<u>The summing up</u>.

After having worked a full month on EWD659 through EWD662 --dealing in turn with the four language proposals submitted to the DoD-- I feel an urge to collect and order my impressions of the whole undertaking. Due to lack of more time, this note is produced in the same manner as the previous four: it is composed while sitting at the keyboard. For linguistic imperfections I offer my apologies.

My overwhelming impression is that, in particular in combination with the Revised "IRONMAN" Requirements --about which more later-- the competitive situation in which the four design groups had been placed, has been disastrous. Instead of trying to design the most adequate programming language, they have all four tried to get the next contract, and that is a completely different thing! It has had two nasty consequences.

Firstly the four reports I studied were all an almost inextricable mixture of technical documentation and salestalk, and that made their study unnecessarily difficult and unpleasant. It made the reading unpleasant because quite often the salestalk was so close to dishonesty or nonsense that my ability to approach the design with feelings of sympathy was severly strained. It made the study unnecessarily difficult in the sense, that they showed what was intended to be possible, but left to the reader to discover for himself what had been ruled out.

Secondly the four designs have been much to "obedient" with respect to the IRONMAN Requirements. Now, if requirements are perfect, it should be a pleasure to try to comply with them; if requirements are not sensible, a design team should have had the liberty to point that out, but on the whole that liberty has hardly been taken; if requirements are contradictory, a design team should have pointed this out, but on the whole the designs have tried to cover this up. It is here that teams may choose between the qualifications incompetent and dishonest; I blame the competitive situation for having placed them in that uncomfortable position.

From the above it follows that I view besides the competitive situation the IRONMAN Requirements themselves as a major cause of the miserable failure.

A main complaint about the IRONMAN Requirements is that they are not requirements in the true sense of the word: instead of listing the goals to be reached, IRONMAN already starts the design by prescribing "features" from which it is often hard to reconstruct or guess which sensible goal they are supposed to serve. (An obedient designer is in still worse a position when on the one hand it is clear which purpose the prescribed feature is supposed to serve, while on the other hand the prescribed feature is no good for that purpose!) I have been told that the major and most common error systems analysts make is to produce a rought design instead of formulating the requirements; the IRONMAN Requirements seem to be a school-example of that error.

The unavoidable result of that mistake is that the requirements become overspecific and --the authors of the requirements being amateur language designers-- silly and contradictory. A "low level" example is provided by section "2. General Syntax" where on the same page (!) it is stated (in 2D): "Source program line boundaries shall be treated like spaces." and (in 2I): "shall allow stand alone comments (i.e., a comment introduced by a special symbol as the first nonspace character of each line)." What glorious nonsense! If source program line boundaries shall be treated like spaces, how can you talk about the first nonspace character of each line? And how overspecific! And how confusing! I thought every language designer knew the essential difference between the abstract syntax of a programming language and the question of text representation. If this mistake had been carried one --but only one!-- step further, the requirements had included on "General Syntax" an upper bound on the maximum number of holes to be punched in each single card. The proper reaction to such silly requirements is, of course, to ignore them. It would have been much better, however, if they had not been given at all, because by their mere presence they induce a general state of cynicism.

Other examples of overspecification are (in 6B): "Explicit statement delimiters shall be required." and (in 6G): "There shall be an explicit mechanism for control transfer (i.e. the "go to")." It is clearly impossible to claim that either semicolons or go to statements are essential for the envisaged military strength of the USA; they are not like the famous horse-shoe's nail. Both are means towards an end, and the requirements should have stated the end instead of prescribing particular means.

With respect to aliasing and side-effects again the requirements are half-hearted: without taking a firm stand on either of them restrictions are imposed on the language that only don't encourage the phenomena to occur. (Amazingly enough the IRONMAN Requirements don't show any trace of the consideration that a programming language could be blamed for the property of requiring a run-time check for the detection of "use" of uninitialized variables!)

One of the sentences that, in retrospect, has been most harmful is (in 3A): "The language shall be strongly typed." This was terrible because it is not clear at all, what should be meant by it. The following sentence, offered as explanation, is most certainly not an further definition of what "strongly typed" should mean: "That is, the type or mode of each variable, array and record component, expression, function, and parameter shall be determinable at translation time." You see, the requirement as described here is very easily met by introducing only one universal type and defining all those things to be of that one and only type. But we can be pretty sure that that was not meant!

The issue of types has been further complicated by the requirements of section 11 "Specifications of Object Representation" (which I couldn't make sense of). In most of the proposals , as a result, a very messy notion of type, frequently not free from contradictions, is proposed. Disregarding the IRONMAN Requirements I have spent a day of reflection, trying to get a clear picture of the purposes a notion "type" could be asked to serve. I tried to come of with a well-justified choice of what "strongly typed" should or could mean. (I may devote a later EWD to it, this "summing up" has to be off my chest first.) I think I managed to come up with a few very well justified candidates for the notion "type". In all cases, however, I found their purpose defeated by (in 7A): " functions [...] and procedures shall be extendible to new data types (i.e. overloading shall be permitted)." It was an illuminating exercise!

It is so illuminating because it shows in a nutshell what havoc is created by not stating your goals but only prescribing partial means intended to solve your problems. It has seduced the authors of IRONMAN to pose a requirement defeating the purpose of another requirement. And it is not surprising to see exactly in this area a sizeable part of the complication, contradiction and mess in the submitted languages.

It makes also quite clear why the new programming language cannot be expected to be an improvement over PASCAL, on which the new language should be "based". (I am pretty sure that the new language --if it ever gets designed at all-- will be much, much worse than PASCAL if they proceed in this fashion.) You cannot improve a design like PASCAL significantly by only shifting the "centre of gravity" of the compromises embodied in it: such shifts never result in a significant improvement, in the particular case of PASCAL it will be extra hard to achieve any improvement at all, as most of its compromises have been chosen very wisely. What is wrong with PASCAL is that its notion of "type" --although on the whole a step in the right direction-- is still blurred. And you can never improve significantly on PASCAL without having clarified this conceptual trouble spot.

The moral of the story is that the DoD shouldn't have handed out four contracts for the design of a "strongly typed" programming language without knowing what it was asking for, and why. The DoD would have gotten much more value for its money --I guess-- if it had sponsored a few research contracts aimed at exploring whether the vague notion "strongly typed" can be given a clean, purposeful content, and if so, which are the options. These research contract would have been infinitely more helpful, orders of magnitude cheaper, and the responses could be measured in terms of pages --all of them making sense-- instead of kilograms obfuscating development documentation.

I would suggest that the same moral applies to the notions "aliasing", "side effects", and "enforceable language restrictions" (as in 1F).

* * *

In all four language proposals two important lessons we have learned from the past have been ignored. The one lesson is that a programming language should be defined without reference to its implementation and, a fortiori, without reference to aspects of the implementation such as compilers and computers. The other lesson is that the writing of a reference manual for a programming language absolutely requires someone of Peter Naur's intellectual calibre. For the fact that these vital lessons have been ignored the IRONMAN requirements themselves can be blamed. Firstly IRONMAN itself blurs the distinction between a programming language (definition) and its implementation, secondly IRONMAN fails to require that the reference manual should be short and that its English should be perfect. (I am not intimately familiar with the organization that grew PL/I, but fear that some of its more disastrous mechanisms have been at work in the DoD as well.)

* * *

I quote from a letter I wrote to a friend and colleague some weeks ago:

"It shows once more what an absolutely amazing job Peter Naur has done 18 years ago. Although I wouldn't recommend ALGOL 60 as a standard, I would like to recommend the quality of the ALGOL 60 Report as a standard for programming language reference manuals!

"I did not start with high expectations: the requirements almost exclude a nice language --they seem to have done so indeed-- and the documents had to be produced under all sorts of commercial pressures. But I was not prepared for such junk as I have seen. Isn't this world a disappointing, depressing place? Do these people not know how much care such a design requires? Or did they cynicly ignore that requirement?

"We all know how terribly difficult writing is, and if a text has to be produced with such a cruel deadline, we shall blame no author for minor imperfections: we all send our stuff to our friends solliciting comments and suggestions for improvement and they may have lacked the time. But I find it hard to forgive such sloppy texts [...].

"I have now done two of them, but need one or two days off, to collect some courage, before I dare embark on the third."

In his answer he discussed the phenomenon that, compared to PASCAL (with all its imperfections and shortcomings taken into account), the proposed languages are such a gigantic step backwards. I think he gave the correct explanation. I quote from his answer:

"[...] let us admire PASCAL, not for its "perfection" or for its "power", but for its <u>practicality</u> as a <u>compromise</u> between valuable programming ideas and existing machine designs, and the kind of ideals that we would like to pursue. That is why IRONMAN is so misguided and dangerous. You cannot improve a compromise by asking it to meet more exactly the irreconcilable objectives which it was trying to compromise between. And especially not if you add a few more even more irreconcilable objectives! Oh dear, how can we ever make them see sense?"

One fatal mistake is not to recognize where the outcome of (possibly new) scientific research is required. Another fatal mistake is to state objectives while disregarding the technical question of their (combined) feasibility. Both mistakes have been made.

* * *

And this concludes a month of most depressing work. Why does the world seem to persist so stubbornly in being such a backward place? Why do people refuse to learn from the past and why do they persist in making the known and well-identified mistakes again? It is all very saddening.

Of ALGOL 60 C.A.R.Hoare once remarked that it was a significant improvement over almost all of its successors. What can we do to prevent PASCAL from sharing that fate?

2nd of April 1978

Plataanstraat 5
5671 AL NUENEN
The Netherlands

prof.dr.Edsger W.Dijkstra
Burroughs Research Fellow