

Rapid Systems Integration of Navigation Avionics

Larry Zetzi
Naval Avionics Center
Indianapolis, IN 46219

Andrew Fuller
SofTech, Inc.
Fairborn, OH 45324

Roger Andrews
Merit Technology
Beavercreek, OH 45431

ABSTRACT

The testing of newly developed navigation software for military aircraft typically represents a major portion of the development cost. The navigation system performance must be tested to insure all requirements are met. Since changes cost less in the development stage, it is desirable to perform as much testing as possible in the laboratory. Navigation sensors must be physically moving to provide realistic input data to the navigation computer, making it difficult to test dynamic real-time software. Simulating navigation sensor inputs using general-purpose computers provides a realistic environment in which to perform the tests.

Computer simulations of avionic equipment can be developed and modified more quickly than military standard hardware. This allows system integration to continue while portions of the actual avionics hardware and software are still under development. Discoveries made during this process can benefit the equipment developer so that delivery schedules can be shortened.

A simulation environment provides the capability to put the system in widely varied and even high-risk situations that would be difficult or unsafe to attain in flight test. This can help to determine the fault-tolerant characteristics of the design.

This paper illustrates the techniques and philosophies of using simulations as a tool to help reduce the cost and schedule associated with navigation system integration. The ES-3A navigation integration will be used to illustrate specific implementation considerations and lessons learned. The approach for this project was to reuse generic aircraft flight simulation software developed under previous programs and add avionics modules unique to this platform as needed.

INTRODUCTION

Navigation development is a major cost contributor to a new avionic system integration, whether for a new aircraft or as part of an avionics upgrade program. The navigation function is critical due to safety concerns but it is also critical (and is usually a schedule driver) because it feeds information to other systems, such as antenna steering and situational display generation, that cannot be debugged and integrated until the navigation system works properly.

The task of integrating a navigation sub-system can be approached in several ways. One method is to acquire all the actual gear, install it as a pallet on a large aircraft, and use flight time to dynamically stimulate the devices so that integration problems can be resolved. Another approach is to develop hardware emulators for each box in the sub-system so that some limited functional tests can be performed in the lab. This paper offers a third approach which can further reduce the overall development schedule and program cost.

The approach is to use commercial minicomputers to simulate the communication and functionality for the sensor parts of the navigation sub-system so that the mission computer software and interfaces can be tested and debugged. In current sophisticated avionic designs, this capability is an engineering tool for system-level design and development that is as valuable as an oscilloscope. The tool includes providing realistic, dynamically changing data to the unit(s) under development so that much integration can be done in the lab. One advantage of this approach is that developers do not have to wait for the entire aircraft to be completed to start "flight" testing. Life cycle software cost studies show that the earlier in the development cycle a problem is discovered, the cheaper it is to fix. This approach permits integration testing to

begin early so that critical design trade-offs can be studied before a poor design choice impacts the sub-system. This also reduces the number of actual flight tests needed (and its associated cost of \$1500-4000/hour). Another advantage of this tool over a federated hardware emulation is that multiple sensors can be correlated easily. An avionic box supplier will sometimes provide an emulator for their box, but often there is a requirement to test the full sub-system with all sensors providing coordinated realistic data which is difficult with many stand-alone emulator systems. A third advantage is that the simulation tool is flexible so that when a requirement changes or an equipment upgrade occurs in the future, the tool can be easily modified to integrate the new configuration. This concept can be expanded so that multiple projects can benefit from the engineering tool. As each project adds to the capability of the tool set, other projects with similar design goals can benefit.

Commercial computers are used because the simulation developers can take advantage of the mature operating systems and compilers, advanced software environments, and large user groups to reduce risks (cost and schedule). This also provides the opportunity for the simulation to be ported to other laboratories or flight test centers that may have the same computer resources or that can purchase the readily available hardware.

During the navigation sub-system development, requirements for the simulations must be established. Designing and integrating a system that measures movement in an environment that is not moving (a laboratory) presents a challenge. The simulation environment must provide realistic and coordinated responses to the unit(s) under development. This may include responses on all digital and analog interfaces that are realistic in timing and data content. There are several valid reasons for simulating a device:

- 1) if the actual unit is not available (no spares or unit still under development),
- 2) if getting the actual unit to respond realistically is cost-prohibitive (3-axis full motion table for an inertial assembly), or
- 3) if the actual unit would have to be destroyed in order to perform the test (checking failure modes).

ES-3A IMPLEMENTATION

One specific example of applying this approach to navigation integration was the ES-3A program. The Naval Air Systems Command tasked the Naval Avionics Center (NAC) to perform the system design and integration for a converted S-3 airframe to replace the aging EA-3B. Lockheed

Aeronautical Systems Company was tasked to install the equipment and also had responsibility for equipment retained from the original S-3A. NAC tasked SofTech Inc. to develop ES-3A mission and navigation software. The navigation software was to be based on the S-3A/B Navigation and Steering Computer Program Performance Specification (CPPS), written by Lockheed. SofTech's approach was to use the CPPS as a baseline and implement a modular design that could be easily tested and integrated. The avionics software was required to be written for the Navy's AN/AYK-14 standard airborne computer.

As the integration environment was discussed, it became evident that a dynamic test environment did not exist. The Integrated Test Facility (ITF) at NAC, which was being used to integrate and test the mission hardware and software, was inadequate for testing the navigation and steering software. The ITF had no aircraft models or navigation sensor simulations, and the time required to integrate the necessary hardware and software would have caused schedule overruns. It was not feasible to try to create a dynamic environment by physically moving sensor equipment. This presented a major problem in that the navigation software needed to be fully stress tested prior to delivery to the aircraft. The answer to this problem appeared to be the Digital Avionics System Laboratory (DASL) at NAC.

A tilt-rotor aircraft simulation had been previously developed in DASL to aid in the testing and support of the V-22 software. Although the aerodynamics models would need modifications and several navigation sensor simulations needed to be developed, a solid foundation existed. SofTech and NAC personnel began work to design the most effective and dynamic avionics environment that could be built within the strict schedule and cost constraints.

DASL DEVELOPMENT

Using the existing computer suite in the DASL, the imperative simulations and equipment were identified and a top-level design was formulated. A comprehensive software architecture had to be laid out to define the needed interfaces between the different simulations. A suitable hardware architecture also had to be developed and implemented to support the necessary simulations and models, as well as the cockpit controls and displays.

The existing models had been developed for the JVX program (the precursor to V-22). We identified that the aerodynamics model needed to be modified to more closely reflect the flight

The existing DASL configuration consisted of a suite of commercial computers, cockpit hardware, display hardware, and network hardware (see

figure 1). The environment and aerodynamics models, the avionic simulations, and Heads-Up-Display (HUD) and Out-the-Window (OTW) generation software ran on the Hot-Bench VAX 8550 computer. An Ethernet connection to a Silicon Graphics Iris computer allowed Merit Technology's MAGIKSCENE display processing software to display the HUD symbology over an OTW picture that included the Defense Mapping Agency's Digital Terrain Elevation Data (DTED) and Digital Features Analysis Data. The VAX 11/780 received cockpit inputs and also displayed the OTW and HUD data on an Evans and Sutherland display. The VAX 8550 used Digital Technology's BIU-153 1553B interface card to communicate with the avionics. To allow flexibility in configuring the DASL for multiple projects, a Design Analysis Associates Avionic Laboratory Interface Device (ALID) shared memory network hardware was used to allow all VAX computers to access the cockpit stick position and switch data and OTW and HUD display information.

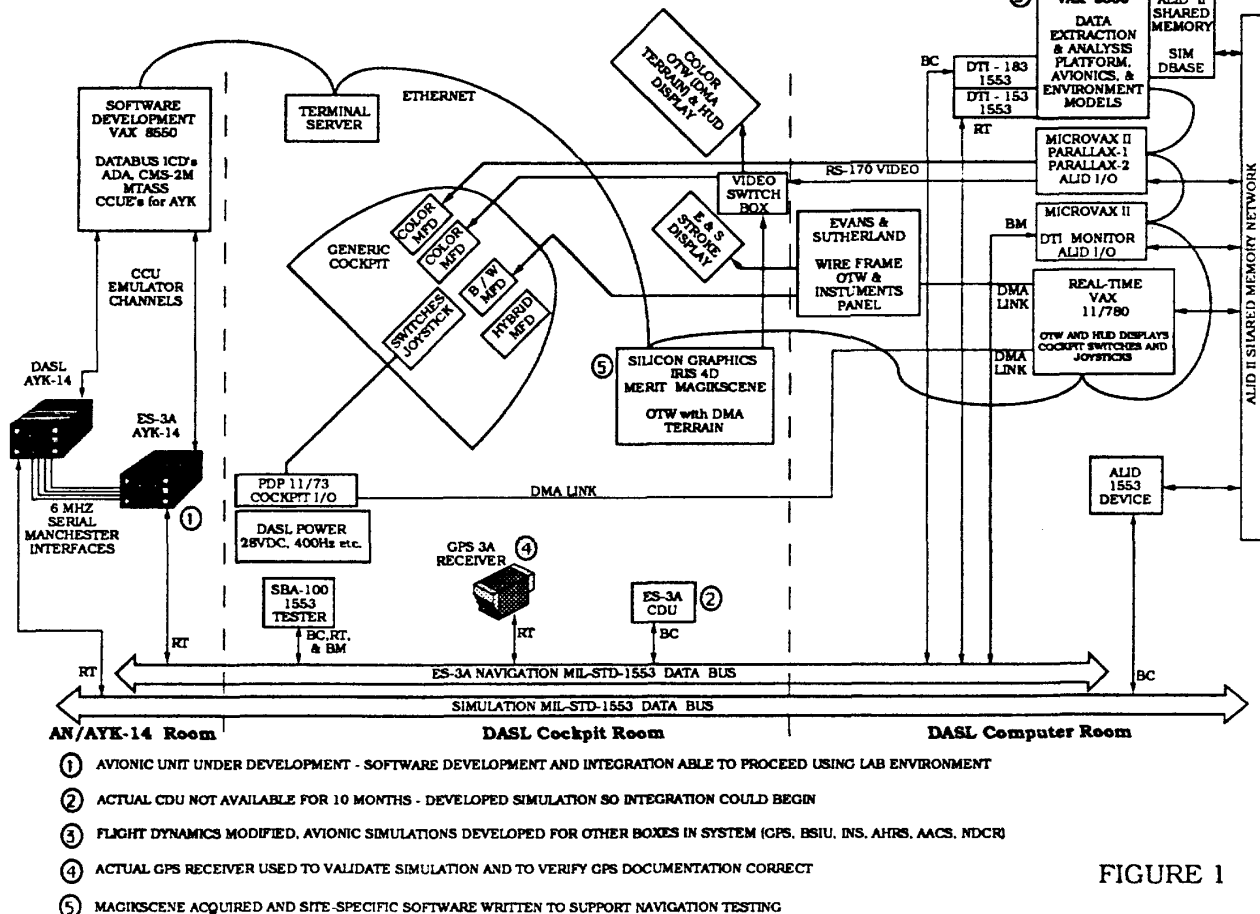


FIGURE 1

Several DASL hardware changes were necessary before the environment could be used for testing the ES-3A navigation and steering software. An ES-3A standard configuration AN/AYK-14 was installed in DASL to run the ES-3A navigation and steering software. Since there were no commercially available 6 MHz Serial Manchester digital interface cards for a VAX (as there are for 1553), a method to simulate the communication between the AN/AYK-14 and four of the navigation devices (Inertial Navigation System, Airspeed Altitude Computer System, Attitude Heading Reference System, and Navigation Data Converter Repeater) was needed. The solution was to install a second AN/AYK-14 which converts 1553 data from the Hot-Bench VAX 8550 into the 6MHz digital protocol. This provides a full feedback loop where the autopilot inputs steer the aircraft and the new position is reported by the navigation sensors which affect the calculation for the next autopilot steering command.

The GPS and BSIU simulations were verified by capturing 1553 interface data that had been time-tagged. This data was compared with the documentation for each device being modelled or with the actual unit if it was available. For the navigation sensors that used the 6MHz digital interface, the processing of mode changes was compared with the device documentation. The actual interface protocol was tested at Lockheed's System Integration Lab using a bus analyzer they had developed for the S-3B program.

Static Software Integration

Once a portion of the flight navigation software (including system executive, CDU command/response, waypoint control, sensor data integration, periodic message output, dead reckoning, and position keeping subprograms) was developed and integrated into an executable unit, it was taken to the DASL. At this point, the AN/AYK-14 code had been tested with a simple 1553 bus tester program that generated CDU messages to provide confidence in the Input/Output and flight navigation software. This portion of the software was sufficient to test the device interfaces and all of the static software for both the DASL simulations and flight navigation software.

The first task was to make the CDU simulation software operational. This included work on the user interface as well as the 1553B bus controller and specific meta-protocol used by the RTOS operating system in the ES-3A AN/AYK-14. All of this had been developed beforehand, but it required debugging with the actual flight navigation computer and executive software.

Once a reliable CDU interface and bus controller were established, the device interfaces were debugged. For each problem encountered it was necessary to determine whether the difficulty was with the flight navigation software or the device simulation model. A relatively short turn-around time for either simulation or flight navigation software modifications (about 20 minutes from start of compile to end of system load) was a definite advantage during this stage of development. Another advantage was that the flight simulation aircraft, cockpit, and environmental models were proven and trusted.

The mode processing subprogram, which was required to gracefully degrade in the event of a sensor failure in the navigation suite, was tested by disconnecting the sensors one at a time and verifying that the proper actions were taken. The simulations were used to test partial failures also. For example, the INS simulation could be set to quit supplying heading data while retaining valid data in the remainder of the message.

Finally, the combination of a cockpit to control the flight simulation and the ability to quickly change the aircraft location around the globe allowed efficient testing of all of the device interfaces. Situations dangerous or impossible in the actual aircraft were also tried, such as extreme altitude and inverted flight attitudes.

Dynamic Software Integration

By the time the position keeping, sensor integration, and waypoint control software in the flight navigation load was working well, the dynamic steering software was ready to test. The waypoint control subprogram was intensely tested by placing waypoints on opposite sides of the equator, poles, and the prime meridian and verifying that the aircraft properly flew to the waypoints and that the aircraft position keeping subprogram correctly maintained the aircraft's position. Once again, the main difficulty was determining whether problems were caused by the flight navigation software or simulated avionics.

The autopilot model was a particular problem. Because the flight navigation software is a closed-loop system, and has almost no aircraft-specific parameters, actual S-3 flight dynamics were not considered necessary. Since the simulated aircraft did not faithfully reproduce the flight dynamics of the S-3 airframe, some of the parameters had to be adjusted. For instance, if the simulated aircraft was to be rolled 30 degrees, the resulting turn radius would be considerably smaller than if an S-3 was rolled 30 degrees.

Therefore the roll command was scaled down by the autopilot to a more reasonable value.

Further, the rate constants used to convert tracking errors to control surface deflections had to be determined empirically. This was difficult since the flight navigation computer was initially responding very erratically. The initial constants were chosen to be very conservative, and adjusted for a more lively response once the flight navigation software was functioning better.

Finally, the autopilot model was initially designed carefully with great attention to theory and detail. It became apparent during testing that most of the detail was not needed and that a very simple model worked best. For example, the original design called for limiting control surface deflection rates and damping of their motion. While these are necessary in a real aircraft to prevent over-stressing of the airframe and hydraulics systems, the flight simulation had no difficulty with undamped control surface motion, and was inherently stable. The altitude-hold portion of the autopilot model (added later, and unrelated to the flight navigation computer) was designed to be as simple as possible. It was driven by a straightforward first-order comparison between actual and desired altitude. This type of system would be unacceptable in a real aircraft, but it proved to be effective for an avionics simulation.

To insure that the AN/AYK-14 could communicate with the actual 6MHz devices, some testing at Lockheed's S-3 System Integration Laboratory was done. Pitch, roll, heading, altitude, and airspeed were quantities that could be altered by moving a sensor or by using a device to blow air by pitot tubes. Although dynamic data could not be generated, this permitted testing of I/O, mode processing, and some cockpit display integration (since DASL did not contain the aircraft Horizontal Situation Indicator display).

Integration Problems and Solutions

One problem with using a simulation is the lack of visual references outside the aircraft to determine autopilot response and aircraft location. This problem was solved by using Merit's MAGIKSCENE software to provide a split-window display for OTW terrain as well as a top-down view of the area around the aircraft. Another advantage to using this software was its ability to display objects in the area. This capability was used to display navigational waypoints. The waypoints were constructed of variously colored mesh spheres that could be seen for many miles. Each waypoint consisted of three

concentric spheres of successively smaller diameter, centered at the altitude, latitude and longitude of the waypoint. The use of these visual aids made it easy to precisely study the dynamics of the flight path commanded by the autopilot.

Since the flight navigation program was required to plot great-circle courses, a method was needed to insure that the aircraft was indeed following a great-circle flight path. One of the more useful facilities of the flight simulation package was the ability to log flight parameters during a simulation. The latitude and longitude of the aircraft were logged while the aircraft flew from one location to another (typically from one waypoint to another). These logs were then transferred to a personal computer on which a great-circle path calculation program was running. The logged flight paths were compared against mathematically derived flight paths to verify that the aircraft actually flew the optimum path.

Another particularly useful facility of the simulation system was the ability to capture large amounts of 1553 bus traffic for later analysis. This capability was used extensively when trying to debug the 1553 periodic traffic problems encountered on this project due to the number of messages involved in the protocol. This capability also proved useful when trying to predict AN/AYK-14 response during the design and development of the firmware for the CDU.

Finally, when the flight navigation computer was integrated with the rest of the avionics systems in the full aircraft integration facility, the flight simulation system was valuable in providing realistic data.

Lessons Learned

1. Avoid reinventing the wheel. The accuracy of the simulated aircraft turn rates and autopilot response may not be exact but as long as a reasonable feedback system is maintained, the ability of the navigation system to navigate can be verified. The support environment such as Earth, ground, and aerodynamic models are available from other programs. If these must be developed (perhaps for a new computer system), design them to be generic so that flexibility is a goal. For example, the aerodynamic model can be adjusted by altering a data file containing lift and drag coefficients.

2. Make contingency plans during the simulation requirements phase for each piece of gear in the system. If the avionic box development falls behind schedule, a simulation may allow the

overall sub-system integration schedule to stay on track.

3. There is a requirement for the actual gear or at least test reports from the actual gear so that the simulation can be validated. Some particular areas of concern are interface data rates, length of time to change modes, and data format.

4. Some requirements for selecting a simulation computer environment include ability to control resources in real-time (interrupts, CPU allocation), maturity of compiler, full debugger support, and quick turnaround in the Compile/Link/Run integration phase.

5. System requirements WILL change. A simulation system that has a good user interface, is easy to modify, and is well documented will provide the flexibility needed to handle these perturbations. For example, on the ES-3A program, the GPS receiver firmware changed, the Omega system used was not the one originally specified in the requirements, and the Standard Central Air Data Computer replaced the Airspeed Altitude Computer System late in the development schedule.

6. Developing a test and integration environment at the onset of the project, rather than waiting until integration and testing become an issue is highly recommended. By identifying the effort required to develop a test bed, we could have justified procuring the necessary hardware, software, and interface equipment earlier, which would have allowed a longer period for testing and working out system anomalies.

Summary

The DASL has been a valuable asset in testing and integrating the ES-3A navigation and steering software. The lab offered us a dynamic environment in which to test the entire navigation effort of the flight computer in high-risk situations which could not be tested during flight test. The environment allowed us to prove the fault-tolerance of the navigation and steering software prior to flight test.

Using the DASL allowed us to test the navigation and steering software anywhere in the world without actually flying there, thus allowing us to test at critical points (such as the equator, poles, the prime meridian, and different hemispheres). We carefully tested the aircraft position keeping subprogram and steering subprograms and the performance of the system upon crossing these critical points. We expect flight tests to proceed with only a few minor modifications necessary.

This will provide more time to concentrate on the mission areas of the aircraft with confidence that navigation is providing correct data.

These integration techniques are applicable to new systems as well as the many avionic upgrade efforts that are currently being considered to implement GPS into navigation suites.