Copyright Notice

Tripreport E.W.Dijkstra, St.Pierre-de-Chartreuse, 12-19 Dec.1976.

It was a meeting of IFIP Working Group 2.3 on "Programming Methodology", hosted by Gerard Veillon of the University of Grenoble in Hotel Beay Site in St. Pierre-de-Charteruse, a place for its inaccessibility once selected for a monastery. Coen Bron --the other Dutch participant-- wanted to go by car --after the meeting he remained there for a skiing weekend-- and picked me up on Saturday morning. Driving alternatingly, we arrived in Beaune on Saturday evening, having had one major stop. Along European highways, and particularly along the French ones, there is now a chain of highway restaurants under the name of Jacques Borel and in one of these we had lunch. It was the kind of mistake one makes in one's life only once.

In the centre of Beaune we found a nice hotel with (for once) a perfect kitchen. After dinner we made a little evening walk through the sleepy town and were quite surprised --and pleased!-- to encounter a big statue of and dedicated to Gaspard Monge! The next morning we continued our travel and it was about noon when we reached our destination. It was a most pleasant trip, but for the fact that Coen felt that it was quite safe for him to read the maps while driving and insisted on showing that he could do so without causing an accident. (When I refused to show signs of discomfort, he first allowed the car to shift to one side of the lane; eventually he asked me "to keep on his behalf my eyes on the road". I then told him what he wanted to hear, viz. that I did not like it.)

My journey back was less successful. With Ross, McKeeman and Horning I went (by a French train" from Grenoble to Geneva, where the other three had hotel reservation as they would fly from Geneva the next day. I had to catch a connection to Basel where I would pick up the Italy-Holland Express, for which I had a reservation in the sleeper. I intended to be home early Saturday morning. According to the schedule I would have 54 minutes in Geneva, but the French train accumulated a delay of more than one hour and I missed my connection. I had dinner with Doug Ross (who was very hungry), found a room in his hotel. and slept until 4 o'clock in the morning, packed and took a train leaving Geneva at 4.40. With changes in Bern, Basel, Mannheim and Köln I came home in the middle of the afternoon (still cursing the French railway system).

* * *

Particularly the first half of the meeting was not successful. It was a coincidence of circumstances. Mike Woodger had been W.G.2.3's extremely successful chairman, but he had given his chair to Jim Horning, who had to get used to the role; besides that, Jim was hit by "Napoleon's revenge". This, however, was probably the minor cause. The more important cause, I think, was that we had an exceptionally great number of "observers" and that --we had had "speaking observers" in the past-- many of them were eager to present their thing. This got somewhat out of hand.

In W.G.2.3 a member used only to "instruct" the other members if he had something new to tell of which he felt that it was --or could be-- very relevant. More often, the speaking member would seek the others' advice or opinion. The many speaking observers either did not know that rule or felt insufficiently secure to expose their uncertainty. The result was that the meeting was dangerously beginning to look like an ordinary conference with unrefereed papers. The third cause --but this I only realized after the meeting had been closed-- has been that, a month prior to the meeting, Zahm had sent the so-called "specification" of Peter Henderson, as it occurs in his article "An exercise in structured programming" (or something like that) as a challenge to the participants. Too many

people had picked up that gauntlet (instead of ignoring it), and, in view of their preparation, felt entitled to present their experience. With the exception of McKeeman's, all presentations inspired by Henderson's specification were terrible. (This was to be expected, for a more appropriate title for Henderson's paper would have been "A demonstration of the mess generated by indiscriminate use of sloppy English.") In short: we had a very false start.

On Wednesday morning the observers left the room so as to allow the members to attend to "Working Group matters". It was only then that we realized that, up till that moment, the meeting had largely been wasted, and that all of us were totally miserable about it. We wondered what had happened! Had we run out of steam? Was Programming Methodology completed or exhausted? Should we disband? As Wednesday afternoon was the official afternoon off, the members unanimously decided to cancel all other arrangements and appointments they had made for that afternoon, and to reconvene after lunch to have a meeting with just members, in order to take the experiment whether, "Among ourselves" so to speak, the spirit could be recaptured. It could and after working from two till after six, most of us felt that disbanding --what had been discussed so seriously that morning-- would be premature. There is still enough to be done!

The next two days were rescheduled and, thank goodness, much better (although not sufficiently so to compensate completely for the "lost" Monday and Tuesday; but that would have been too much to ask for).

* * *

I shall not review the week's program in any detail. I shall try, instead, to sort out my feeling, impressions and hopes concerning Programming Methodology in general and W.G.2.3 in particular.

We all know that an ideal program has more virtues than planets will ever be discovered in the universe. To mention but a few: it is correct, efficient, robust, portable, expandable, easy to modify, easy to maintain, easy to read, easy to understand, easy to write, etc.! We also know that Programming Methodology has been successful insofar it has been able to separate those concerns and to deal with them in turn. We now know, for instance, quite clearly that the unfactored criterion "A program is good (enough) as long as it satisfies your customers." is too woolly to be of <u>any</u> help. We now know, for instance, quite clearly that "correctness" is only meaningful with respect to precisely stated functional specifications, which act as a kind of logical firewall between the correctness aspect pf the design and its usefulness aspect. (Which mathematician worries about the correctness of a proof for a vague "theorem"?) We also know that the successful isolation of a nontrivial aspect is always a significant scientific contribution (e.g. the discretization of synchronization requirements, BNF to describe the context-free aspect of programming languages, the postulational semantics that abstract from computational histories, etc.). It is from such discoveries --i.e. the isolation of nontrivial aspects and successfully treating them in isolation-- that Programming Methodology can profit, probably even can profit more than from anything else.

Such a separation is traditionally opposed to by the people for whom (for lack of a better term in my vocabulary) I have coined the term "integralists". We always had a few integralists in W.G.2.3 and they always caused the problems that are to be expected, but I used to consider them as a useful antidote, and quite healthy when taken in small doses. This time we had too many integralists. Such rigorous separation of concerns is nowadays (politically!) impopular. The current misgivings about the influence of science in general and of technology in particular are in no way better expressed than by the cry for "interdisciplinary

approach", "systems thinking" etc., and he who concerns himself for some time in depth with only one aspect can be sure of getting accused of narrow-mindedness. Yet, the impopular separation of concerns is more necessary than ever, and W.G.2.3 (not obliged to produce a Magnum Opus) has in this respect not only special opportunities, but by this very fact also special obligations. In its last meeting this has too often been forgotten; this was largely, although not entirely, due to a number of observers who had (mostly unconsciously, I guess) accepted political prejudices of their respective environments as scientific constraints. We should not allow this to happen again. (Large conferences are becoming uniformly boring, nearly everybody reporting how he has tackled the same "wrong" problems with the same inadequate techniques. I am beginning to feel that this uniformity of the behaviour of the scientific world in our field is largely caused by the homogeneity of political objectives, prejudices and pressures in the Western world. And often they seem pressures to abstain from trying really effective solutions because they are at the time and place politically impalatable. To interpret the boring uniformity of these large conferences as a symptom of "completion" of the field would be a serious mistake.)

With Programming Methodology in our charter, the effectivity of patterns of reasoning has always been a serious concern. It is, for instance, in the name of that effectivity that systems of postulational semantics have been developed so that we may come to grips with the semantics of a program without being forced to do so via the detour of the class of possible computational histories. Niklaus Wirth made no joke when he wrote that programming languages should be defined without any reference to computers or compilers. (I would like to phrase it still stronger: "independent of any underlying computational model".) Among ourselves most of us really try to stick to that rule (and when, for instance, Tony could not, he apologized for the absence of proof rules!) Now, regretfully, we had a great number of speakers who were unaware of the desirability to abstract from the computational histories, could hardly grasp what was meant by it and "talked operationally" with all its traditional clumsiness as if we still lived in 1965. We should not allow that to happen again.

We have forgotten that "thinking" as a topic of explicit concern is a very sensitive subject (because we all think and hope to do it well). This very intimate activity of thinking is closely intertwined with our public activities of writing and speaking, and, therefore, how we write and speak should be of equally serious and explicit concern. But this time --and we should not allow that to happen again-- the way in which people expressed themselves could not be discussed openly, and we had to subject ourselves for several hours to the most barbarious slipshod haberdashery. (I tried once, asking the speaker for clarification after a few nonsensical sentneces. His comment "Are you commenting on my language or on what I am saying?". I shrugged my shoulders and left it at that, for it would have been too painful to explain in public that he made an empty distinction and that he spoke words but said --and probably thought-- noting worthwhile. And that was terrible: in W.G.2.3 we are not used to avoiding discussions that might become painful.)

Some people's mixed attitude towards thinking also surfaced during one of the discussion (but it was not pursued....). We were shown experiments in "program transformations" that, while retaining semantic equivalence, may influence efficiency drastically: a fully legitimate and sometimes even fascinating topic. It <u>may</u> provide a way of separating in time our concerns about correctness and efficiency: one first writes a correct one and then transforms it into one that is efficient as well. To advocate such an approach now, however, seems premature to me. A few examples given were most unconvincing, because the derivation of the "inefficient but correct program" that could serve as a starting point for the transformation

process had taken their designers orders of magnitude more time than has been needed to solve the problem directly: one or several days versus 15 minutes. (A possible explanation could be that, when efficiency is ignored, one receives less heuristic guidance and the "solution space" becomes too great.) When I draw attention to this discrepancy, one of the participants --a full professor at a (once?) famous university!-- more or less disqualified that 15-minute solution by remarking that its design had required competent thinking and, therefore "did not count" because nowadays you could not expect your students to try to learn to do so. He seemed to feel that in the future his system could provide a welcome Ersatz. I draw another conclusion: it confirmed my opinion that there is no substitute for a good brain, and that we would commit the cultural blunder of the decade if, seduced by the promises of Artificial Intelligence, we were to forsake our educational obligations towards the next generation. (I am afraid that the blunder is already being committed on a large scale.)

I think that I can understand the world better if I don't regard Artificial Intelligence and General Systems Thinking as scientific activities, but as political or quasi-religious movements (complete with promise of salvation). Back home I was chagrined to learn that the NSF has a "Program Director Intelligent Systems".

An afterthought. What in modern American --my 1973 Webster doesn't mention it yet-- is called "deskilling a job", boils down to changing a task in such a way that it can be done by less educated --that means: cheaper-- labour. It is mostly inspired by economic considerations; whether it is worth the cultural price to be paid for is another matter.

Some of the people at this meeting seemed engaged in, or to justify their efforts in terms of "deskilling the programmer's job". Quite apart from its desirability, which I don't feel tempted to discuss here, we should consider its feasibility. If we share the dreams of the Artifical Intelligentsia, the feasibility is no longer a point of discussion: given greater machines, more time and more funding, the whole programming problem will just disappear. For two reasons I happen not to share that dream: it seems technically as inattainable as automatic theorem proving, and someone will have to take the responsibility to believe (and to act accordingly) that the design is, indeed, the useful engine it was intended to be, and neither confidence or responsibility are things that can be delegated. (In addition to this I have been exposed to Pat Goldberg!) What can be done, is done --and I think: should be done-- is to try to mechanize the tedium. Mechanizing the tedium, however, increases the density of difficulty of the task that remains! I don't object to it, for it increases mankind's programming ability, but we should be aware of the fact that it is the contrary of "deskilling the programmer's job". (It creates already serious ocial problems for the thousands and thousands of old practitioners!)

And finally: I can sometimes not escape the impression that the mechanizers of the tedium are overselling their techniques and overstating their case by the (sometimes even mechanic) generation of quite avoidable tedium. (End of afterthought.)

Plataanstraat 5
NL-4565 NUENEN
The Netherlands

prof.dr.Edsger W.Dijkstra
Burroughs Research Fellow