LESSONS FROM THE FRONT LINE: THE ROLE OF FLIGHT TEST IN AIRCRAFT UPDATE PROGRAMS

Capt David J. Hoey
Capt Matt E. Skeen
Maj Evan C. Thomas
416th Flight Test Squadron
118 N. Wolfe Ave
Edwards AFB CA 93524
United States

INTRODUCTION

Many nations today face the choice between procuring new aircraft or upgrading their existing fleet aircraft. An upgrade is frequently seen as a cost-effective solution to meet new mission requirements in a timely fashion. An upgrade allows the user to capitalize on technological advances since the development of the basic airframe. A key aspect of any aircraft program, whether an upgrade or an initial development, is the flight test phase. Flight test is the final stage where the new capabilities are evaluated for their likelihood to deliver added utility to the warfighter. However, given an avionics upgrade for a proven aircraft system, such as the F-16, the need for a flight test program is often questioned. "After all, it is only software" is a common comment. This paper will explore the need for, and benefits of, flight test in upgrade programs. It will address the economics of testing, examine the limitations of upgrades, and touch on issues of incorporating new technology into existing weapon systems. Examples and lessons learned from actual programs either tested or currently under test at the 416th Flight Test Squadron, Edwards AFB, California will be incorporated. These flight test lessons can be easily applied to other procurement programs.

BACKGROUND

The 416th Flight Test Squadron is responsible for over 50 ongoing F-16 test programs. Projects range across the spectrum of testing from a simple field service evaluation of new brakes, to testing a complete avionics modification kit, or entire new aircraft versions. This paper will focus on some lessons learned and as such may give the impression the F-16 is a weapon system infested with software errors, or 'bugs'. Nothing could be further from the truth. These are the experiences resulting from a large volume of flight testing, spanning a large number of customers and subsystems. The upgrades performed have led to the addition of very complex combat capabilities to further expand the F-16's operational capabilities.

I. What can be done economically?

Flight test is expensive, and fiscal realities will force a balance between desired and required testing. Despite advances in modeling and simulation, flight test remains an essential component of any test program. The breadth and depth of testing has a direct impact not only on the cost required to complete testing, but also on the confidence with which the upgraded system can be fielded. Testing of modern, complex systems poses a challenge which leans more toward increased testing depth. At the same time, modern expendable weapons are generally very expensive, making traditional full-scale firing trials a rarity. A live weapon delivery will greatly increase confidence in the system under test if it is performed in an operationally representative scenario. However, such scenarios are often costly and thus must be carefully chosen to get the most value from each test dollar. Specific examples include inertial-aided munitions (IAM) integration and advanced medium range air-to-air missile (AMRAAM) launches. This section will also consider regression testing and adaptation of commercial-off-the-shelf (COTS) systems.

Modern weapons are becoming increasingly complex. Flight test of these weapons becomes an integration effort involving multiple subsystems most often produced by different organizations. For example, an air-to-air missile may require tracking information from the fire control radar which is processed by a central computer to perform its role. The normal flow of evaluation for such a system involves bench testing of each subsystem followed by laboratory testing of the integrated system and finally flight test. Flight test often consists only of captive carry missions but may include a live weapon delivery. The benefit of integration laboratory testing is that it allows a thorough checkout

Paper presented at the RTO SCI Symposium on "Aircraft Update Programmes. The Economical Alternative?", held in Ankara, Turkey, 26-28 April 1999 and published in RTO MP-44.

of the aircraft weapon interface. The test team can quickly move through a large matrix of scenarios checking each mode of the interface. The benefit of captive carry testing is that the aircraft and target dynamics are present. An often-unrecognized disadvantage of both laboratory and captive carry testing is that an engineer or computer must verify proper accomplishment of each of potentially hundreds of steps involved in delivering the weapon. Confidence in the results is as much a function of the thoroughness of the engineering analysis as it is the thoroughness of the matrix of test conditions. Live weapon deliveries have the advantage that each of the steps required to operate at the test condition is verified by real world results.

For example, a live delivery of a new weapon at the F-16 Combined Test Force uncovered a problem when a sequenced multiple release, known as a ripple delivery, was performed. A ripple delivery was one of the scenarios tested in the laboratory, but the precise timing of the solenoids, which were energized to remove an arming pin from the weapon, was not verified. During the live delivery, the solenoids for the second weapon did not energize in time to pull the pin that armed the weapon. As a result, the weapon hit the target but did not function correctly. Post-mission laboratory simulations of the event clearly identified the problem. Without a live weapon delivery at this test condition, the problem would have gone unnoticed until the system was fielded. In the case of a high-value weapon which is infrequently delivered in training, the problem may not have been discovered until the weapon was employed in combat.

In another example, an F-16 was upgraded with a new central computer and fire control radar. The software for the new computer and the radar was rewritten in a different computer language starting from the specifications. In this scenario, it is common for the developers to downplay the need for testing because the functionality of the systems have not changed—the software has merely been converted to run on a new system. However, when software is rewritten from the specifications, there are opportunities for errors resulting from mistakes in both designing the code and in interpreting the specifications. A mistake in interpretation of the specification may not be uncovered until the integrated system is tested because the developer will test the system against the misinterpreted specification. If a problem such as this passes bench testing at the vendor and through functional testing in the laboratory, it may manifest itself as a performance problem that will require a realistic, operational scenario to uncover. In our example, latent data were being provided to a radar guided missile, due to an error made in the interpretation of the specification. The problem was not uncovered in contractor bench testing because the system was performing according to its design. The problem was not uncovered during integration laboratory testing because the integration laboratory used targets of opportunity to test the system instead of dedicated target aircraft instrumented to provide precise position truth data. Captive carry flight testing of the system allowed measurement of the data accuracy in a scenario closely related to operational employment. In this example, the problem was only uncovered during the detailed analysis of captive carry flights that preceded a live launch of the missile. At this point, it is useful to note a practical argument in favor of conducting live weapon deliveries. A live weapon delivery actually buys more than just the data from that specific test condition. It is the nature of the compressed timelines associated with today's development programs that engineers must prioritize the time they spend looking at any particular set of data. A live weapon delivery is a milestone in any program and forces the development team to focus its attention on the system to be tested. In the case of the F-16 missile launch, it is certain the problem would not have been uncovered until the weapon system was used in actual combat had a live launch not been scheduled. The data had already been received and undergone an initial review and the latent data were not discovered. Because of this experience, we recommend conducting live weapon deliveries that demonstrate the capabilities of greatest interest to the future system operators. This example also highlights the sophisticated test and range assets frequently required for flight test of complex weapons. Without precise position truth data, this deficiency would likely have been fielded in a production software release.

Many of the upgrade programs tested at the 416th Flight Test Squadron are primarily software modification programs. In order to limit the cost of flight test, less testing is performed on systems which have only slight modifications from a previously tested design. The test team must take care when determining the scope of a particular change. Rehosting software on a newer computer may reasonably be considered a relatively small change. Rewriting software to a more modern computer language like C or ADA should not be considered a small change. Because such software upgrades are commonly performed by producing the code from scratch, based on specifications, the resulting system is actually untested. If the goal is to reproduce the capability of the preceding system, testers may have the benefit of a performance baseline against which to evaluate the new system. It would not be correct; however, to assume that minimal testing is required for the new software code.

Evaluation of previously existing functions to ensure changes to the software did not alter previously existing capabilities is called regression testing. Experience at the 416th Flight Test Squadron shows most upgraded software will contain a few regression errors; typically of small impact, but some of serious consequence. Aircraft software is conceived of and built by humans who occasionally make mistakes. As

anyone who has worked in the spectral world of programming knows, these bugs can be difficult to find. To expect any complex software code to work flawlessly the first time would be unrealistic. However, a complete test of a software suite following changes to a few lines of code would be very expensive and not prove any more useful than a carefully selected test matrix of regression points. One method of reducing the scope of regression testing has been to develop modular, stand-alone software codes, sometimes known as 'plug-and-play' modules. The design concept is that software changes or updates will be focused in a specific module, and only that module will require detailed test scrutiny. Problems arising from 'plug-and-play' software updates will be addressed in the technological limitations section later in this paper.

All testing, whether it be laboratory, ground or flight test, is essentially risk-reduction. The cost of executing the test is weighed versus the cost of fielding a flawed system. Without a very large budget, it becomes an art to sift through the plethora of possible test scenarios and build a test plan. It is human nature to build a plan which focuses on new or changed capabilities. A complete test plan must also incorporate an effective way to check for regression errors which are very likely to exist in upgraded software.

Commercial-off-the-shelf (COTS) systems are often presented as requiring less test than systems which were newly designed. This is true in the sense that COTS systems will require less testing at the bench level because they are often well understood at this level. However, because a COTS system was not designed specifically to integrate with a particular aircraft, testing of the interface of the COTS subsystem with the aircraft as a whole must receive more focus. Experience shows two areas where a COTS system may have problems when integrated into an upgraded aircraft. First, a relatively new COTS system may not properly interface with the older architecture of the upgraded aircraft. Second, the COTS system may not have the desired military utility.

The integration of the digital terrain system (DTS) into the F-16 provides a good example of a COTS update program. The DTS was based upon the TERPROM™ system using radar altimeter readings and a stored digital terrain database to determine the aircraft's geographic position. The DTS predicted the aircraft's flight path using current position, velocity and attitude information. The DTS then compared the prediction to the digital terrain database. If a collision with the ground was predicted, DTS generated warning cues to the pilot. Other capabilities, such as obstacle avoidance and a terrain cueing system, similar to a terrain-following system, were also available. The DTS was a self-contained 'black box', with the primary algorithm operating on stand-alone hardware housed in the existing data transfer cartridge.

In Phase 1 of integration, the first task was to link the DTS 'black box' to the F-16's core avionics software. Changes to the core software were necessary because the DTS required information from the radar altimeter and inertial navigation system (INS). The DTS system also provided data to the F-16 core software, used for generating status and warning displays for the pilot. Once the F-16 core was modified, DTS was ready for flight test. Because DTS used a 'generic' fighter performance model with a limited envelope, the initial integration was expected to require some algorithm 'tuning' before fielding for operational use. The anticipated performance tuning and envelope expansion proceeded essentially as forecast. However, when DTS was matched with the F-16's high performance characteristics, unforeseen problems arose with the secondary capabilities of DTS. These problems required extensive analysis and algorithm modification. Additional software refinements were added as the users sought to take advantage of other potential capabilities. In the end, the 'tuning' process evolved into another full integration phase, with two more F-16 core software releases and numerous DTS software changes. The DTS / F-16 integration provides a vivid illustration of how a COTS system required not only significant modification of the F-16's existing core software, but also modifications to the COTS software, and a large flight test effort to produce a system with the desired military utility.

Conceptually the economics of flight testing aircraft upgrade programs are quite simple. The time and money required to conduct a flight test program should be balanced against the potential cost, in dollars and lives, of fielding a flawed system. The preceding examples were intended to provide the reader with some insight into the types of problems which are commonly uncovered in a flight test program. Hopefully this insight will be helpful in determining the appropriate amount of flight test for an aircraft upgrade program. The following section will address three types of limitations frequently discovered in testing an upgraded aircraft.

II. What are the limitations to upgrades?

Limitations, which remain unidentified until the flight test phase, tend to fall into one of three broad categories: technological, programmatic or operational. Unforeseen technological limitations may result from such things as avionics bus architecture, timing and protocol issues, mixing analog and digital systems, or the existent growth capability in the system. Some causes of programmatic limitations are being forced to 'do more with less' or the bureaucratic inertia of multi-user projects. Operational limitations are marked mainly by pilot-vehicle interface (PVI) problems, and

unforeseen shortcomings which appear during system employment.

Some technological limits arise from the special requirements of flight test. It is often impossible to monitor system operation without adding flight test instrumentation, which changes, to some degree, the system under test. This problem becomes more significant as the aircraft computer systems are consolidated resulting in fewer black boxes on the aircraft. In one upgrade program, the component which performed weapon ballistic computations and the component which calculated the aircraft height above target were replaced with a single component which performed both of these functions. Before the upgrade, the flight test instrumentation system could record the aircraft calculated height above target as it was communicated from one component to the other via the MIL-STD-1553 avionics multiplex (mux) bus. After the upgrade, the software in the new component was modified to include a "data pump" which put the height above target on the mux bus so the instrumentation system could record it. The 'data pump' was later disabled when the system was fielded. This resulted in a system under test that was different than the fielded system. Flight test instrumentation becomes more and more reliant on 'data pumps' as more and more operations are performed within a single aircraft component. Engineers require data from points within the operations in order to troubleshoot software problems. However, the greater number of 'data pumps' present in the flight test software may lead to greater differences between the system under test and the fielded system. An increase in the differences between the tested system and the fielded system causes a decrease in the confidence in the validity of the test results.

The chief differences between systems with and without a 'data pump' are the timing and quantity of messages transferred via the data bus. In one test program, computer halts and crashes occurred when the 'data pump' was functioning because the multiplex data bus did not have the capacity to handle both the normal data and the flight test 'data pump'. However, when the 'data pump' was disabled to allow production-representative bus traffic, no diagnostic information was available to troubleshoot a performance anomaly. In a new development program, the 'data pump' usually takes up some of the excess capacity of the mux bus. In an upgrade program, this same excess capacity may be required by a new capability. The 'data pump' may have to be modified to provide more types of data while using less data bus capacity. This generally equates to a more complex 'data pump' and thus a less production-representative system under test.

Developers implementing 'plug-and-play' modules can encounter technological limitations if they do not have a thorough understanding of the intended use of their subsystem. A thorough understanding of the integration of multiple subsystems can be hard to attain when multiple organizations are involved in a program, and each organization is striving to reduce the time and money spent developing its individual subsystem. The key component is often the core software, which integrates the different subsystems. As modules are pieced together one or two at a time, the core code and architecture should be capable of handling the load. As the modules become more numerous, ensuring they do not overload processing hardware capabilities or cause timing/interrupt problems becomes more difficult. Unforeseen interactions between the modules can also lead to serious deficiencies.

This exact scenario has been demonstrated several times on various programs at the 416th Flight Test Squadron. Main computer crashes have occurred when system A, which was developed for one customer and system B, which was designed for a second customer were implemented together on an aircraft for a third customer. In another program, computer halts occurred during bombing runs because the developer did not envision the pilot's use of an identification, friend or foe (IFF) airborne interrogator while the aircraft mission computer was configured for bombing. These problems are often the result of hardware resource conflicts and are not uncovered until the system is used in a particular scenario. Resource conflicts become more difficult to avoid as software becomes increasingly complex. Because the programmers and laboratory testers are not fighter pilots, they may build and test the code using false assumptions about system employment. Flight test planners should focus on operationally-representative scenarios in order to ensure the modes needed by the user will work as desired.

In this particular case, an aircraft is much like a personal computer system at work or at home. The major software companies' profits and viability ride on making each application easy to operate and resistant to crashing. Despite the best efforts of the programmers and testers, few people can say their computer has never crashed while performing an apparently routine task. Modern aircraft are more complex than personal computers, and there are many opportunities for problems to be caused by the interaction of the various subsystems. Also, the safety implications for an aircraft's computer crashing far outweigh the safety implications of a home computer crashing.

Another technological limitation observed by the 416th Flight Test Squadron is the difficulty in adapting analog communications systems to transmit digital data. Any system, which utilizes the existing aircraft radios to transmit or receive data, may encounter technological limitations. The radios in most fielded aircraft were not designed for digital data transmission. An upgrade program aimed at adapting these radios for digital data transmission may encounter some difficulty,

and the contractor may be unlikely to highlight these limitations beforehand.

The sheer inertia and bureaucracy of large multi-national or multi-service development efforts can result in programmatic limitations to what can be accomplished through an upgrade program. The Link 16 Multifunctional Information Distribution System Low Volume Terminal (MIDS/LVT) and the Joint Direct Attack Munition (JDAM) are examples of multi-user development efforts. Multi-national and multi-service development efforts have the benefit of allowing multiple users to pool their resources to develop a subsystem. However, with multiple users come multiple sets of priorities. Each user has their own operational doctrine and views on how best to employ the upgraded system. 'Joint' programs from more than one service of a single country have enough problems with this. 'Joint' programs involving more than one country can encounter such large programmatic limitations the ultimate result is program cancellation. Making significant changes to a program once it has reached the flight test phase can be extremely difficult.

From the flight test perspective, such programs have the disadvantage of being inflexible. Because each user must approve the subsystem design, it becomes difficult or impractical to change the design when platform-specific deficiencies are uncovered in flight test. From a technical standpoint, the fix to an integration problem might be more appropriately accomplished in the new system, but the inertia of the program may make this impossible. The aircraft developer will be forced to make changes to fix the problem on the aircraft side of the interface. Such unplanned changes will certainly increase costs, could strain avionics system resources, and may decrease the combat capability provided by the upgrade.

Operational limitations occur when a system functions as it was designed, but turns out to be less useful than anticipated. In aircraft upgrade programs, cockpit displays provide many good examples of operational limitations. Most cockpit displays were designed when there was less information available for display. When new information is added to the display by successive upgrade programs, it is possible to overload the pilot with too many symbols. An example might be a horizontal situation display that shows each aircraft in a four-ship formation and the target being tracked by each aircraft. Add navigation routes, geographical borders and radar steerpoints and the display may become so cluttered as to actually decrease the situation awareness of the pilot.

A related problem can be caused when a new system adds to the list of symbols being displayed to the pilot, but the display processor is not upgraded to handle this additional information. In some cases, the display processor has been overwhelmed and 'locked up', thus causing a complete loss of situation awareness instead of just possibly a degraded state of awareness by the pilot. This anomaly requires the pilot to perform some type of 'head-down' operation to restore correct operation of the displays.

Problems will be uncovered during the flight test phase of any upgrade program. A clear, well thought out 'feedback loop' needs to exist to incorporate the findings from flight test into the production tape. This process needs to be well defined in the early stages of any program and not wait until the first 'show stopper' anomaly has been encountered. It is essential to incorporate the most refinements into the production version of the upgrade. These problems are more likely to remain in the fielded system and become limitations if the upgrade program was planned with very little margin for error in a technological, programmatic or operational sense.

III. How can technological advances be integrated?

The underlying theme in today's technological advances is complexity. As operational requirements for weapon systems grow, so does the complexity of the integration. Precision weapons are inherently more complex; that's what makes them 'smart'. The complexity of these systems makes performance analysis difficult for both developers and testers alike. With older, less intensive avionics suites, system performance was usually readily apparent to the pilot. Scoring no hits on an aerial target, with a stable tracking solution on the target, was a straightforward indicator of a gun sight computation problem. Likewise, stray bombs could highlight a bombing deficiency. Today, the weapons themselves communicate with the aircraft via the avionics data bus, and it is prudent to evaluate the communications for all weapon modes, using the most economic testing methods available. This might be analogous to evaluating the internal communications between a computer's CPU and a floppy disk drive. Clearly, the typical computer user would have extreme difficulty evaluating the 1's and 0's passing between the core software (CPU) and the subsystem (floppy drive). This example is useful in illustrating not only the depth of testing required, but also the breadth. Imagine the effort required in testing the computer with a wide variety of applications that might use the floppy drive.

With increases in effective weapon range and system complexity, the question of how to evaluate such weapons without an actual launch or delivery has become more important. The displays to the pilot may or may not be linked to actual system performance. Models used by the avionics computers, such as launch zone and time of flight computations, may have areas of inaccuracy. Even worse, the models may be engineering approximations coded into the software before the actual weapon performance characteristics had been determined. This is increasingly becoming the

case as weapons development timelines continue to lengthen. Concurrent development on the aircraft side of the weapon interface is used to shorten the overall development cycle. Using approximations throughout the integration effort can have significant drawbacks, and possibly result in significant problems being found very late in the development cycle.

A careful distinction must be made between using flight test results to verify a model, and using a model to verify flight test results. In one example at the CTF, the proper operation of an existing ground recovery algorithm was regression tested by a number of programs. The aircraft's flight parameters were entered into the 'known' algorithm to verify if the pilot warnings were displayed when the model predicted they would be. The model was used as a truth source because it had been extensively evaluated years before. Unfortunately, an error had been introduced during a minor algorithm refinement. When an experienced engineer used flight test results to verify the model, the flaw was found. Subsequent ground-based testing allowed economical identification of the erroneous portions of the algorithm for correction.

Summary

Aircraft upgrades can be an economical alternative to new aircraft purchases. The key is careful overall program management from the beginning. Any upgrade project must include a realistic testing phase, including flight tests where appropriate. The cost of flight testing a particular feature must be carefully weighed against the risk of fielding a system or component that might not be operationally useful. Although a COTS system should not require very much component testing, it will require a fair amount of integration testing with the host vehicle. The test phase must be allocated enough resources to handle any technical, programmatic, or operational limitations encountered during testing. With the complexity of today's aircraft upgrades, it is unrealistic to expect to have no errors in the first iteration. System complexity may also drive a need for increased test range support capabilities, engineering analysis costs, and possibly expensive live weapon deliveries. A well thought out process to feed the results and refinements from flight test back into the production tape must be established in the early phases of a program. Care must be taken when integrating technological advances into an upgrade program. Concurrent development on both sides of the aircraft interface is usually required and drives the need for using models and approximations in the early phases of design. However, models and simulations should never be used to verify flight test results. Properly handled aircraft upgrades can achieve a significant increase in combat capabilities in a shorter time span and for fewer resources than a brand new development program. The development history of the F-16, from a daytime dogfighter to today's multi-role, precision-strike weapon system, provides ample proof.