AB29.3.2 ALGOL Colloquium - Closing Word: Zürich, 31.5.1968

by Niklaus Wirth

[The following is the author's text of the "closing word" to the Zürich Tenth Anniversary ALGOL Colloquium, reported in AB28.1.1. - Ed.]

The world of computer programming is in disorder and disorientation. Little progress has been made lately towards better understanding of the principles of computing. In spite of the existence of sophisticated languages, most programmers still work with octal, hexadecimal, or other cryptic dumps, with jumps and condition codes and other chaos promoting devices, worry about saving a bit here or there and a cycle now and then, and usually cannot bet a penny on whether a program written holds its promises or not. On the other hand, our vocabulary has expanded to unforeseeable dimensions. Interrupt, reentrant, time-sharing, modular, environment-adapted, paged, syntactic sugar, macro definable, dataset, push-down-controlled, micro-programmed, coercety and reference-to-union-of-MODES-and-PREFIX-structured-with-a-MODE-named-TAG-mode-row-of-row-ROWSETY-NONROW-slice, are examples of phrases not known 10 years ago. It is all part of a phantastically successful sales and habilitation campain of the emerging and confused computer and computer-science world. We witness the birth of an attempt by mankind to cope with the phenomenon of complexity, to specify and control logically complex processes handling vast amounts of complex structures of data. It is clear that such a new field of endeavour is prone to commercial exploitation. It is tempting and relatively easy to intrude nomansland and erect flags at exposed sites, but it is another matter to conquer and master the land, to cultivate it, develop a well-organised economy and to subject it to a sound and coherent system of laws.

We have hastily intruded the world of complexity, have produced and sold complex equipment, have erected some spectacular land-marks, have clobbered the land with many billboards (our vocabulary), but are left with a feeling of being lost. As scientists, we have attempted to see beyond our immediate environment, to bring some order and orientation into the new land, to develop and discover its underlying basic laws. But so far, we haven't progressed very far. The prospectors are much faster and more ambitious, and with their new discoveries constantly overthrow our modest results.

We have recognized that the cornerstone of the new field of complexity is that of specifying recipes and data structures on which our recipes operate. Such specifications must be given in terms of some logical system, which we have become accustomed to call language, or programming language. It is evident then, that our main effort in developing order and soundness concentrated on the design of languages. The language we speak influences profoundly our ways and powers to think and express ourselves. It should be clear, however, that the language is nevertheless only a tool, a medium, and that education must go beyond, far beyond, the teaching of language. I think, as scientists, we have concentrated too long on the refinement and sophistication of languages, so that we had to neglect the systematic education in their disciplined application, while the demands on applications programming continued to grow rapidly. Which are the results of this faux-pas ?

In order to find out, I interviewed several people:

First I met a scientist who since many years is engaged in problems of numeric computation and large scale number crushing. "What is your programming language?", I asked: "FORTRAN. It is the only tool, except pure machine code (which I abhore), that guarantees reasonable efficiency. Our main computations are floating point arithmetic and indexing. FORTRAN is sufficiently restrictive in its indexing facility so that extensive automatic optimization is possible inside loops. We are not very much interested in new and supposedly more powerful languages (such as ALGOL), because we have already such a large program library that any change-over would be out of question because of the necessarily large reprogramming effort."

Secondly I went to see a staff member of the state administration. "What is your programming tool?" I asked again. "Well, there isn't much choice but plowing through core dumps." "No no, I wasn't referring to debugging aids, but to your programming language." "Just about a year ago we switched to COBOL, which allows us a greater degree of machine independence. Our main concern is that of record- and file structures, sorting and merging and input/output editing. COBOL's capabilities in this respect are quite versatile. It is somewhat wordy, but one gets used to it, constructs quickly one's own repertoire of abbreviations and learns to avoid those features that seem dangerous and mysterious. Moreover, COBOL is designed for the minds of our programmers: we think in terms of layouts of characters, and as on paper we use templates of celluloid, we define data structures which group and name our character strings.

Nobody has ever seen a real number in a computer, or have you; but characters we can imagine very well." "Do you think of ever switching to another language?" "Hardly, I don't see a need; and our large program investment wouldn't warrant such an effort".

Thirdly, I interviewed the director of a University computing center, where a new machine had just been installed. "Are your user's happy with the new facility?", I queried. "Mixed feelings! Particularly our Algol customers, of whom we had a fair share so far, are disgruntled. They all have to change their input/output sections from using FORTRAN to the new rules of the Knuthput. Many don't survive and go back all the way to FORTRAN. It is a pity.

Actually, all we would have needed to make Algol really a success here, would have been some minor additions to the range of data types: double precision, complex, and character. With the availability of character arrays we could define our own convenient standard I/O conversion procedures, and would not have to teach an abominally complicated mechanism, which everybody recognizes as being artificially grafted on top of purist Algol 60".

Fourthly, I visited a colleague who had been engaged in programming language design and teaching for quite a while, and I asked him about his present preoccupation. "I am discouraged! I thought I had recognized the short-comings of the commercially available languages; they provide no guidance to programming discipline, and lack a logically coherent structure. Algol 60 had provided an answer and a solution. Unfortunately,

it lacked some features which many practictioners badly need. It was relatively easy to remedy some defects and some defaults of Algol 60. We had implemented such an extension of Algol; it contains the data types long real, complex, and character strings, along with one other major extension. But guess how many requests we have received for this system. None! Nobody is interested in a new language, particularly if it is not supported by the big manufacturers or has received the blessing of some standards committee. But I see no chances that either a big manufacturer or a committee will ever produce a language acceptable to our standards or clarity, simplicity, and rigor. As a result, everybody is designing his own language or software package. The old dream of a universal language is gone with the wind."

In the fifth place, I met a friend engaged in hardware design. "I heard that you have recently switched to language design", I said: "Not exactly. We are still planning new machine organisations. But you have no idea what archaic methods are used in this field, and what cataclysmic blunders are committed as a result. My aim is to develop a tool which permits the precise specification of hardware down to the level of component interaction and pulse timing. But there simply is no convenient tool in which to express these problems. We will probably have to develop a language and program it interpretively in PL/1; I can foresee that it will be horribly inefficient, and I would prefer some assembly language; but PL/1 is company policy!"

In the last place, I went to ask for the opinion of a college teacher. "What language do you teach now?". "I have finally switched to PL/1", she said. "I have some pride in teaching in a clear and understandable way, and to organise the material in such a way that everything I say has its logical place, holds its validity with everything which will follow, makes use only of concepts previously introduced, and appeals to common sense. Yet there is now not a single lesson where I can avoid blushing and cursing myself for teaching something which contradicts my own principles and standards of scientific rigor, not a single lesson where there isn't a student asking a "why", and I have to reply "don't know, just accept it as it stands".

"So why have you chosen to teach PL/1?", I asked. "I used to teach Algol, where things weren't ideal, but a lot better in those respects. But pressure was mounting to teach a language which the students could readily use after leaving the haven of school. Employers don't ask "do you know the principles of programming?", but rather "do you speak FORTRAN?". In order to avoid making this step backwards, I chose PL/1, which first seemed to be a satistory compromise."

"And what do you think of the new Algol?", I continued. "New Algol?" I pulled out my copy of the draft report on Algol 68 and showed it to her. She fainted. I guess she had had a hard day; and then I realised how teachers live under the constant threat not to be able to cope with the ever growing demands of an ever more complicated science.

But enough fiction! What has the Algol Working Group been accomplishing? It has managed to live with the belief of being an important institution. It has not clearly recognized the down-to-earth problems of those who are mostly indifferent toward ALGOL, and it has for too long a time let down those which were waiting for an amended ALGOL.

It has merely produced reports on a subset of Algol and on a set of input-output procedures, which were duly awared with the official IFIP stamp. Apart from this its members met regularly once or twice a year. The meetings followed a standard pattern: first came disarray and dispute, then a general feeling of discouragement and despair, and at the end of the week emerged one or several saviours promising to work on a solution until next time. The rest of the members dispersed and forgot about Algol. Finally the saviours had worked themselves so deeply into subject matter, that the rest couldn't follow and understand their thoughts and terminology and longer. Meetings became principally tutorials. But even this didn't avert that everyone started to speak a different language. What hampered progress even more, was the fact that a goal had never been specified with sufficient clarity. The implied and rather vague goal was the specification of a universal language, a sensible goal in 1960, even 1964, but an utopia in 1968; a goal which, if pursued faithfully, invariably lead towards a monster language, a species of which there already exists a sample hardly worth nor possible to compete with. Having proved to be too rigid and ineffective, and with a vage guiding motive which has been declared obsolete by a fast moving world of facts, the Working Group has failed and should draw the consequences: dissolve.

Would there ever have been an alternative? I believe so. It would have required the renunciation of the immodest ambition to erect another monument or mile stone along the road of progress in computer sciences. It would have required a shift from a fanatic perfectionism in formal definition methods towards honest consideration of pragmatics. It would have required the concentration of effort on a much more restricted goal. Of the people I visited before, I believe we should first have served the last one, the teacher. By defining and making available a simple, logically and pragmatically sound, appealing and efficiently implementable language, fully equipped with an official stamp, an enormous service could have been made to the computing world at large. People trained on such a language would have carried over their acquired working habits, their ways of thinking and designing and their standards of quality into their places in research, development, application, and education in computing. Such a language we had expected from the illustrious gremium WG 2.1 a few years ago. The emphasis on a simple and didactically appealing language would automatically have resulted in the development of a nucleus of a self-extendible language, perhaps the most promising approach of development. Such a language provides a facility for programmers to define data structure patterns, and operators which apply to these patterns. The crucial point here is that the language nucleus must be of utmost simplicity, and every aspect of the nucleus language must be very efficiently implementable on any reasonable computer organisation.

It is deplorable that the Algol Working Group has missed its chances. It has displayed a poor judgement for timing its activities, selecting tasks which are in measure with its capabilities, and sensing the important areas of need. It has given the impression not to care about realities.

It has been sitting on an ivory tower, and when it was recognized that the tower was leaning, it was too inflexible to make a decision for further action. But this seems to be the fate of committees. Next week, we will carefully watch the tower of Pisa.

In closing, I wish to thank Professor Rutishauser and his collaborators for organising this "happy birthday party", and thus making an interesting get-together and exchange of ideas and opinions possible.

Mathematisches Institut der
Universität Zürich.