Visuals for BP's Venture Research Conference

- When we had no computers, we had no programming problem either.
- When we had a few computers, we had a mild programming problem.
- Confronted with machines a million times as powerful, we are faced with a gigantic programming problem.
  - The programming problem is gigantic because
  (i) viewed as string of operations on a collection of variables, each individual computation is a gigantic object;
  (ii) the individual computation depending on the input, a single program has to be able to control a gigantic number of different possible computations.
- As a result, programming has become one of our most demanding intellectual activities, requiring great clarity of expression and the utmost economy of reasoning.
- This conclusion has never been refuted; it is, however, regularly denied because of its uncomfortable implications.
- It is vigorously denied by the computer industry, which would sell you its products rather as solutions to your old problems than as the source of terrifying new ones.
- It is vigorously denied by those customers that otherwise would have to admit that their computer manufacturer has fooled them.
  - [Remember that computers are preferably sold by dealing with such a high level of the customer's hierarchy that incompetence in computing is assured and objections from the technically competent can be overruled.]
- It is vigorously denied by those in computing that by its implications would be demoted to the rank of amateur.
  - [Remember: the surest way of making software design prohibitively expensive is viewing it as a production job to be done by cheap labour.]
- It is denied in those companies whose top management consists of lawyers and accountants, as their management lore has all its roots predating the advent of the high-technology industry.
- It is also denied by the personal-computer enthusiast that fails to distinguish between a barber and a team of surgeons.
- Back to the irrefuted conclusion that programming is very difficult; its acceptance gave rise to Programming Methodology as a topic of explicit scientific concern.
- In 15 years of Programming Methodology we have seen the combination of spectacular progress and sharp limitations.
Many formerly notoriously opaque algorithms now have "ingeniously simple and effective" explanations, which are jewels of clarity.

Sophisticated new algorithms are being derived which, 15 years ago, would have been absolutely impossible to conceive. But......

Current mathematical style --which grew in response to other challenges-- limits the applicability to relatively small programs.

The circumstances shaping the challenge, we try to redefine Mathematics from "The abstract science of space, number, and quantity" (COD) to "The (art and) science of effective reasoning".

We have learnt that calculi of all sorts have a major role to play.

We have learnt that it pays to design, for each calculus to be used, with great care a notation geared to one's manipulative needs.

We have learnt that most philosophers (those of mathematics included) are eminently ignorable.

We have learnt that a formalism's major purpose can be to free us from the shackles of our native language and the reasoning patterns induced thereby.

[Being "counter-intuitive" should not be held against any formalism that enables us to accomplish what is beyond the unaided mind.]

We have learnt that programming methodology and mathematical methodology in general are not so far apart at all. (For instance, a conscious separation of concerns is equally valuable for both.)

We have learnt that a purely syntactic analysis of the formal requirement can give heuristic guidance to the point of generating the best possible solution.

We have learnt that the potential for improvement can be dramatic, e.g. reducing a formal proof of two dozen steps to an equally formal proof of one step.

(Example.)

We have learnt that it pays to be ruthlessly pragmatic, but......

We have also learnt that it is still very hard to sell to industry the economic value of mathematical elegance.

Austin, 5 June 1986

prof.dr.Edsger W.Dijkstra
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712-1188
USA