Initial users' reactions: What do they really want?

By H. D. Baecker*

Commercial autocodes or source languages are contrasted with machine assembly codes. Ten factors that affect the overall efficiency of a data-processing system are listed and examined to discover how the use of a commercial autocode will affect them. This paper was presented to the B.C.S. Symposium on *Practical Experience with Commercial Autocodes* held in London on 25 March, 1964.

My own reactions to users' reactions to commercial autocodes are coloured by my experience as a translator writer. Different users require different things from the commercial autocodes that they wish to employ, but unfortunately they are, on the whole, ignorant of the cost of providing the facilities that they demand on the equipment that they are prepared to pay for.

To take a specific example, let us consider two users of the same commercial source language who require it to serve very different purposes. One user has required the translator to generate fast and compact object programs that would have to be run several times daily. The time of translation is relatively immaterial because of the intensity of use of the translated product. This user wished to use a commercial autocode, rather than the machine assembly code, because he is in the throes of establishing an integrated system of accounting and production control, and he values the advantages of a common language for the expression of office procedures that also relates closely to their transfer to machine processing. The discipline of documentation of the source language is also expected to ease the problems arising from staff changes in all departments associated with the system.

The second user looks to a commercial autocode for the speedy solution of *ad hoc* problems of analysis of the company's activities. For this translation time must be minimal, and it must be an easy and quick task to modify source programs. Because a given object program is to be used rarely execution time is of less importance, being overshadowed by translation and program-testing time. Thus efficient facilities for the latter are of great importance. The major saving anticipated is, of course, in the writing of the many diverse programs in the first place.

Both users share a reluctance to invest in a large machine configuration, particularly of input/output units.

Desirable features

Certainly these examples are not unique. We may compose a profile of the desirable features of a commercial autocode. It is impossible, as yet, to describe an ideal standard, one can only contrast different methods of arriving at the same objective, a computer run that will process given inputs to produce the desired outputs.

A great deal of the present enthusiasm for commercial autocodes in the U.K. stems from past experience of extremely rudimentary program coding schemes. Unfortunately there are current-generation computers being marketed here with assembly codes and operating systems that are more primitive, that offer less flexibility and fewer facilities, than the assembly codes and associated loaders that were common in the U.S.A. almost a decade ago. Virtues are therefore being ascribed here to commercial autocodes without a basis for reasoned comparison with adequate assembly systems, because of the paucity of facilities of the assembly codes we have had to suffer.

Nevertheless, let us try to state the important advantages sought by the user:

(*a*) reduce program writing time;
(*b*) reduce the skill and experience required of programming staff;
(*c*) reduce the skill and experience required of operating staff;
(*d*) reduce program-testing time;
(*e*) reduce translation time;
(*f*) reduce object time machine use;
(*g*) reduce program-punching time;
(*h*) use a minimum machine configuration;
(*i*) enable translated programs to be altered in source language;
(*j*) produce maximum documentation automatically.

It is immediately clear that whilst some of these points may serve to assess the relative merits of different commercial autocodes, others serve more to contrast autocodes and assembly codes. Furthermore, there are two further facets that enter into any assessment, the characteristics of the translator of the language and the characteristics of the operating system within which translator and object program are embedded. Let us consider the above points in more detail.

(*a*) A commercial autocode should certainly score over any assembly code in the speed of writing programs, but the relative merits of various autocodes will depend greatly on the complexity of the task attempted by the user.
(*b*) The use of an autocode will reduce the demands made on the skill and experience of programmers, in contrast to an assembly code. The relative

* *Computer Analysts and Programmers Limited, 62/63, Queen Street, London, E.C.4.*

flexibility and power of the available programming languages is the feature that most distinguishes between them, and so affects the difficulty of programming. However, no language is going to mitigate the difficulties of getting a quart-sized job onto a pint-sized computer.

(*c*) A translator for a commercial autocode is not necessarily easier to operate than an assembly program; the reverse may well be the case. As for object programs, in the particular circumstances of a given installation good programmers using an assembly code may well devise and implement a set of operating conventions far easier to observe than the generalized facilities built into high-level language systems. But no operating system allows for any complacency on this score, nor for the employment of unskilled or careless operators.

(*d*) It would be a very bad autocode indeed that did not significantly reduce the testing time of a major program, compared with that program written in an assembly code. Still, good programmers in any programming language will do better than indifferent staff using the same language.

(*e*) A commercial autocode will prove very expensive in translation time. One hopes that ease of testing will reduce the number of translations required in all, compared with assemblies, and so the total machine time used will be decreased.

(*f*) Similarly, the object program resulting from a source program in a commercial autocode will not be faster than that program written in assembly code by a competent programmer. It will usually be appreciably slower. This is sad, but arises from the generality of the facilities incorporated in autocodes; translators cannot yet be structured to take advantage of local restrictions to increase efficiency in the way a program tailored by hand can. Often users are to blame for some of this inefficiency because they look with disfavour on a source language that requires explicit declarations of restrictions; if the latter are incorporated then translation could be far more efficient.

Note that both of the above points (*e*) and (*f*) are as much connected with the translator used as with the autocode language to be translated. Thus two different translators for the same source language on the same machine may give very different results.

(*g*) Reduction in punching time will depend not on the autocode but on the operating system within which it functions. Either a commercial autocode or an assembly code may have or fail to have facilities for machine amendment of chosen portions of the source text, and it is this facility which will provide savings.

(*h*) The more facilities and flexibility that are provided in the translator for a commercial autocode the more equipment it is likely to use during operation. In particular it will use more magnetic-tape working files, or equivalent bulk storage, than the assembly code on the same machine, and the faster the process of translation, or the greater the attempt to optimize the resulting object program, the more hardware the translator will demand.

(*i*) For large programs the facility to alter translated programs in source language is very important. Again, this facility is not a property of the source language but of the translator. However, to achieve this end the translator must preserve all sorts of tables and general parameters that arise during the course of translation, if the amendments are to be translated correctly and related properly to the existing part of the program. This facility will therefore sharply increase the equipment required to run the translator.

(*j*) The programmer in a commercial autocode must define his data structures, as well as his procedures, explicitly. Thus an autocode will provide a clearer statement of the proposed operations, in a "problem orientated" manner, than an assembly code, and so will provide some documentation automatically.

Software economics

Finally, let me draw attention to a very obvious but oft forgotten fact. Software is not supplied free by a manufacturer. He must recoup the cost of developing commercial autocodes and other programming aids from the sale of hardware. But because software is an implicit charge it is difficult for the customer to assess whether he is receiving value for money. He can only assess the total system of hardware and software.

If users feel dissatisfied with the software offered to them, then there would seem to be grounds for requiring manufacturers to distinguish more clearly between the costs of the hardware and software that they supply. A further step might then be an option to buy hardware with only limited software, leaving it to an association of users to specify and implement their own software. This has often been done with universities and similar establishments. Unfortunately the result would hardly be an integrated operating system, but the individual translator writer can hardly fail to welcome the prospect of users becoming intimately aware of the costs and difficulties of providing the programming aids for which they clamour.