27th February 1975

"Craftsman or Scientist?"

(Luncheon Speech to be held at "ACM Pacific 75" at San Francisco, Friday 18th April 1975, by Edsger W.Dijkstra, Burroughs Research Fellow.)

My somewhat elliptic title refers, of course, to the programmer; so much you may have guessed. What, in all probability, you could not have guessed, is that I have chosen to use the words "craftsman" and "scientist" in a very specific meaning: they have been chosen to characterize the results of two extreme techniques of education, and this luncheon speech will be devoted to a (be it short) discussion of their role in the education of programmers, in the teaching of programming. For the transmission of knowledge and skills both techniques have been used side by side since many centuries.

The future craftsman joins a master for seven meagre years, he works as an apprentice under his guidance and supervision, absorbing gradually, by osmosis so to speak, the skills of the craft, until he may be called a master himself. Craftsman typically form Guilds and the guild members tend to keep their common craft as a well-guarded secret among themselves: not blowing the gaff is one of their rules of professional conduct. Note, finally, that old crafts have been lost, dependent as their survival was on the continuing transmission from one generation to the next.

The future scientist learns his trade as a student from a teacher, who, in contrast to the master who transfers his knowledge implicitly to his apprentice, tries to formulate the knowledge and to describe the skills as explicitly as possible, thereby bringing both into the public domain. The latter technique is the prevailing one at the Universities. It is no coincidence that the rise of the Universities occurred when the printing press became widely established, and it is no accident that each University regarded its Library as its greatest treasure: the library was the embodiment of its specific calling. The scientists regard the free interchange of knowledge and insights as essential, and, in consequence, being non-secretive is one of their rules of professional conduct.

To this very day, both techniques are applied side by side: physicists, for instance, are mostly scientific, physicians, however, are mostly much more like guild members. Mathematicians are somewhere in between: mathematical results are published and taught quite openly, but there is very little explicit teaching on how to do mathematics, and publishing besides the results also the heuristics that led to them is regarded by many as "unscientific" and therefore, bad style: quite often the editor's censorship will try to prohibit their publication.

I have sketched for you two extreme educational techniques, but this was only preparation: my real topic is "Where along this scale should we place the teaching of programming?". This, as I have learned by sad experience, is a risky subject to discuss, because one always discusses it with people who themselves are involved in one way or another in the programming profession, and their personal involvement tends to evoke strong emotional reactions. Let us try to understand them, for only then we may be able to cope with them.

To make implicit knowledge explicit and to discuss how to describe skills, so that they can be transferred, implies, if not the birth, at least the conception of a new science. But we should realize that changing a craft into a science, and making public property of the secret knowledge of the guild will always cause the guild members to feel threatened. For many a "puzzle-minded" virtuoso coder of the early sixties, the scientific development of the last decade has been most unwelcome. He feels like the mediaeval painter that could create a masterpiece whenever his experience enabled him to render proportions well, who suddenly found himself overtaken by all sorts of youngsters, pupils of Albrecht Dürer and the like, who had been taught the mathematical constructions that were guaranteed to surpass his most successful, but intuitive renderings. And with nostalgia he looks back to the good old days when his experience and feeling made him an outstanding crafstman. And we should realize that, as far as programming is concerned, the battle is still going on. From a European country, the name of which I shall not divulge in order to avoid personal complications, I recently studied a proposal for the organization of its computing science teaching at University level. The majority of its authors --all of them professors of computing science in their country-- should be characterized as "craftsmen". As a result, their proposal had a pronounced anti-intellectualistic flavour: it stressed that the students should be taught how to solve the problems of "the real world" and that, therefore, the curriculum should pay as little attention as possible to "abstract subjects". Such utterances are unmistakable and, undoubtedly, you recognize them. So much for the pure craftsman's point of view.

At the other end we have the pure scientist: if we give him the power of decision, the result will be equally disastrous. He will see his discipline --be it automata theory, recursive function theory, formal language theory, logic or queueing theory, you name it-- with the exceptional clarity that we are entitled to expect from the modern scientist, but one thing is for him nearly impossible to accept, viz. that his beautiful and formal apparatus, indispensable as it may be, does not necessarily suffice. Since Turing we have the complete theory of how to manipulate bits and is not that, what all computing boils down to? And why all that fuss about the problems of "the real world"? His theory proves, that all these problems can be solved, so why bother about actually solving them? Also such utterances are unmistakable and, undoubtedly, you recognize them.

So, the extremes are no good, we must blend them. But now we must be careful, for "blending" is no longer a one-dimensional question. It is not just "so many per cent. craftsman and so many per cent. scientist", but "this from the craftsman and that from the scientist". To drive home that message I shall describe to you a disastrous blending, viz. that of the technology of the craftsman with the pretence of the scientist. The craftsman has no conscious, formal grip on his subject matter, he just "knows" how to use his tools. If this is combined with the scientists approach of making one's knowledge explicit, he will describe what he knows explicitly, i.e. his tools, instead of describing how to use them! If he is a painter he will tell his pupils all he knows about all brushmakers and all he knows about the fluctuating price of canvas. If he is a professor of computing science, he will tell his students all he knows about existing programming languages, existing machines, existing operating systems, existing application packages and as many tricks as he has discovered how to program around their ideosyncrasies. And in a short while, he will not only tell what the manual says that should be punched in column 17 of the first card in order to indicate your choice of priority queue, but he will also tell and explain the illegal punching in column 17 that will place your program in the highest priority queue while only charging you for the lowest priority one. Again, the symptoms are unmistakable and, undoubtedly, you recognize them.

This disastrous blending deserves a special warning, and it does not suffice to point out that there exists a point of view of programming in which punched cards are as irrelevant as the question whether you do your mathematics with a pencil or with a ballpoint. It deserves a special warning because, besides being disastrous, it is so respectable! You see, on the one hand you stick to the problems of the real world and no one can accuse you of being overdemanding with regard to the powers of abstraction of your students, on the other hand you are as explicit as possible and everything you tell is the objective, undeniable truth. And when someone has the temerity of pointing out to you that most of the knowledge you broadcast is at best of moderate relevance and rather volatile, and probably even confusing, you can shrug your shoulders and say "It is the best there is, isn't it?" As if there were an excuse for acting like teaching a discipline, that, upon closer scrutiny, is discovered not to be there.... Yet I am afraid, that this form of teaching computing science is very common. How else can we explain the often voiced opinion that the half-life of a computing scientist is about five years? What else is this than saying that he has been taught trash and tripe?

With a little bit of knowledge of human nature, after the above tirade ~~tribade~~ against the wrong blending, all of you will now expect me to say that my sympathy is with the inverse blending. This expectation is correct: as teachers of programming we should try to blend the technology of the scientist with the pretence of the craftsman.

Sticking to the technolgy of the scientist means being as explicit as we possibly can about as many aspects of our trade as we can. Now the teaching of programming comprises the teaching of facts --facts about systems, machines, programming languages etc.-- and it is very easy to be explicit about them, but the trouble is that these facts represent about 10 per cent. of what has to be taught: the remaining 90 per cent. is problem solving and how to avoid unmastered complexity, in short: it is the teaching of thinking, no more and no less. The explicit teaching of thinking is no trivial task, but who said that the teaching of programming is? In our terminology, the more explicitly thinking is taught, the more of a scientist the programmer will become.

This, of course, raises the question of the feasibility of the teaching of thinking. In order to make this question realistic, we shall qualify it somewhat: knowing how to teach thinking will not imply that each student is also able to learn it. This need not deter us: in this respect "thinking" would not differ from any other subject that we try to teach. So, let us consider the question after this qualification: can thinking be taught? The blurb on the backside of my 1957 edition of Polya's "How to solve it." is quite positive: "Deftly, Polya the teacher shows us how to strip away the irrelevancies which clutter our thinking and guides us toward a clear and productive habit of mind."

Fine, but that is only the blurb: on the other side it has been remarked, that its first edition dates already from 1944 and that Polya's larger work on the same subject "Mathematics and Plausible Reasoning" has been cooly received by the mathematical community and has had at most a very minor influence on the teaching of mathematics at university level. Its cool reception by the mathematical community says at second thought, however, nothing against the feasibility of Polya's project. On the contrary! For its cool reception can also be interpreted as the rejection by the mathematical guild that feels threatened, as all guilds do, when the secrets of their trade are made public. To publish 30 years ago a book about the making of mathematical discoveries was heresy, as it still is in the eyes of many mathematicians today. And to quote from "Management and Machiavelli" by Antony Jay: "In corporation religions as in others, the heretic must be cast out not because of the probability that he is wrong but because of the possibility that he is right." In other words, the relative rejection of Polya's work on heuristics tells probably more about the intellectual inertia of the mathematical establishment than about his books themselves and I suggest you this time --unusual as the advice may seem!-- to believe the blurb.

I regard Polya's "How to solve it." as a promising and significant first step. It presents heuristics as a kind of checklist of standard questions which may be helpful in not overlooking a simple, but somehow unexpected solution, if there is one. When I first read it, I was somewhat disappointed by it, a disappointment that was a direct consequence of my already being deeply involved in programming: I felt that my problems as a programmer were for a large portion beyond the scope of what Polya covered. At first I hesitated to say so aloud, because stressing the exceptional nature of one's own field is usually a sure way of making oneself utterly ridiculous. But after careful consideration I concluded that the intellectual challenge presented by the programming task is, indeed, as unprecedented as the high-speed automatic computer itself. And it had caused in my mind a shift of attention from "how to discover the unexpected" towards "how to avoid unmastered complexity", towards "how to reduce the demands made on our quantitatively limited powers of reasoning".

You must take my word for it, that past experience has made me a firm believer that this newer aspect of thinking, i.e. how to avoid unmastered complexity, can indeed be taught. This strikes you perhaps as a strong statement, it becomes only stronger when you also know that I am usually not given to unwarranted optimism. Among other things it can be done by the identification and subsequent description of the more productive "complexity generators".

But it is good to remember, that there are also some intrinsic limits to the degree in which thinking can be taught explicitly, "in the scientific manner" so to speak. To quote Polya: "The first rule of discovery is to have brains and good luck. The second rule of discovery is to sit tight and wait till you get a bright idea. It may be good to be reminded somewhat rudely that certain aspirations are hopeless. Infallible rules of discovery leading to the solution of all possible mathematical problems would be more desirable than the philosopher's stone, vainly sought by the alchemists. Such rules would work magic; but there is no such thing as magic. To find unfailing rules applicable to all sorts of problems is an old philosophical dream; but this dream will never be more than a dream." And it is there, where, unavoidably, the teaching of thinking becomes more like the teaching of a craft, where the student picks up by unconscious imitation: it is here, where, as in the good old days of the guilds, an inspiring master can do wonders and can found a School by his example.

To those of you that are in the academic teaching business I have only one urgent plea: please be not ashamed of the extent in which your teaching of thinking is "unscientific"! It is good to remember that all the unfathomed depth of the human mind is already at play in the process of human communcation. We have --despite what psychologists, paedagogues and the like may think-- not the faintest idea how knowledge, insights and habits are transferred. It is not unlikely, that the actual transfer is always by imitation, and that all the explicit teaching in the scientific tradition is no more than giving the student some verbal handles, which are no more than an aid to memory. If this is true, then all purely "scientific teaching" --i.e. the explicit rules and no more-- is bound to be, and to remain forever, a barren activity.

To end up my talk I would like to tell you a small story, that taught me the absolute mystery of human communication. I once went to the piano with the intention to play a Mozart sonata, but at the keyboard I suddenly changed my mind and started playing Schubert instead. After the first few bars my surprised mother interrupted me with "I thought you were going to play Mozart!". She was reading and had only seen me going to the piano through the corner of her eye. It then transpired that, whenever I went to the piano, she always knew what I was going to play! How? Well, she knew me for seventeen years, that is the only explanation you are going to get. Since then I believe that it is vain to try to understand what goes on in the classroom between who teaches and who learns, and that having no model of that process is safer than having one, of which the crudeness has been forgotten.

I thank you for your attention.

5th March 1975
Plataanstraat 5
NUENEN - 4565
The Netherlands

prof.dr.Edsger W.Dijkstra
Burroughs Research Fellow

G.Polya "How to solve it." (Anchor A 93) Doubleday and Company, Inc, Garden City, New York, U.S.A., 1957

Antony Jay "Management and Machiavelli", Penguin Books Ltd., Harmondsworth, Middlesex, England, 1970