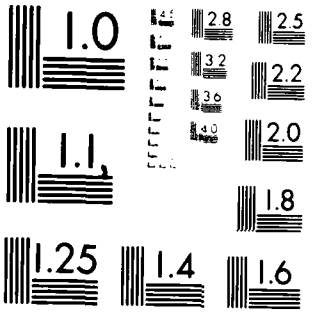
AD-A132 030 AVIONICS SOFTWARE: WHERE ARE WE?(U) RAND CORP SANTA 1/1
MONICA CA W H WARE SEP 82 RAND/P-6786 SBI-AD-E750 779

UNCLASSIFIED F/G 9/2 NL

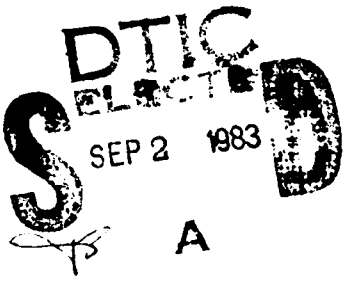
END
DATE FILMED
9 83
DTIC



AVIONICS SOFTWARE: WHERE ARE WE?

Willis H. Ware

September 1982



P-6786

The Rand Corporation
Santa Monica, California 90406

AVIONICS SOFTWARE: WHERE ARE WE?*

Abstract

Since the digital computer first flew in an avionics system 25 years ago, the art has progressed from small very slow vacuum tube machines with limited memory to fast chip-based machines that not only do sensor processing but also integrate a variety of data sources into many capabilities--among others, navigation, sophisticated weapons delivery, programmed menu-displays to the air crew. As onboard computer hardware has proliferated, software inescapably has also. From a few hundreds of program words at the beginning, flight software is commonly many tens-of-thousands of words; frequently, a few hundred thousands; and in some cases, even a million. Thus, implementation and management of software resources has become a major problem area for military services. The paper explores dimensions of the issue as it now exists, suggests many positive actions under way, and proposes a direction in which the future may well move. It concludes that software will continue to be troublesome; progress will come slowly.

Before asking where are we in avionics software, we first ought to ask where have we been; then we can ask where we are going. Among the earliest, if not the first, digital computers to fly was one in the MA-1 fire control system. It was developed in the early 1950s for the F-102 fighter to control the Falcon missile and folding fin rockets. A vacuum tube machine using subminiature tubes, it operated only at hundreds of operations per second. It could not accommodate the full dynamic behavior of the fire control and launch calculations; primarily it supplied constants to the analog computation of the fire control equations. Its operational program, measured in hundreds of words, was contained in

---

*Presentation at AGARD AVP Symposium on "Software for Avionics" at The Hague, Netherlands, September 6-10, 1982.

a small magnetic drum on which individual instructions were spaced circumferentially to accommodate the latency time of memory access. Parenthetically one might wonder whether anyone today would still know how to do minimum latency programming. The program was done in a machine language, and life-cycle support of software had not yet emerged as an ongoing operational issue for military services. Interestingly, MA-1 still flies in a solid-state version in the F-106; its operational life has exceeded 25 years.

By contrast, the Air Force F-16 fighter is among the most highly automated of deployed operational aircraft. Instead of vacuum tubes, F-16 avionics uses modern microelectronics; its fire control computer has 32,000 words of memory and runs at some half million operations per second. While 128,000 words of computer program fly in every operational F-16, of this only about 30,000 words are written in the high-order language JOVIAL J-3B-2. The rest are in the machine language of whatever computer happens to be involved, e.g., the microprocessors in the stores management subsystem, the signal processor in the radar, the microprocessor in the heads-up display. As a second example, the U.S. Navy F-18 flies 400,000 words of program which executes in some 13 machines; about 2 percent of it is in an HOL. The LAMPS helicopter also has some 400,000 words of program on board; of it about 30 percent is in an HOL. Turning to larger aircraft, the U.S. Navy P-3 land-based airborne early-warning vehicle has 700,000 words on board, 128,000 words of core memory, but also over one million words of diagnostic and maintenance programs. Originally done in assembly language, it is now approximately 70 percent in HOL after a major upgrade. The analogous Air

Force AWACS has 512,000 words of memory. Some modern-day military aircraft have a centralized computing function with point-to-point wiring for signal paths, whereas others--notably recent tactical fighters or upgrades--use data bus architectures with some measure at least of distributed processing. Some onboard systems have a capability for degraded modes of operations as equipment fails, but others have virtually no fallback capability.

Indeed, we have come a very long way in terms of the amount of software that flies with modern-day aircraft, a long way in terms of the speed, size, and power attributes of electronics, and a long way in the level of automation. Obviously, though, the penetration of HOL languages is neither as uniform nor as deep as is often believed. It varies from a few tens of percent in contemporary fighters to many tens of percent in the larger vehicles, especially those that have been upgraded. The MA-1 system is where we came from; the F-16, F-18, P-3, and AWACS are representative of where we are.

To explore where we are, let us note some dimensions of modern avionics software. First of all, what is "the total software job" of a modern aircraft? The first and most obvious component is that which flies onboard--software in a radar, in the heads-up display, in the stores management system, in the inertial navigation computer, the air data computer, possibly a fuel management subsystem, perhaps a separate display and controls management subsystem, or perhaps an entirely separate system (as in the B-1 bomber) to monitor all else for malfunctions and to capture maintenance data. In the most complex of aircraft, all such systems will be networked together by a bus arrangement often

presided over by a central computer complex. Just as we have not progressed all that far in the use of HOLs, neither does the equipment always exhibit outstanding field reliability. It might be, however, if one were to consider reliability relative to the increasing complexity of systems, perhaps progress would be found better than normally perceived. Be that as it may, onboard equipment does malfunction and so modern aircraft have an extensive array of ground support equipment, which for the most part is software controlled.

Among the ground-based diagnostic and repair facilities is one often called the Avionics Intermediate Shop, generally a highly automated complex of test stands that can examine equipments which either have or are presumed to have failed. Supporting the AIS level of maintenance is a depot or rear echelon capability that generally deals with problems at the electronic card level rather than at overall equipment level. Here one finds a wide variety of automated test equipments, most of which are software driven and controlled.

Just as equipment must be maintained in operational status, so aircrews have to be properly trained. For that purpose crew training devices of many kinds are utilized for modern aircraft. Perhaps the most widely known example of this technology is not an airborne one but rather the crew training simulators used by NASA for manned space missions.

All the software implied by the prior discussion had to have been developed somehow, so one commonly finds one or more program development environments for the working programmers which includes all the diagnostics, testing tools, recordkeeping tools, languages, compilers, etc.

used in modern software development and its management. Contrary to the common perception of 25 years ago that operational software would never change, it is now widely understood that it does and will continue to change for a variety of legitimate reasons. Therefore, a modern deployed highly computerized fleet of aircraft must be supported by a facility for the life-cycle support of software. Among other things, the latter includes a software development environment for each of the computers whose software is to be maintained and for each of the languages that are flying and in ground support.

However, it must also include a variety of test and simulation tools to assure that software changes have been properly made and will not lead to new anomalies of behavior. In addition, there usually also must be appropriate flight vehicles, perhaps especially instrumented, to make certain that all is well with the updated software before it is dispatched to the field. Sometimes, especially when the computer involved is a microprocessor, any software involved will be regarded as simply another component of the equipment, which will be tested end-to-end for functional performance without special explicit tests of the software. This view has been taken, for example, in new commercial aircraft; and hence, any life-cycle software support in such instances is seen as an obligation of the vendor supplying the basic equipment.

Thus, when one speaks of "the software job" for a modern aircraft, it proves to be an enormously large undertaking. Moreover, it implies ongoing attention to change just as does physical modification of aircraft. At minimum, the software job for a modern combat aircraft is a few hundred thousand lines of code; but if all of the software develop-

ment tools must be built as well, then it can be many hundreds of thousands. For large sensor platforms with extensive diagnostics the corresponding number can exceed a million. The many components of the total software job share a unique characteristic however; they must all be kept in lock step. The ground-based test equipment must examine equipment boxes as they are, not as they were a year or more earlier. The crew trainers must mimic the aircraft as it is, not as it was. Life-cycle support must match equipment as it is, or in some circumstances as it will be. Finally, we may have to do configuration control of the software by production block numbers, and in some circumstances even by the tail number of an aircraft.

Thus, in addition to all the usual management difficulties associated with large and diverse software undertakings, there is now a time synchrony aspect that can be difficult to accommodate. It is especially so, given that the several components of software are, in the United States Air Force at least, handled by different groups of people at different places, in different organizations--some military and some contractor--and often with different planning and funding arrangements. In one sense it can be observed that the wide exploitation of computer technology in modern aircraft has led us into a morass of difficult organizational and technical issues; but it is part of the price to pay for military air power capability.

Where do we stand? Computing technology continues to be very dynamic. Software is a very manpower-intensive undertaking that is difficult to manage and frought with danger. Seemingly, the number of successes as measured by budget and schedule are far outweighed by the

number of failures. Software generally is entirely too unpredictable at the outset in regard to its eventual cost, the date of its eventual completion, and its ability to realize the user requirements as he really wants them to be, versus what he perceived them to be at the outset of the program. Finally, there is always the question of the resources needed for ongoing life-cycle support.

What is being done about this array of problems? Special languages have been developed and are gradually coming into use to support the initial requirements analysis phase, and to assist in tracking the translation of such requirements into corresponding software capability. Some automated design techniques are slowly appearing, and there is discussion of, but not too much success in, reusable portable software. New languages--such as ADA--are being completed and will be introduced into military software programs. Standardization is finally achieving some level of success; at least in the Department of Defense there now exists military standards for data buses, for instruction set architectures, and for acceptable HOLs.

Some of these techniques have just moved from the research laboratory into the development world, and hence their effect has yet to be felt. Clearly, some installations and some contractors are in the lead; but even in the best places software disasters continue to occur. Beyond that though, there is still a major part of the software community supporting the Department of Defense that has yet to be introduced to the most contemporary tools, techniques, and management approaches, and to be trained in their effective use. Some things thought originally to have been truths are yet unvalidated and may prove to be

mythology. It remains to be seen, for example, whether extensive use of an HOL will result in cost savings either in the initial development or in the ongoing life-cycle support. Limited evidence suggests that it will not happen. Similarly, the arguments for standardization suggest significant cost savings in terms of logistic support, replication of software development support tools and environment, training of people, etc; but in this case also, the penetration of standardization is presently minimal and its payoffs are yet to be realized and measured.

Hence, there are advances under way that promise future benefits for accommodating the difficulties that have appeared through wide exploitation of computing technology in modern aircraft. There are other things coming along, however, whose consequence is harder to judge. Ahead, for example, is a new VHSIC era in semiconductor technology. Might it make possible capturing specific functional capabilities on a chip, thereby eliminating at least some part of a software job? Yet to be felt are the fullest impacts of the microprocessor advances. As they grow smaller and more capable, will we see an increasing array of equipments that are microprocessor based, may contain extensive software, but are treated, tested and maintained as functional equipment without regard to software content? Is there some possibility, for example, that some part of the software job can be passed off to the suppliers of equipment? Clearly, there has been progress in dealing with software; there are new ideas and advances in train. What though might the future look like? What conjectures can be made about it?

Admittedly, it is a judgment call, but in my view the odds are that every problem we now have with software we will continue to have, and

perhaps more. Why? The automation levels on newer aircraft are bound to increase as the task environment in which the aircrew must function becomes ever more complex. Even today's pilots commonly state that the job in the cockpit is far beyond what can be accommodated by an individual. Many, for example, say that of an aircraft's total capabilities, a given pilot is familiar with and exploits perhaps only a third of them. Probably, however, each pilot specializes himself to a different third.

One can project, I think, that avionics systems will get smarter in the sense that their behavior will be perceived by users as having some level of intelligence. The evolving technology of expert systems and knowledge-engineered systems is bound to find its way into aircraft, and as it does the "IQ", so to speak, of the avionics system will gradually increase. There are some very profound technical consequences of projections such as these not only for implementing better systems but also for maintaining them. If we are successful in building smarter avionics systems, will we be equally successful in assuring that they can be maintained in operational status and repaired in the field by military manpower?

The computer programs to provide higher levels of automation and expert systems will be enormously more complex than even the worst of today. The programming job will be more difficult to do and will use more sophisticated techniques; thus one can imagine that the management task of producing such software, and testing it to assure that it is adequately error-free, will also increase in difficulty. If smart and highly automated systems are to be accepted by their users, then they

must be available when needed and they must perform as expected when needed. Thus we have ahead of us the task of handling much better the whole business of malfunctions, anomalous behavior, fault detection, maintenance and repair.

It is commonly acknowledged throughout the industry that there now is a serious shortage of properly trained personnel--both for creating computer programs as well as for managing the process--and the situation is not likely to improve. Computer technology is still diffusing so rapidly through the world at large that the demand for such talent is high everywhere. Hence, the commercial organizations that support military systems will have to compete with a rapidly expanding commercial world for what is already in short supply.

Everything clearly suggests that aircraft will continue to get ever more expensive; therefore looked at as a resource to be maximally exploited for military advantage, a faster sortie turnaround rat will be essential. This point reflects itself partly in the issue of managing faults and error problems, but it also reflects itself in the ground maintenance aspect--notably rapid testability and identification of trouble, efficient means for removing-and-replacing, and fast checkout of an aircraft on its way to the next sortie.

Look now at technology push and technology suction. Whether one country likes it or not, potential opponents will make technological advances that result in new capabilities and opportunities for military action. To counter such ever-increasing threat, at least equal progress will be needed in order to remain superior. Thus, the military establishment will always require the best of technology to accommodate an

enlarging threat. The technologist himself, as he perceives the need of the military services, will encourage use of his capability. Thus, military systems will always be on the forefront of technology.

For most of the parameters important to flight performance and the technologies behind them, there simply is not much room to grow. Progress in such things as propulsion, lift-to-drag ratio, or thrust-to-weight ratio will be measured in a few tens of percent at the best. In contrast, growth in raw computing power still promises at least a factor of 10--and perhaps even 100--as we move into new more elegant semiconductor and switching technology. If--and it's a big if--we can build software whose growth in capability approximates that of computing hardware, then we may be able to vigorously exploit a resource not commonly perceived as part of an aircraft, namely data available to it from sensors and information that can be derived from such data. It is conceivable to my mind, that while the usual vehicle performance parameters may grow only slowly, large increases in military capability may nonetheless be achieved through a highly automated, wholly integrated information infrastructure to manage the vehicle and support its crew.

As we better understand decisions and actions that a pilot makes, complex information processing systems can be designed to do things that were formerly the prerogative solely of the human mind. With new progress in electronic technology and therefore in computer hardware, we will be able to architect systems whose individual components are not only highly reliable but whose system availability is even higher. This will not minimize the whole issue of health status monitoring, fault diagnosis, and repair; but it probably will affect how maintenance is

done, the magnitude of the logistics tail, and the geographical location of repair facilities. Military aircraft will continue to push the forefront of information technology; and because skilled personnel will continue to be in short supply, and because we will be building ever more elegant systems, things in the software world are not likely to get much better in terms of the overall job, in spite of all of the positive efforts now under way.

Life-cycle support is not likely to change either, although its details may somewhat. Historically, what was once called maintenance of software was seen as a nuisance to be minimized. It is now understood that the changeability of software--as awkward as that may be or as demanding of resources as it may be--is still the easiest way to accommodate inevitable change in the military threat and the unavoidable changes demanded by operational users. Whatever the problems of modifying software might be, it is still easier and less expensive than modifying physical equipment. Among other things, replication of software is automated, error-free, and inexpensive. It can be shipped to the field for installation rather than bringing every serial number of some equipment back to a depot for modification.

While computer hardware apparently will be no problem in a real sense because of progress in the semiconductor industry and such specialized efforts as VHSIC, sensors and instrumentation may be a problem; they have been in the past in other areas of automation. We will need to know, for example, what the aircrew is up to, which way they are looking, what they are intending to do, what they are planning to do in the next minutes. How will the crew communicate its intentions to

automated systems? I can imagine that the development of appropriate sensors to make measurements not only on the world but also on the crew and the aircraft--and they will obviously be computer-based ones--might be a pacing item in moving up the level of automation.

If this is a valid projection for the future, then one must focus immediately on the software issue of the future. Clearly, airborne computer hardware is progressing extremely rapidly, and one can stipulate that it is unlikely to be the pacing item. On the other hand, we will continue to have all the problem dimensions now associated with and understood about software. We will certainly encounter new ones, some of which have been suggested. We may have to build software intensive systems that function on various aircraft, systems that are cross-service, or even multinational systems. One way or another though, we who are in the computer hardware and software world of military aircraft can look forward to a continuing growth of onboard computery, and a continuing ever-widening exploitation of information. After all, information is the universal commodity that keeps all complex systems functioning, and to make better use of it, we have only digital technology.

The military user will state his operational requirements as best he can perceive them, but software intensive systems are so complex in their eventual behavior that the user often cannot understand what he wants until he has first seen and has used the end product. For relatively simple software-based systems, this issue may be inconsequential or even absent. For the most involved ones, however, we will probably have to accept a fairly intense modification, update, and support activity in the early part of operational life as the user really comes

to understand what the system can do or might do. This is quite aside, of course, from whatever design flaws or anomalies of behavior remain in the delivered software for whatever reason. The odds are that life-cycle support of software will never stop throughout the entire life of an air vehicle or its systems. Moreover it has become clear that everything is likely to have a much longer operational lifetime.

The real pacer of an information intensive avionics future is almost certainly to be the frontend intellectual understanding of just what functions the user wants, and how they become software. We will need research emphasis for many years to understand how to develop and manage not only user requirements, but also their translation into software requirements. In this regard, one must note that the precision of dialogue demanded by the software design process is rarely matched by the precision with which user requirements can be stated and transferred to software designers.

While we should not predict the future with gloom and doom, nonetheless I think we must realistically acknowledge that software will continue to be troublesome. Moreover, it will almost certainly continue to be seen as the culprit for a whole variety of ills. I do think that changeability of software, and hence its life-cycle support demand, will increasingly be acknowledged as a positive feature rather than an annoying nuisance that cannot be avoided. Thus, some of the past negative attitudes toward software in this regard will gradually erode. While the software world has devoted itself in the decade of the 1970s to understanding and developing specialized management techniques for implementing it, the problem in software is often an incomplete intel-

lectual understanding at the beginning of what it is really supposed to do.

Of course the computer software scene will improve somewhat; there are a lot of positive actions under way. We will improve our mechanics of organizing and managing software projects. Occasionally some software project will even go smoothly because it will represent a task that is conceptually simple; or a task that is fully understood intellectually; or a task that has already been done before, and therefore a kind of software prototype exists. In particular, the intellectual prototype of the job to be done will be understood. To be sure, HOLs will improve and they will help; but system developers are moving slowly into them. New HOLs such as ADA will undeniably be very useful in providing a structured way in which to describe software requirements. Moreover, such languages will provide an unambiguous way to communicate among people that are involved in the user requirements, in system requirements, and in software requirements phases.

Let us acknowledge all the positive steps now under way to improve the whole software situation in the broad. Let us take credit for all the improvements that will happen in and from HOLs, improvement in management techniques, improvements in descriptive languages, and all else. No matter how good we get, software in my view will continue to be troublesome and progress will come slowly. There will be the ever-increasing demand for military capability to counter new threats; software that must be built will ever-increasingly try to implement complex intellectual information processes; the unavoidable intellectual understanding of the frontend requirements process will continue to be a

pacer. It is not a dreaded future though; it is an exciting one as we in the avionics business learn how to create a fully integrated information infrastructure for military aircraft. Unlike the relatively stable future of many technologies involved with the air vehicle itself, we in the avionics business have several decades of dynamic future as we learn how to exploit information as a resource to achieve large advances in military capability.


DATE
ILME