Computers in air traffic control

THE TOPIC OF COMPUTERS in air traffic control is receiving a good deal of attention in the United Kingdom press, and it is useful to examine some of the issues involved. Computers for the control of air-defence traffic have been in successful use for upwards of ten years, but the application to civil air traffic control has been more recent and from many points of view more difficult. The task of the air traffic controller is to maintain the maximum flow of aircraft within his area of control while ensuring absolute air safety. This is an intensely personal activity requiring skill, experience and the ability to take swift and accurate decisions.

Any updating and automation of ATC hardware can never be a substitute for the judgment and aptitude of the individual controller. All one can hope to do is to alleviate the drudgery of his task by removing certain of the mundane processes he formerly carried out, and by providing him with more accurate, more rapidly updated, more readily assimilated and more easily available facts. If the controller is relegated to the position of monitoring a machine, the concensus of opinion is that his skills and integrity will be lost, and his ability to over-ride the machine and react to an emergency will rapidly decline.

The introduction of modern high-speed on-line computers to air traffic control has met snags in virtually every instance, generally involving a longer introductory period, increasing programming effort and extension to computer storage requirements. American experience has been quite startling in this respect.

It is difficult to assess the root cause of these problems, particularly since they have been substantially more severe than in other areas, such as simulation, air defence (as mentioned above), industrial control, etc. It may well stem from the difficulty of true communication between the operational user and the data-processing expert. Until a common language is achieved, progress must inevitably be slow. It is now probably true to say that such exchanges are taking place and that the operational air traffic controller can express his needs and requirements in terms which the system analyst can appreciate (and hence translate into software specifications).

Similarly, the analyst can put forward to the controller in mutually comprehensible terms the full potential of a data-processing system. This state of affairs certainly did not exist five years ago, and this "communication gap" was probably responsible for many of the difficulties encountered in the early and mid-1960s.

Another contributory factor has probably been the rapid evolution of hardware, which has overtaken the ability to incorporate new techniques into systems or fully to exploit new developments.

In any system of air traffic control, whatever the combination and relationship of equipment and human operator, air safety must depend on the reliability and integrity of the whole; in devising and developing a system, this must be a prime consideration. In the data-processing role there are two types of reliability to be assessed—hardware and software. The pattern of hardware reliability is well known and is now reasonably predictable. After a "burn-in" period, a piece of equipment will settle down to a fairly regular pattern of reliability, and under consistent environmental conditions the designed mean time between failures (MTBF) will be achieved. Thus, with the knowledge of the MTBF of individual elements, techniques exist for combining these elements in a total system of virtually any desired reliability.

On the other hand, software reliability is rather different. The "debugging" processes which take place during commissioning progressively remove software faults until the system becomes almost fault-free, and even the most obscure faults should be eliminated during the early stages of operation. The great difference compared with the hardware case is that once a software fault is cured it stays cured (or it should). Thus hardware reliability remains constant, and can be set at a planned level through the life of the system, whereas software reliability improves steadily through the commissioning period. In most cases software faults become insignificant early in the life of the system, even after taking into account the software changes and updates that are necessary from time to time.

The key to success lies in the software, and there are several points to be emphasised here. The most important is that the task must be specified from the outset in a way which (a) guarantees to the operational expert that the system will meet his requirements adequately, and (b) also provides the system analyst with all the detail he requires to construct the many flow diagrams he will need and from which the individual programming packages can be derived. This, of course, involves the full operating functions to be performed and their inter-relation. Equally important, and frequently glossed over, are the interfaces between other related systems and peripherals. If a start is made before all these factors have been thrashed out, the result will almost inevitably be prolonged time-scales and a less-than-fully satisfied user.

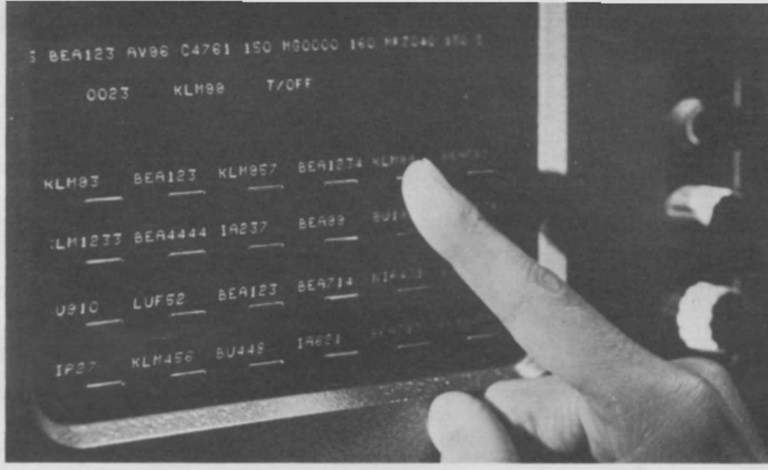
The quality of software is also most important. A skilled analyst and team of programmers will ensure efficient software packages and, above all, the economic use of computer storage. Failure to control the quality of software leads to a proliferation of programming effort and computer usage. In certain American cases, for example, results have been achieved by the "sledgehammer approach," by wheeling up enough number-crunching power to achieve the result, but using substantially more resources than would be needed with a more efficient approach.

The first data-processing task to be undertaken on the United Kingdom air traffic control system for London, code named Mediator, was the first stage of the Flight Plan Processing System (FPPS). Later phases have been contemplated but have not reached the stage of contract action. The object of the FPPS was to rationalise the handling of flight plans, the vital data on each aircraft around which the whole practice of air traffic control is built. Fundamentally FPPS was concerned with five main activities: the acceptance, verification, distribution and display of all relevant flight plans and the computation and extraction of data from them. It in no way replaces the skill and discretion of the controller, and indeed its function lies in easing his task so that his efforts may be more fully concentrated on the essential work of making the most efficient use of the airspace while maintaining absolute air safety. This it does by giving him accurate, rapid, up-to-date data in their most readily available and usable format.

Recently there have been many published inaccuracies and misrepresentations on what has been the intention of the introduction of computers to Mediator. Many of the functions which will ultimately be a part of Mediator, such as co-ordination of radar data with flight plan data and the search of available data for potential conflict, (i.e., the computers' direct participation in the avoidance of collision), are not, and have never been, intended to be a part of the stages which have currently been implemented; nor are they a part of the current phase of the American FAA system. Both systems have developed in parallel, the US starting in 1962 and Britain in 1966, and regrettably—in common with so many software tasks which were breaking new ground in the evolution of data-processing application in the 1960s—both have slipped in time-scale, the US by some six years and Britain by about three.

The slippage derives from much the same cause. This was the difficulty of estimating the scope of an inadequately defined task, not because the tasks were ill-defined by default, but because in the early 1960s the experience of drafting definitive software tasks for very large real-time systems was limited. In Britain it was almost a three-year "round-table" job between the contractor, the user and his technical advisers to reach a full definition.

In the United States the problem was basically similar, but there were even greater and more prolonged difficul-


BEA123 AV96 C4761 150 H60000 160
0023 KLM98 T/OFF
KLM93 BEA123 KLM957 BEA1274
LM1233 BEA4444 IA237 BEA99
U910 LUF52 BEA123 BEA714
IP27 KLM456 BU448 IA621

An important peripheral to an air traffic computer is the method of calling up data. Here an operator is using a Marconi touchwire to specify a particular flight

ties in establishing a dialogue between the user and the contractor. Furthermore it appears that there was also a long history of changes and "improvements" which made the stabilisation of the specification almost impossible. Both systems also met with computer storage problems: in America the contractor was instructed to revert to a larger machine, while in Britain the contractor offered a larger machine but the user preferred instead to restrict the facilities within the scope of the smaller unit.

The philosophy on reliability and integrity varied somewhat on the two sides of the Atlantic, basically because the FAA started out with the intention of 16hr-a-day operation, whereas Britain planned on the basis of 24hr-a-day. Thus the American machine was developed with internal redundancy on a "main/standby" basis, possibly with the thought that software unreliability, due to regular changes and updating, would be more significant. The British philosophy was to use a triplicated system with a "consensus" type of output, leading to a downtime, subsequently substantiated by measurement in the actual environment, of less than 30sec in five years. This philosophy assumes that with skilful programming and "de-bugging" the software reliability can be worked up to a high level. This is feasible, as explained earlier, since most software errors are cured once and for all, and the fault level will steadily fall through the commissioning and early life of the system to a level where it no longer has significance.

The decision to buy the FAA system for the next stage of Mediator has been announced; while it is difficult to take a dispassionate view of this decision and the events leading up to it, one can only express the belief that it derives from a different approach in America and Britain to a very similar set of problems presented to the user authorities in about the same time frame. In both cases the system "ran out of storage"; in the US the contractor's offer of a bigger machine was taken up, whereas in Britain the contractor's proposals for a larger unit were discussed and debated for some 16 months, by which time the possible time scale for implementation was substantially eroded. In the meantime the Americans had steam-rollered the problem with the bigger machine, being prepared to accept the inefficient use of storage which such an approach to programming inevitably produces, and had offered the resulting system to Britain with the now well publicised result.

The American supply will be restricted to the machine and associated software for the current operations. The original British contractor is likely to be responsible for the main peripheral equipment, including the data-display system and the various interfacing tasks, and will also play a significant part in the management of the system and software. J.S.

FLIGHT JULY '72