# One man's meat: Part 2—now let's pretend

*by* F. I. Musk*

**This is the second of a series of papers which attempt to define one computer user's philosophy. The philosophy is extended here from the provision of information for decision making into the decision area itself.**

In Part 1 of this series (Musk, 1966) the following philosophy was advanced.

> "To an industrial concern, the initial function of computing is to provide management at all levels with all relevant but no other data, in the most easily assimilated form, at the precise moment when a policy decision has to be taken."

CRESTS (Craig, 1966) was considered to be a tool useful in urging the translation of this philosophy into practical terms.

The philosophy as it stands has its limitations. It is but one step removed from "data processing" which is the mere aping of earlier procedures for accountancy and routine administration, even though these procedures have been streamlined by systems study. It is true that it has a beneficent effect upon the problems of decision makers. It can save time in data digging, a crucial advantage when, for example, developing the design of a new factory. It can, where time must be taken by the forelock, limit to some extent the area of risk. Still, it impinges upon the decision area only tangentially. It pretends to inform, but never to advise.

If another tenet is to be added, this must deal with the decision area itself. This is not to assume the functions of management. A machine can never do that, unless management is concerned only with routine decision. A computer can be an *aid* to decision making. Management can take a "computer-aided" decision.

Let us assume that the following is the second tenet in our philosophy.

> "To an industrial concern, the second function of computing is to provide management with a means of calculating a choice of decisions, and to give an indication of the likely results of each choice."

The first tenet provides for an appreciation of an existing situation. The second indicates a choice of suitable future actions, and the results likely to accrue from each such action. It says no more than that computers should be used to implement the fruits of operational research investigations. There is nothing novel in this idea, but of all the thousands of case histories that have been talked about, written about, since the Operational Research Club became a society, it is to be wondered how many were implemented, how many once implemented are still extant, how many once implemented and still extant are effective.

The University of Aston Designate (when it was the Birmingham College of Advanced Technology) once sent a group of students who had been introduced to statistical quality control, out into the field (certain engineering companies in the area) to learn the practical lessons of quality control from firms which had embraced the technique a year or two ago. They reported back that very few firms on their list still used the technique, and those with pretensions to do so were simply enacting a hollow ritual.

What had happened? Like the production machines themselves, statistical quality control must be regularly maintained, else it will break down. Things change. External conditions, and conditions in the machine shop, change. Old products evolve, and so do their specifications, or they are supplanted by new products. The SQC man moves on and tells his successor what to do, but not why he is doing it.

The same fate awaits optimum schemes developed by the use of operational research, unless they are regularly tended. The bases on which they were originally conceived are gradually eroded away, and their pure forms become encrusted with secondary trivia. Unfortunately, and particularly if they were programmed (at great effort and expense) for a computer, such decayed schemes, although now far removed from the optima they were contrived to attain, are likely still to be workable, since the cunning operational research worker tested them out for stability. That an SQC scheme is no longer valid can be proved by complaints from customers, but that a computerized operational research scheme no longer attains optimum yield, or productivity, or joint cost of stock holding and changeover, is often very difficult to determine.

What can be done? Let us assume that it is the responsibility of someone to keep fresh the data on which the computerized scheme relies. This can be an automatic updating of the program parameters derived from our constantly refreshed Data Base. But not only do the data change. So do manufacturing processes, both in machine renewal and technique. So also do sales patterns and management policy. The need for such change presents a formidable problem in program maintenance. Organizations in which computerized

* *Computer Dept., Courtaulds Ltd., Matlock Road, Coventry.*

operational research schemes develop most readily tend to be the most vibrant of their kind. They react most quickly to changes in their environment, and as a result their production processes are turbulent streams of change, in which optimum decision schemes must be supremely adaptable if they are to remain valid and retain their original direction.

Some operational research schemes, but not many, are capable of solution by such wooden methods as linear programming. Computer programs of this nature require only the continual refreshment of their data. How do we cope with the others?

According to Spock, children at play are learning the skills required in adulthood. Sand-table exercises and tactical exercises without troops are still considered necessary pretences for the training of army officers. We play business games in a competitive atmosphere to teach fundamental precepts to potential managers. Wherever the real thing is either dangerous or costly, men immediately resort to mock-ups and models. Analogy is inherent in us. So the idea of simulating the behaviour of factories under the stress of a variety of production loads, or of a transport fleet with a variable collection and delivery pattern, or a pilot plant, or an overloaded road system, or the flow of paper work in an office, or of a whole firm struggling for survival in an economic environment, is not strange to us.

The physical causes of change in an organism (say a monkey or a weaving shed) are relatively easy to set down one by one. What is difficult is to predict the complex patterns which emerge when all these causes are working simultaneously. If we had a simple means of building a working model of the system, feeding in the elements of change continuously in all their rich complexity, we could pretend to change planning policy, or product loading, or customer behaviour, or stock holding, or documentation—anything. The result would be an opportunity to study, over long periods of simulated time, the several effects of various policy changes, on the system under review.

In effect, a simulation program on a computer is just such a working model. All computerized operational research schemes simulate the behaviour of the system they pretend to optimize, even formalized stock control or scheduling schemes, and so they are handled by the computer essentially as simulation programs. But think —to maintain a simulation routine—what a tedious task by machine, symbolic or mnemonic coding!

Let us return to our second tenet. We are not saying to management, "You tell us what policy you will adopt and we shall show you the likely result". We are not saying "Leave it to us and we shall tell you your optimum policy, take it or leave it". We do say "You give us your discrete policy field, and we shall supply the likely result of each policy, as well as providing, if you want it, the optimum policy, even if it lies outside your policy field".

It is obvious that we must already know intimately the system within which policy decisions are going to be made. We would already have, if at all possible, an up to date motion picture of the system in the form of a simulation program. Apart from the difficulty of keeping the program up to date, we do not know what policy question is likely to be thrown at us. If we were working in a low level language, the policy question could so disrupt the simulation program as to require a virtual rewrite, but implicit in the service we are to give is the fact that a decision is to be made now, not six months from now. We would be in a better situation if we had a high level simulation language. Let us assume, though, that we have a computer without such a facility. This happens to most of us, and it happened to us.

We knew of course, that simulation languages existed, but not for our computer, and so we determined to devise one, and to write a compiler for our computer. This was a Honeywell 400. Dr. Tocher says it is extremely difficult to find a decent acronymic starting with the initials of his organization. I believe him, but we are more fortunate, and we called our simulation language CAPS, or Courtaulds All Purpose Simulator. It was in fact a development of CSL.

With some further development of the language, Honeywell made themselves responsible for the compiler for the H200 series, and it is now named Extended CSL. The language development was devised by, and both compilers written by Alan Clementson, who describes them in the following paper.

## References

MUSK, F. I. (1966). "One man's meat: Part 1—the uses of adversity", *The Computer Journal*, Vol. 9, p. 1.
CRAIG, T. K. (1966). "CRESTS—Courtaulds Rapid Extract, Sort and Tabulate System", *The Computer Journal*, Vol. 9, p. 3.