SOFTWARE IN THE EH101 - A CASE STUDY

Dr P J Whittle

Westland Helicopters Limited

Yeovil, Somerset, UK

Abstract

Software is used in modern avionic equipment for a variety of reasons; operational capability unobtainable by other means, flexibility (although this can be a disadvantage if it is not properly controlled) and accuracy as compared with analogue devices.

At the time of the original EH101 equipment specification (1982) there were no well established standards for High Integrity software although these did become more widely available as more and more avionic manufacturers used software in their equipment.

A set of standards for the EH101 therefore gradually evolved throughout the project and are still evolving now as we contemplate and plan the RAF Utility variant.

Software is used in a number of control systems on the EH101 with varying levels of system criticality. The basic philosophy with regard to Vendor Management and product certification, however, remains essentially the same.

Introduction

Why is Software used in modern avionic equipments? Within the engineering community software systems have a reputation for being undependable, always late and always over cost. Everyone has heard the stories about the Space Shuttle being delayed because of software faults, or the Venus probe which was lost because of a punctuation error. More recently, the Russian Mars probe failed because of a software error which allowed the ground controller to accidently corrupt the programme. Interestingly it was the ground controller who appeared to get the blame rather than the software development teams who did not detect the error.

In the software community the problem is known to be more widespread. If software is so untrustworthy, then why do not we, as engineers, avoid it and use discreet components and analogue computation?

There are three main reasons that are pertinent to avionic systems:

1. Software allows more operational capability to be given to the operator. Inherent power and flexibility allows a wide variety of complex information to be presented to the operator in all sorts of different ways, tailored to be useful for any particular application. All of this can be achieved using less space and power than equivalent non computerised systems.

2. Logic implemented in software is, in theory at least, easier to change than logic implemented in hardware. This gives great flexibility and easy upgrade paths for the operator. This reconfigurability is a particularly important attribute for mission control systems where the operational requirement is likely to evolve substantially during the lifetime of the equipment. However, it should be realised that this relative ease of changing a program, at least the mechanical part of editing, recompiling and reblowing an EPROM can give rise to other difficulties. These can only be controlled by utilising

rigorous configuration and change control procedures.

3. Software technology allows more logic to be built into the system. Calculations can be performed to much higher levels of accuracy compared with hard wired systems. The software can also perform more checking of data inputs and outputs and of the hardware elements themselves. Thus overall reliability is improved. If a dual redundant architecture is used then a failed device can be disabled and a backup switched in to take over.

The EH101 uses all of these useful features of software systems.

EH101 Software Project Management

At the outset of the EH101 project it was recognised by all the parties concerned, WHL, Agusta and the British and Italian Government procurement agencies that a large amount of software would be required to be written for the EH101 avionic system, a lot of it from scratch. In order to devise an effective strategy for the management of this development it is important to understand the similarities and differences between hardware and software controllers.

Engineers have long known how to specify a hardware control system by means of a black box model. In such models the inputs and outputs can be described explicity as mathematical functions. In theory, software implemented controllers can be described in exactly the same way. However, there are fundamental differences between software and other approaches which demand particular attention.

Complexity

Software systems are, typically, much more complex. Precise documentation, even for a small system, can fill a bookcase. Similarly it can take a very long time for a programmer to become sufficiently familiar with a particular system before he/she can be trusted to make improvements on his/her own.

Error Sensitivity

Software is very sensitive to small errors. This is largely not the case in analogue hardware systems where the concept of tolerance is meaningful. However, no useful interpretation of tolerance is known for software. As several well known examples prove, a single punctuation error can be disastrous. Perversely, however, quite large errors can equally have negligible effects.

Hard To Test

Software, by its very nature is notoriously difficult to test. The number of different permutations of possible input data quickly becomes astronomic for all but the most trivial program. Testing by interpolation where one assumes that devices which work at two close points will also work at all points in between, although valid for analogue devices is not valid for software.

For these various reasons it was recognised that rigorous control of the EH101 software development would be vital to the successful completion of the overall project. This was particularly important when the early estimates for software indicated a requirement for approximately 500,000 lines of code to be written for airborne software alone. As well as this two significant other software development activities were required, Integration rigs (involving for the main part software modelling and emulation of sensors) and Simulation (involving the prototyping and testing of Man Machine Interfaces).

It was decided to mandate very precise software standards and also specific software tools and languages. These were to be applied primarily to airborne avionic equipment but were also applied to the integration rigs as these have similar integrity requirements.

Constraints in the choice of simulator hardware and the fact that it does not have the same integrity requirements as airborne or rig equipment meant that different, but nevertheless similarly rigorous quality standards were more appropriate.

Standards Applied

At the time of the original specification (1982) there were no well established international standards in common use in Europe (such as DOD-STD-2167 and RTCA/DO-178A). Therefore a detailed software standards document was produced which mandated rigorous development methodologies and deliverable documentation from suppliers. Westland itself does not manufacture avionic equipment, and so another important aspect of the overall project was the management of the software development activities of equipment vendors, and auditing them against the detailed standard. As other standards became more widespread in their application and acceptance, in particular DO-178A, it was comforting to note that other experts in this field saw similar aspects of the software development life cycle as being of equal importance. In the original specification for the software standards to be applied to the airborne equipment it was decided to mandate a programming language (Pascal) and a development tool set (Perspective). This was felt to be advantageous for a number of reasons, in particular as future software support arrangements could have involved an independent third party. However, experience showed that this presented unforeseen complications which needed to be addressed. In particular some avionic equipments were derived from existing equipments which had already been developed using a different language. Clearly it was not justified on cost terms to rewrite this software in Perspective Pascal as well as the fact that this would actually be likely to increase the overall programme risk. Each of these cases had to be considered on its own merits.

Results (Good)

What was the result of all this risk reduction and management? well, the first avionic aircraft, PP4, has its first flight with all its avionic equipment fully functional.

The first RN aircraft had fully functional Radar, ESM, and Sonics Systems all controlled by and displayed through the Mission Computer Unit. The capability of the basic flight systems is best illustrated by the fact that this aircraft was landed in the dark on its first flight using these systems.

Results (Not Quite So Good)

The EH101 Software Requirements Document referred to above, even though it covered most of the software developement lifecycle did subsequently prove to have a few weaknesses. In particular it didn't mandate exact documentation formats such as is done in DOD-STD-2167. We subsequently discovered that different suppliers interpreted our requirements in different ways. This added significantly to the work of the Westland review teams. It was also difficult to get suppliers to introduce a metric reporting scheme, which we believed was highly desirable. However, because it was not in our contractual requirments it could not be enforced. Some of the equipments had other developement difficulties but the establishment of the Vendor Managment Group was vindicated by the fact that these potential problems were identified early enough to enable adequate recovery plans to be negotiated and implemented in time to still meet critical aircraft flight dates.

Lessons Learnt

History has shown that the approach adopted by Westland - that is having a small Vendor management team whose job is to attend design reviews, review documentation and perform audits, has proved extremely effective. Also justified is the statement at the beginning of the project as to the

software standards to be employed. This proved to be an essental part of the contractual conditions placed on suppliers. In retrospect, however, it was not strong enough in certain areas.

Early concerns about configuration and change control did not materialise. Westland employs rigorous configuration control on site and insists on it being enforced at vendors sites.

Overall these procedures proved effective. From the vendor management group point of view the most powerful tool in their arsenal proved to be the audit, both scheduled and unscheduled. It proved to be by for the most effective way of really understanding a suppliers software development programme, what the problems were, and how they were being addressed.

The Future

Software will be used more and more in Avionic Systems. We already have a fully civil certificated fixed wing aircraft using fly-by-wire (the Airbus A320) which is dependant on execution of software programs for its continued safe flight. Because of the general nervousness of certification authorities, it does, however, have some reversionary capabilities. Military aircraft are currently flying with Active Controls Technology (ACT) where the aircraft is designed to be statically unstable, artifical stability being introduced by the software. If this is extended to civil aircraft and can be proven safe enough, enormous savings in fuel can be made.

Westland has, for some years now, been studing ideas for ACT applications in helicopters. The levels of integrity required are not hugely different to those applied to systems on the EH101. The problem will be one of convincing the regulatory authorities that their concerns about safety are addressed by the best software technology available at the time and that the potential advantages to the operator far outweigh any potential risk. A number of techniques for increasing integrity are being developed such as static code analysis, formal methods of specification (and ultimately mathematical proof of correct code) and the use of dissimilar processing. The true value of these techniques is still being hotly debated.

Finally, would Westland continue to use a project wide software standards document for any future software development? The answer is yes, but it would be much more detailed and would follow more closely the current de facto standards of DOD STD 2167 and RTCA/DO-178A.