The Emperor's Old Clothes :

My Views and Comments

C. A. R. Hoare, in this exquisite contribution to the computing world, makes significant views based on his experiences throughout his career. His major contributions include the axiomatic approach to formalisation of programs, the Quicksort algorithm and many more. He has a recurring contribution to the development of ALGOL. All through the lecture, he speaks about his life as a programmer and puts forward all of those situations where he failed as a programmer, as a consultant and also as a project head, hoping that programmers today learn from his mistakes.

As he narrates, he also tends to show in a very strong sense how 'Simple yet Powerful' should be the main characteristic of any programming language. He says programming languages are used to represent the solution of a complex problem and by making them complicated and bulky, they seem to become a part of the problem itself rather than of the solution. He relates this ideology to the story where an emperor becomes so obsessed with clothes that he keeps on wearing them one on another, until one day when he realizes his mistake and leaves, people fail to realise that the clothes actually had no emperor. In the context of programming languages, he says one should not rush into adding new "unnecessary" features to the language, leaving behind the efficiency, performance and simplicity of the language.

He also speaks about the importance of having a formal method to represent languages. Languages must be provable so that their output becomes reliable. In this context, he introduces his version of formal description where he uses axioms to represent programs. Then he proves their correctness based on principle of consistency in an assertion at a state before and after running a command.

While at the early stages of designing a Programming Language, he set out four principles which are valid even today. The first being Security, that a syntactically incorrect program should never be executed and vice-versa. The second one was about the brevity of the object code and compactness of run time working data as extra capacity can always be used for other worthy results like increasing the quality of his program, its simplicity, its ruggedness, and its reliability. His third principle was that the entry and exit conventions for procedures and functions should be as compact and efficient as for tightly coded machine-code subroutines. This was one pointer towards functional programming. The last principle was about the compiler being a Single Pass Compiler.

I feel Hoare has done more contributions to the thought process in guiding people to understand what they actually needed. His notion of simplicity gave a way to Functional Programming. And, Proving Programs has always been a highly researched topic in the community. I believe he had clear intuitions about futuristic demands in the world of computers.