Datamation May or June 1968

the forum

THE MEANING OF IMPRECISION

Mr. Pollack has replied in the March issue to my Forum article published in February. He makes a number of points, valid in themselves, which suggest that he has misunderstood my thesis. May I try again?

Firstly, let me clear up one point immediately. Like Mr. Pollack, I am strongly in favour of using decision tables. I don't think that they necessarily impose a rigorous methodology on the user; on the contrary, I believe that they offer one of the few avenues of escape from the rigours of procedural programming. But they must be properly used.

There are many problems which cannot be successfully handled by techniques relying on rigorous, consistent, complete and non-redundant program specifications. Decision tables provide an opportunity for developing alternative techniques for those problems, with a better chance of success. The present danger is that we may be turning our backs on that opportunity.

Secondly, my proposal is not that we should develop new kinds of computer programs. No computer program (none, at least, for any machine available today) can meaningfully or usefully be incomplete or inconsistent: it can't be incomplete because something specific must happen under any set of circumstances—even if what happens is a program-check interrupt; it can't be inconsistent because the machine is fully determined. I am not proposing new kinds of programs: I am proposing new techniques of program creation—new ways of specifying, writing and developing programs.

Thirdly, these techniques would not be of universal validity. There is a large class of applications which are best programmed by rigorous and precise methods; most current business dp systems fall in this class, and I am not suggesting that we should strive to replace order by chaos. But there is another class of applications which cannot be successfully tackled by rigorous and precise methods, and for these we must look for different techniques; my thesis is concerned with these applications and these techniques.

Rigour and precision fail when the job to be programmed cannot be completely specified. There are several possible reasons why complete specification may be unattainable. We cannot specify a practicable chess-playing program of grand-master quality because we don't really understand what the grand masters themselves do. We cannot specify a program for reliable translation of natural languages because the problem is too diffuse: there is no difficulty in specifying the desired translation for any particular sentence, but any set of general rules proves to have exceptions, which themselves have exceptions, and so on. Nearer home, a payroll program may defy complete specification simply because the rules themselves are imperfect: management may have negotiated several agreements with different labor unions; in marginal cases these agreements may be in conflict with each other or with statutory legal requirements. A business information and control system may require development on a tentative "try it and see" basis; management may be unable or unwilling to specify in advance what elements of the system are to be open to change.

In the business dp environment the systems analyst is strongly tempted to adopt a high-handed approach to the computer user. The advent of the computer is seen as a catalyst which brings the chaotic and undisciplined manager into contact with the clear-thinking precision of the computer men, and so provides an opportunity to enforce a rigorous rationalisation of out-of-date methods. Often enough, this approach is justified. But sometimes it is simple arrogance, used as a cover for the inadequacy of the computer methods. No one in his right mind would suggest that the difficulties of machine translation prove that the English language ought to be rationalised; we must be very sure of our ground before we castigate the manager whose business systems defy our present dp techniques.

I would dearly like to describe in detail the techniques I want to see. But I am not announcing an achievement; I am proposing an endeavour. I can only point to one or two crude examples of what seems the right approach.

Mr. Pollack's point about the ELSE-rule in decision tables is well taken; it is an excellent device for allowing the analyst to provide an incomplete specification. And its value is enhanced by the distinction between the "intentional" ELSE-rule inserted by the analyst and the alternative "default" rule inserted by the preprocessor. I am happy to accept these facilities as examples of the techniques I advocate.

I am less happy with Mr. Pollack's strictures on contradictory rules. He writes, "As for contradictory rules, I don't see how computers or people can decide which one of two or more contradictory rules should be followed when a transaction acted on by those rules occurs." Surely there is at least one useful prescription that can be adopted: follow the rule that seems to be more specific; probably it is an exception to the more general rule.

A decision table processor can be constructed (and has been constructed) which incorporates the ability to recognize which of two rules is more specific, and ensures that that rule will be followed in cases of conflict. In use, this processor allows a program to be developed by a series of successive approximations. At the outset, the known (or supposed) rules are processed into a program, and the program is run. It becomes apparent that certain cases are being incorrectly handled, and a rule to cover them is added to the table; the existing rules are left undisturbed. The processor recognises the new rule as an exception to the old, and builds the program so that the exceptional case is correctly handled. The analyst is not required to recognise or to resolve the conflict of rules; he need only state the change in result that is required. By running and adjusting the program several times, the analyst can arrive at the correct algorithm (or a close approximation to the correct algorithm) without ever formulating it explicitly.

This facility is relatively crude; we will need to develop techniques of far greater sophistication. But it is clear that such techniques can be developed and used successfully, and that they could be very powerful. If we cling to the constraints of rigorous methods, we will never solve the difficult problems; we won't be able to get the programs right because we are too frightened of getting them wrong.

—MICHAEL JACKSON