Marie-Claude Gaudel,
Honorary Professor,
Laboratoire de Recherche en Informatique,
Université de Paris-Sud & CNRS

Software Testing and Formal Specifications: a turbulent history

About me…

- I am a computer scientist.
  - During years, I have developed my own research.
  - I have been a member of various management committees of research, in France and abroad.
- I am retired. I have time to think
  - about my past research activity;
  - about what I observed.
- I am here as a witness. This presentation is just a personal account with some attempts of analysis.

This talk is about this excerpt of the CFP

… undone science could also refer to the consequences of "theoretical commitments", e.g. dominant paradigms, when they blind us collectively about what is worthy or not of exploration—all the while accounts of paradigm shifts in our young domain remain rare.

In the seventies, it was a bizarre and dubious idea in computer science to link the two sub-areas below :

- Software testing.
- Formal specifications of software, proofs of program correctness.

Namely:

To use a formal specification of a software system to test it

The topic in two nutshells

(Apologies to the computer scientists in the audience)

- A specification: a non algorithmic description of what the program should do.
- A formal specification: a syntax, some semantics, possibility of inference.
- Example:

```
(max3(x, y, z) ≥ x & max3(x, y, z) ≥ y &
max3(x, y, z) ≥ z ) & (x = max3(x, y, z) or
y = max3(x, y, z) or z = max3(x, y, z) )
```

- Only "orthodox" use of formal specifications in the seventies : proof that a program satisfies a specification.

Where the program is considered as a logical formula.

The second nutshell

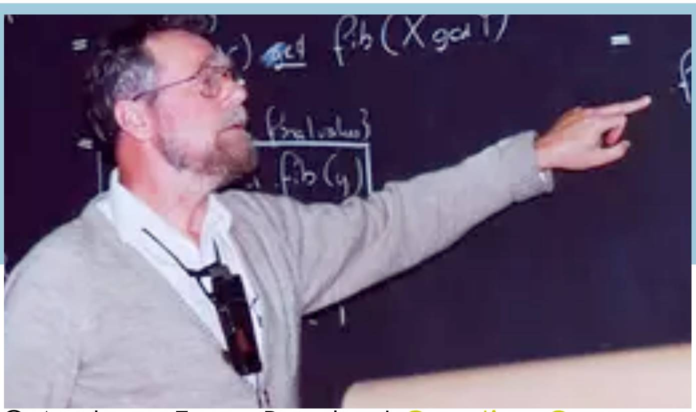
(Apologies to the computer scientists in the audience)

- Software testing: execution of a system with some inputs, and decision whether its behaviour and the outputs are correct.
- Structural software testing: the testing strategy is based on the program. Test inputs are selected to cover some constructs in it.
- Black-box software testing: the testing strategy is based on the specification. Test inputs are selected to check properties stated in it.

In both methods, the specification is essential for the decision.

Both methods are necessary (+ some other ones…)

- In the seventies, the atmosphere was extremely tense between the advocates of program proofs and those of software testing.
- Some influential scientists were exchanging mutual anathemas.

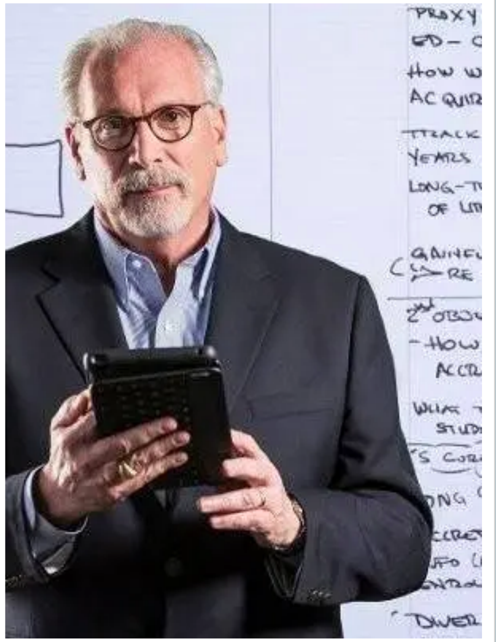


Edsger W. Dijkstra, ACM Turing Award in 1972

" A common approach to get a program correct is by subjecting the program to numerous test cases. From the failures around us we can derive ample evidence that *this approach is inadequate*. "
In EWD303, ± 1970

" Program testing can be a very effective way to show the presence of bugs, but is hopelessly inadequate for showing their absence.
The only effective way to raise the confidence level of a program significantly is to give a convincing proof of its correctness."
In "The Humble Programmer", CACM 1972

“Furthermore the absence of continuity, the inevitability of change, and the complexity of specification of significantly many real programs make _the formal verification process difficult to justify and manage_. It is felt that ease _of formal verification should not dominate_ program language design.”

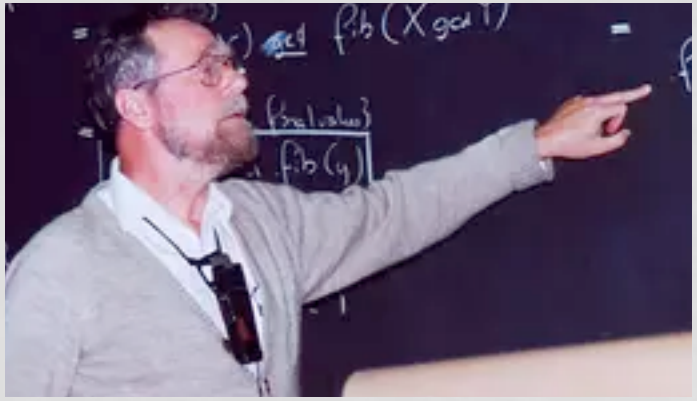


https://www.linkedin.com/in/richard-demillo/

Richard A. DeMillo, Richard J. Lipton, and Alan J. Perlis

Social processes and proofs of theorems and programs.
Communications of the ACM 22, 5 (1979), 271–280.

DeMillo is a professor at Georgia Tech University. He is not a Turing Award, but…
among many other things, he is, with Lipton, the inventor of mutation testing of software. He was also at the origin of a predecessor of NSFNet, etc, etc. He had been the Director of the Software Test and Evaluation Project for the DoD and Chief Technology Officer of Hewlett-Packard, etc, etc



"This note concerns a very ugly paper [...].
Its authors seem to claim that trying to prove the correctness of programs is a futile effort and, therefore, a bad idea."

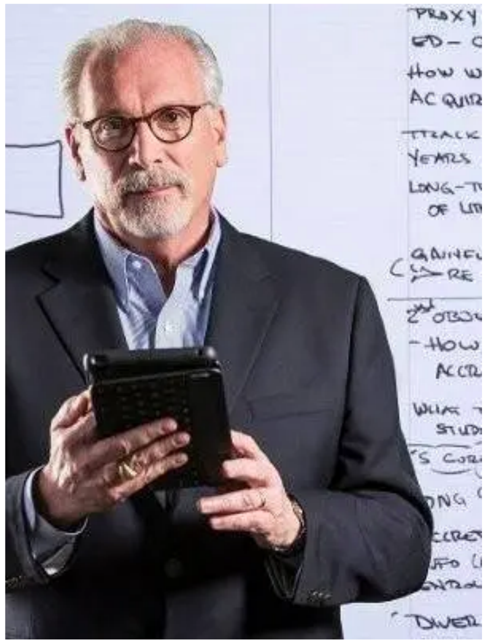
"Deprived of what is generally considered computing's core challenge, American Computing Science is the big loser, and we can not blame the universities, for when the industry most in need of their scientific assistance is unable to face that they are in a high-technology business, even the best university is powerless."

Edsger W. Dijkstra. A political pamphlet from the Middle Ages, EWD638, 1979.

It is nothing but symbol chauvinism that makes computer scientists think that our structures are so much more important than material structures that
(a) they should be perfect, and
(b) the energy necessary to make them perfect should be expended.
We argue rather that
(a) they cannot be perfect, and
(b) energy should not be wasted in the futile attempt to make them perfect.

It is no accident that the probabilistic view of mathematical truth is closely allied to the engineering notion of reliability.
Perhaps we should make a sharp distinction between program reliability and program perfection —and concentrate our efforts on reliability.

See also : Response from R. A. DeMillo, R. J. Lipton, A. J. Perlis, April 1978, https://www.researchgate.net/publication/255678209



SOME OBSERVATIONS

- The dispute was between formal proof of programs and software testing (1).
- With some notable exceptions, the two parties can be characterised by:
  - Most advocates of formal proofs were in *Europe*. They worked on the semantics of programming languages. They published in journals and conferences that often mention "*theory*" or "*science*" in their titles
  - Most advocates of software testing were *in Northern America*. They worked on *software engineering*. Journals and conferences are titled accordingly.
- But it is too definite to see it as theoreticians/engineers and US/Europe since:
  - there were excellent researchers in software engineering in Europe, and excellent theoreticians in computer science in Northern America.

(1) *Formal testing what just not imaginable*

Formal Testing, the beginning

- As often, several scientists got the same idea almost at the same time (I may miss some of them).
  - Bougé, L.: *Modeling the notion of program testing; application to test set generation*. Thesis, Université Pierre et Marie Curie (Oct 1982)
  - Gannon, J.D., McMullin, P.R., Hamlet, R.G.: *Data-abstraction implementation, specification, and testing*. ACM Trans. Program. Lang. Syst. **3**(3), 211–223 (1981)
  - Jalote, P.: *Specification and testing of abstract data types*. In: IEEE International Computer Software and Applications Conference COMPSAC. pp. 508–511 (1983)
- John Gannon (Univ of Maryland) and Pankaj Jalote (Indian Institute of Technology Kanpur) let down this risky research topic after a few years.

- My group pursued it…

Jumping several years later

Bernot, G., Gaudel, M.C., Marre, B.:
Software testing based on formal specifications: a theory and a tool.
Software Engineering Journal **6**(6), 387–405 (1991)

Gaudel, M.C.: *Testing can be formal, too*. keynote lecture
In: TAPSOFT 95, International Joint Conference, Theory And Practice of Software Development. LNCS vol. 915, pp. 82–96. Springer Verlag (1995)

Hierons, R.M., Bogdanov, K., Bowen, J.P., Cleaveland, R., Derrick, J., Dick, J., Gheorghe, M., Harman, M., Kapoor, K., Krause, P.J., Lüttgen, G., Simons, A.J.H., Vilkomir, S.A.,Woodward, M.R., Zedan, H.:
Using formal specifications to support testing.
ACM Computing Surveys 41(2), 9:1–9:76 (2009)

But it is not a full happy end.
There are still some pockets of resistance in both camps

WHY, AND HOW…

- I had been lucky
  - **I am afraid it's the main factor…**
- I was not a beginner. I was what the ERC calls a "consolidated" researcher. I was honourably known various circles: academy and industrial research.
- I got a permanent position in 1983, *then I was not afraid to pursue this risky project.*
- But what do you need to lead a research project?
  - Funds
  - Ph.D students and young researchers to build a team (I had been extremely lucky on this point!)
  - Places to present, discuss and publish
- And for all that you need support from some "big names".

Neighbouring Scientific Communities

- Surprisingly enough, active supports came from outside, namely from big names of some close scientific sub-areas.
- Telecommunication protocols
  - must be agreed upon by various entities. Their specifications are used as bases for conformance testing and certification. Designers were already using formal specifications for that.
- Chips designers
  - have been developing for quite a while test methods and tools based on logical descriptions of such circuits. Some of them liked the idea of similar approaches to hardware and software.
- Researchers in software reliability
  - were very keen on rigorous and formal approaches of system testing.

Moreover, a Win-Win Dialog between SE and TCS was established

- Software Engineering:
  - designing efficient test sets is time consuming and difficult. The fact that, thanks to some formal definitions, some tools can be developed to generate them was attractive.
- Theoretical Computer Science:
  - the academic community was pleased with this argument for the usefulness of their work.

- Passing these messages required much eloquence and conviction…

What was Hard to Stand?

- Good students leaving the boat after their Ph. D. for safer scientific paths.
- Big names sarcasms, or even gentle sermons, because you lose your time and energy.
- Difficult acceptance in conferences and journals.
- Funding agencies? Very difficult,
  - but feasible (with, in a few cases, a touch of insincerity).

- Later on, when sitting in various scientific committees, I have found extremely hard *to deal with original, risky projects in the right way*. Thus I don't hold against those people.

- I push for a new risky (utopic?) project: to state a way to avoid good scientific ideas to be turned into "undone science".

  It's the reason of my presence to-day.