.

On the first: Mr. Everett and ISTQB broaden the definition of testing to cover essentially all of quality assurance. In science one is free to use any term to denote anything as long as the definition is precise, but it makes no sense to contradict established practice. The ISTQB definition goes beyond the dynamic techniques commonly known as testing to encompass static ones. This is inappropriate; hundreds of publications discuss "proofs versus tests" or "static analysis versus tests". Such relationships are of great relevance (including to me as the originator, with Yuri Gurevich, of the Tests And Proofs conferences, http://tap.ethz.ch), but the differences remain clear. Ask people in industry or research about testing; most will describe dynamic techniques. If ISTQB wants to extend its scope to quality assurance it should change its name, not try to redefine decades-old terminology.

The second criticism targets my thesis that the most effective form of testing is directed at finding faults. Mr. Everett calls this view "confrontational". I did not address psychology. How much confrontation effective testing implies is arguable. I am writing this in a plane; whether the flight software's tester hurt the programmer's feelings concerns me less, right now, than whether he exercised best efforts to uncover deficiencies. (Actually, the informational display has been stuck for the past hour at 958 km from our starting point — I hope critical systems were tested more confrontationally.) Psychology is not the key consideration. Some of my best friends are testers, and I even knew one who, most years, called her mother on her birthday. For a more relevant view, think game theory. We have a two-player game: the programmer strives to minimize bugs produced; the tester, to maximize bugs found. Each is grateful for the other's work: the programmer likes that someone strenuously searches for imperfections (it beats having customers find them); the tester likes that the programmer designs for testability. They might even have coffee together — decaf, to dampen confrontational impulses.

The third reproach is that I should have relied on the ISTQB document, which deserves the term — "worthwhile" — that Mr. Everett kindly applies to my column. It provides some excellent advice from people with extensive testing experience, but is not the kind of industry standard that Mr. Everett suggests. It falls short of the quality of IEEE, ISO or ECMA standards and often reads like a manifesto, a different genre. Section 1.2, "What is testing?", starts: "*A common perception of testing is that it only consists of running tests, i.e. executing the software. This is part of testing, but not all of the testing activities.*" The section's remainder is in the same style, discursive and defensive, but contains no definition of testing; a professional standard would include a carefully worded definition reflecting expert consensus. Here we read "*Debugging and testing are different*", and a few lines down, "*The responsibility for each activity is very different, i.e. testers test and developers debug*", a textbook example of how not to write reference works: be repetitive; weaken the focus through such words as "very"; waste the reader's attention on platitudes (if miners mine and programmers program, what should we expect testers and debuggers to do?). Problems or form and substance permeate the document. To cite just one more, section 1.3 describes testing principles. Some are, as Mr. Everett would say, "worthwhile", but take Principle 4: "*A small number of modules contain most of the defects discovered during pre-release testing, or are responsible for the most operational failures.*" This is an empirical observation, not a principle. A principle should guide action, as indeed Principle 3 (start testing early). "Many people forget to call their mothers" is not a principle, although one might work out a principle from it, such as "Don't forget to call your mom on her birthday". Similarly, a true principle may lurk behind "Principle 4", perhaps "Apply to each module a testing effort commensurate with its expected fault density". The ISTQB document is a collection of gems of testing wisdom, ready neither as a standard (which it does not claim to be) nor as a syllabus (which it does).

We do need good testing syllabi and standards. They might include my principles, or some of them, or none at all; whatever the case I hope my column will have brought its contribution, however minute, and thank Mr. Everett for continuing the discussion.