B. SAI TEJA

201301005

The Emperor's old clothes

– C.A.R.Hoare

Abstract:

In this article, the author recounts his experiences in the implementation,design and standardization of computer programming languages and issues a warning for the future.

Description:

- He states that besides failures are much more fun to hear afterwards,they are not so funny at the time and his first task was to implement for the new Elliot 803 computer,a library sub-routine for a shell.

For him, the important task was of designing a new high level programming language for the company's next computer which was Elliot 503 having the same instruction code as the existing Elliot 803 but runs sixty times faster.

In 1961,a course offered on Algol 60 in Brighton,England with peter naur,Edsger .W.Dijkstra where Hoare learned first about recursive procedures and has seen how to program the sorting method which he had found difficulty in earlier in explaining and wrote Quicksort procedure on which his career as a computer scientist started and he have regarded it as the highest goal of programming language design to enable good ideas to be elegantly expressed.

By Robert cook's advice to implement Algol 60 ,he adopted certain basic principles which are:

1**. Security** (in terms of syntax)
2**. Brevity of the object code produced by the compiler and compactness of run time working data** which means the object code i.e.., the machine language code should be precise so that run time of the program will be reduced.
3. **Entry and exit conventions for procedures and functions should be as compact and efficient.** This statement says that the functions declaration in the program should be well balanced so that the program will work out.
4. **Complier should use only a single pass** which is as the compiler was structured as a collection of mutually recursive procedures where each one is capable of analyzing and translating a statement,an expression etc.., The compiler was designed and documented in Algol 60 and then coded into decimal machine code by using an explicit stack for recursion . During that time,it is highly controversial that without Algol 60 concept for recursion ,we could not have written the compiler at all.

Two things that first is we certainly want programs to be read by people and people prefer to read those things in one single pass and second one is for the user of a time-sharing or personal computer system,the interval between typing in a programand starting to run that program is wholly unproductive and it can be minimized by the high speed of a single pass complier.

To structure a compiler finally according to the syntax of its input language makes a great contribution to ensuring its correctness.

The frst version of the compiler was delivered in middle 1963 and many of the customers told him of their fon memories of the Ellot ALGOL system and the fondness is not just due to the nostalgia but to the efficiency,reliability and convenience of that early simply ALGOL system.

In october 11,1963 he suggested to pass on a request of our customers to relax the ALGOL 60 rule of compulsory declaration of variable names and adopt some reasonable convention such as that of FORTRAN where it was pointed out by this suggestion that the redundancy of ALGOL 60 was the best protection against programming and coding errors which could be extremely expensive to detect in a running program.

He states that the way to shorten programs is to use procedures not to omit vital declarative information.

There was also other proposals for the development of new Algol 60 in which the SWITCH declaration of ALGOL 60 should be replaced by a more general feature,namely an array of label-valued variables and that a program should be able to change the values of these variables by assignment.

His second language proposal makes him proud which is a "case expression" that selects between any number of altrenatives according to the value of an integer expression because it raises essentially no problems either for the implemator,the programmer or for the reader of the program.

He then designed another project which is Elliot 503 Mark II software system to provide a range of operating system software for larger configurations of the 503 computer, with card readers, line printers, magnetic tapes, and even a core backing store which was twice as cheap and twice as large as main store, but fifteen times slower.

It comprised:
(1) An assembler for a symbolic assembly language in which all the rest of the software was to be written.
(2) A scheme for automatic administration of code and data overlays, either from magnetic tape or from core backing store. This was to be used by the rest of the software.
(3) A scheme for automatic buffering of all input and output on any available peripheral device,--again, to be used by all the other software.

(4) A filing system on magnetic tape with facilities for editing and job control.
(5) A completely new implementation of ALGOL 60, which removed all the nonstandard restrictions which we had imposed on our first implementation.
(6) A compiler for FORTRAN as it was then.

In designing ALGOL 60 complier,it was not be delivered as its speed of compilation was only two characters per second which compared unfavorably with the existing version of the compiler operating at about a thousand characters per second. They soon identified the cause of the problem that it was thrashing between the main store and the extension core backing store which was fifteen times slower.He also stated that users of microprocessors benefit from a similar protection but not for much longer in the present day.

He then stated about Multics which describes the concepts and features of the newly announced OS 360, and of a new time-sharing project stating that they were far more comprehensive, elaborate, and sophisticated than anything that he had imagined, even in the first version of the 503 Mark II software which tells that eatures of the newly announced OS 360, and of a new time-sharing project.

He then explains that Our main failure was over ambition.
“The goals which we have attempted have obviously proved to be far beyond our grasp”.There was also failure in prediction, in estimation of program size and speed,of effort required, in planning the co-ordination and interaction of programs,in providing an early warning that things were going wrong.There were faults in our control of program changes,documentation,liasion with other departments,with our management and with our customers and also included that you shouldn't trust us intelligent programmers as we can think up such good arguments for convincing ourselves and each other of the utterly absurd. Especially don't believe us when we promise to repeat an earlier success, only bigger and better next time.

The last section of their inquiry into the failure dealt with the criteria of quality of software which is measured by a number of totally incompatible criteria that must be carefully balanced in the design and implementation of every program.He explains that he hadn't seen why the design and implementation of an operating system should be so much more difficult

than that of a compiler and that is the reason why he have devoted his later research to problems of parallel programming and language constructs which would assist in clear structuring of operating systems constructs such as monitors and communicating processes.

He had proposed that a programming language definition should be ormalized as a set of axioms, describing that the desired properties of programs are written in the language and he thought that carefully formulated axioms would leave an implementation the necessary freedom to implement the language efficiently on different machines and enable the programmer to prove the correctness of his programs.He has written his first paper on the axiomatic approach to computer programming which was published in the Communications of the A CM in October 1969.

He concluded that there are two ways of constructing a software design in which one way is to make it so simple that there are obviously no deficiencies and the other way is to make it so complicated that there are no obvious deficiencies but the first method is far more difficult as it demands the same skill, devotion, insight, and even inspiration as the discovery of the simple physical laws which underlie the complex phenomena of nature. It also requires a willingness to accept objectives which are limited by physical, logical, and technological constraints, and to accept a compromise when conflicting objectives cannot be met and no committee will ever do this until it is too late.

Hoare has given two wonderful statements in which first one is When any new language design project is nearing completion,there is always a mad rush to get new features added before standardization. The rush is mad indeed because it leads into a trap from which there is no escape. A feature which is omitted can always be added later,when no design and its implication are well understood. A feature which is included before it is fully understood can never be removed later. And the next one is Programmers are always surrounded by complexity; we cannot avoid it. our applications are complex because we are ambitious to use our computers in ever more sophisticated ways. Programming is complex because of the large number of conflicting objectives for each of our programming projects .If our basic tool,the language in which we design and code our programs,is also complicated,the language itself becomes part of the problem rather than the part of its solution.

He actually knew that it would be impossible to write a wholly reliable compiler for a language of this complexity and also impossible to write a wholly reliable program when the correctness of each part of the program depends on checking that every other part of the program has avoided all the traps and pitfalls of the language. He mentioned that there reliability is the quality that cannot be purchased and the price of reliability is the pursuit of the utmost simplicity where it is a price in which the very rich find most hard to pay and the original objectives of the language, included reliability, readability of programs, formality of language definition, and even simplicity. Also proposed that An unreliable programming language generating unreliable programs constitutes a far greater risk to our environment and to our society than unsafe cars,toxic pesticides or accidents at nuclear power stations.Be vigilant to reduce that risk , not to increase it.

Finally ,In the last word he ended the article with an emperor story that there live an emperor who was very fond of clothes and minister in the court thinks that the tailor in their kingdom will be the best to stich clothes for the king but it went failure when the tailor says that he can't stich clothes to the king which tells that the minister he himself assumed that the tailor is the best choice for stiching clothes for the king but by looking at the tailor's staement,it tells us that he is not the best tailor and the author compares he himself with the tailor in designing ALGOL 60 language compiler.