

# How to Build a 20-Year Successful Independent Verification and Validation (IV&V) Program for the Next Millennium

Jon Hagar and Lisa Boden  
Lockheed Martin Astronautics Company  
Mail Stop H0512  
P.O. Box 179  
Denver, CO 80201  
303-977-1625  
fax 303-977-1472  
jon.d.hagar@lmco.com and lisa..m.boden@lmco.com

**Abstract:** IV&V is common on many critical software government programs. IV&V can save missions and improve the product quality when done right. This means that there must be a balance of people with the right skills, management, processes aimed at verification of each life cycle step, validation with simulation and analysis tools, and a hardware-based test facility. Automation of testing and IV&V helps to reduce cost, but automation is only part of a complete program. While IV&V is not for every software program (because of redundancies), high risk projects where there are critical cost or safety factors can benefit from some level of IV&V.

**Keywords:** IV&V, Verification, Validation, Hardware-based Test Environment

## Introduction

Lockheed Martin started supporting Independent Verification and Validation (IV&V) programs in 1979. Projects supported have included both booster and spacecraft vehicles. A 1991 report from an Air Force review concluded that one Denver area IV&V program had found errors that would have impacted mission performance of the subject software. Errors found by IV&V tend to be more numerous during preliminary development efforts, though review of data shows that some errors have continued to be found by IV&V well into operations and maintenance efforts.

This paper examines some of the lessons learned from almost 20 years of testing and how these will carry the programs forward in the next millennium, with the following considerations:

1. What is IV&V?
2. When is IV&V beneficial?
3. What are the key elements of a successful IV&V?
4. What are the experiences from a history of IV&V on a variety of projects?

## What is IV&V and What Are We Testing?

In Denver, Lockheed Martin's IV&V approach involves different levels of testing and analysis. IV&V testing concepts originated in Air Force programs wishing to achieve reliable software. In our IV&V, Verification is defined as the iterative process of determining whether the product of each step of the life cycle fulfills all of the requirements levied by the previous step. Validation is defined as the evaluation, integration, and test activities carried out at the system level, to ensure that the final developed system "works" [Wallace 89]. The need for IV&V sometimes is questioned. NASA studies have concluded that its effectiveness can be low [McGarry 82]. However, our experience leads us to conclude that IV&V can be successful and effective, where effectiveness is defined as the ability to find errors, if the testing of software is conducted from a user and/or system perspective to assess attributes such as function, performance, usability, quality, and reliability. This systems view of the software test process allows the consideration of all classes of errors that can be practically detected in software. These errors or faults, as defined by [Howden 91], include programming, deductive (e.g., translation from design to code), and abstraction errors. To accomplish this, software evaluation must be done both at a Verification and a Validation level by engineers skilled in more than just the software disciplines.

The software systems we are using IV&V on have the following characteristics: real-time; booster flight control; minimal human intervention possible; embedded within the system it controls; and numerically intensive calculations of such critical items as trajectories, flight dynamics, vehicle body characteristics, and orbital targets. The development programs usually are small, consisting of less than 50,000 source lines of code in two separate computer systems, yet these programs are critical to the control and success of the entire flight system. An example mission profile is depicted in Figure 1.

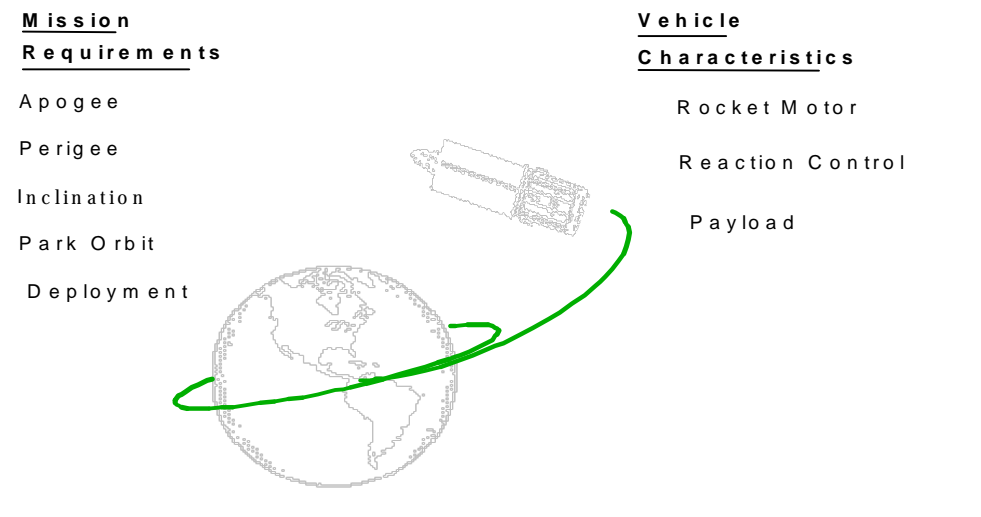


Figure 1 - Complex Software Requirements

Our test approach spans from the software unit level to integrated hardware and software (a system). Additionally, we have issues of configuration control, change management, documentation, and business management. In all of the test activities, we use a variety of supporting software tools and metrics. The goal of our IV&V is to show that the flight software is ready for use and the chance of catastrophic mission loss due to software is minimized. The software under test is the guidance, navigation and control software of the system, and so test programs that involve the fully integrated system are not possible. For example, we cannot fly the software in a "beta" test on an actual system to see if it will work in real use.

### Three Main Process Elements of our IV&V Process and Tools

Our product area's testing tools address various levels of abstraction (Figure 2) of the software system. In our approach, the lowest testing level is structural verification testing conducted with a digital simulation and/or hardware system. At this level, verification testing is done to ensure that executable programs implement such things as requirements, design information, and software standards. This testing is done at a module level with small segments of the code being executed in isolation from the rest of the system. A tool executes code in a simulator to support analysis of individual equations and simple logic structure. The comparison and review of results at this low verification level uses models constructed to replicate the code unit and an automated comparison tool (Adatest by IPL) to reduce human interaction. Results information is captured in a Web-based test procedure tool/system [Hagar et al], as well as in a requirement traceability tool (RTM by Marconi). This process is still human intensive and additional modeling techniques, such as Tvec by Tvec Technologies, are being explored in some Lockheed Martin areas.

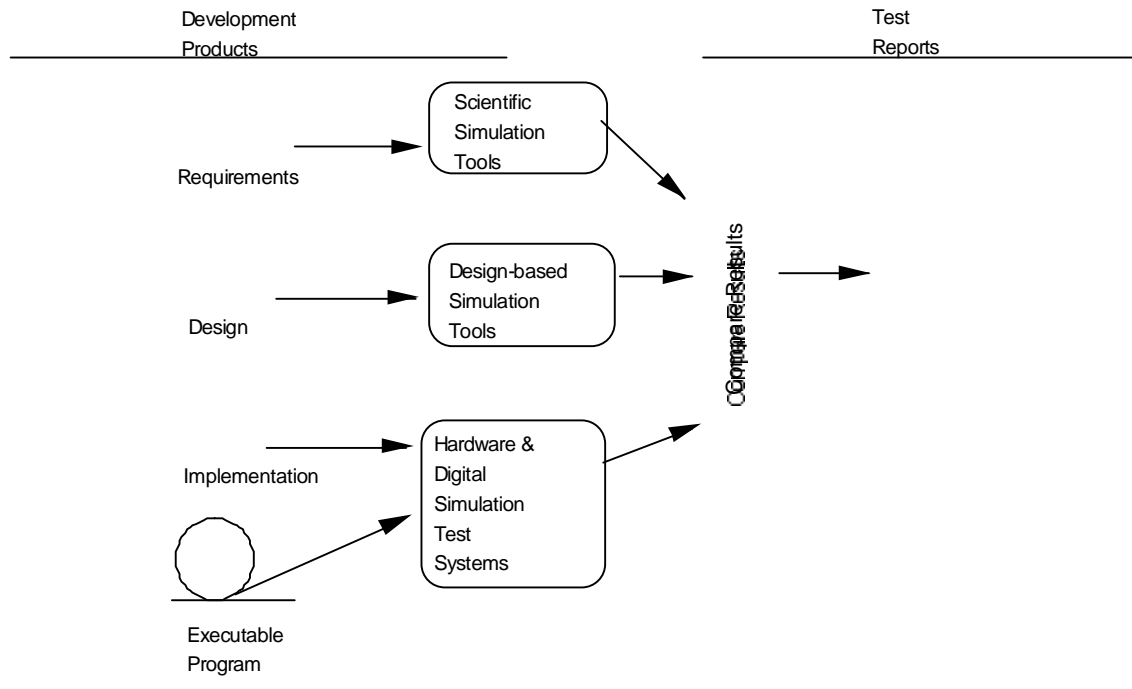


Figure 2 - Tested Products

The next level of testing involves what we call scientific validation, which demonstrates requirements compliance using scientific-simulation tools. These simulations are done in both a holistic fashion and on an individual functional basis. For example, a simulation may model the entire boost profile of a launch rocket with a full 6-degrees of freedom simulation, while another simulation may model the specifics of how a rocket thrust vector control is required to work. This allows system evaluation starting from a microscopic level up to a macroscopic level.

At the system level, software is tested with actual hardware in the loop. An extensive real-time, continuous, digital-simulation modeling and feedback system of computers is used to test the software in a realistic environment. Testing in a realistic environment means testing the software as a "black box", with the same interfaces, inputs, and outputs as an actual flight system. In the test bed, the computer is surrounded with a first level of electrically-equivalent hardware interfaces. Signals are input into the test bed to simulate the performance of the system and hardware interfaces. Since the test system runs in real time, there is no way to speed-up or slow-down the system. We call this level real-time Validation, since it is primarily concerned with testing the integrated end product against "system" requirements.

Numerous tools support the tiers described above, and many of these tools are simulation models based on requirements, design information, or the computer architecture. The tools are stand-alone and often custom-built software programs that execute on separate platforms from the software under test. These tools take data that could be the input to the system under test, and produce expected outputs. The tool and simulation outputs can then be compared to the results generated by the actual software being tested. Some of the tools simulate individual equations or logic sequences, while other tools simulate aspects of the entire system. Scientific simulation-based tools provide success criteria or analysis capability that allow engineers to judge the success of the software under test without relying entirely on human judgment.

Overall, this approach and tool set has been successful in taking input from the development products, performing IV&V and generating test results. An important part of our IV&V culture is to look for improvements in our testing processes and tools for both time and cost savings. Recent improvements

have included the use of unit level COTS automation, tools like RTM instead of custom-built tools, peer-based inspections, reusable assets with tools like Matlab, and Web-based technologies (See Table 1).

Table 1 - Sample of Standard Tools

<u>Activity</u>	<u>Tool</u>	<u>Function</u>	<u>Benefit</u>
Verification	AdaTest by IPL	Coverage	Measurement of test
Scientific Validation	Booster Utility Program	3-degrees of freedom simulation in Matlab	Assessment of data values
Validation	Real-time closed loop hardware test bed	Execution of software supported by Web Tools	Assessment of software realistically

### **Key Element: Hardware-Based Lab Testing with Real-Time Closed-Loop Simulation**

The hardware-based test bed (lab) is the primary validation test facility for our IV&V, and it is integrated with the verification and scientific validation efforts. The lab is used to execute full and partial mission runs, up to eight hours long, designed to validate mission requirements. It provides a certified (the lab itself has been tested) test bed environment for validation of flight load products under simulated real-time conditions using avionics hardware and software simulators. The lab's flight hardware and avionics simulator interfaces with an Silicon Graphics (SG) computer, which simulates the mission environment and vehicle subsystems including propulsion, mass properties, and flight controls hardware. This provides real-time, closed-loop inputs to the test bed. Thus, the lab provides the capability to perform simulated full-mission runs from pre-launch or pre-deployment through spacecraft release and booster deactivation.

The lab consists of two sub labs: a Guidance and Control Lab (GCL) and a Scientific Simulation Lab (SSL). The GCL is made up of flight-prototype hardware and custom-built avionics simulators. The SSL consists of support equipment, such as a SG computer and real-time data linkages. The GCL avionics simulator is used to emulate flight hardware not available in the test bed. Also residing in the GCL is the Checkout Station (COS), which has the capability of issuing uplink commands to the computers, as well as recording downlink telemetry and checkout status of the airborne system. The two areas work together to provide a realistic environment for the flight software. The SSL is used to simulate rotational and translational dynamics of the vehicle. The SSL/GCL combination is then used to make real-time 6 Degree-of-Freedom (DOF) trajectory runs, with the added capability to create restart images along the way. The SSL/GCL test bed is functionally shown in the lab Validation Concept in Figure 3.

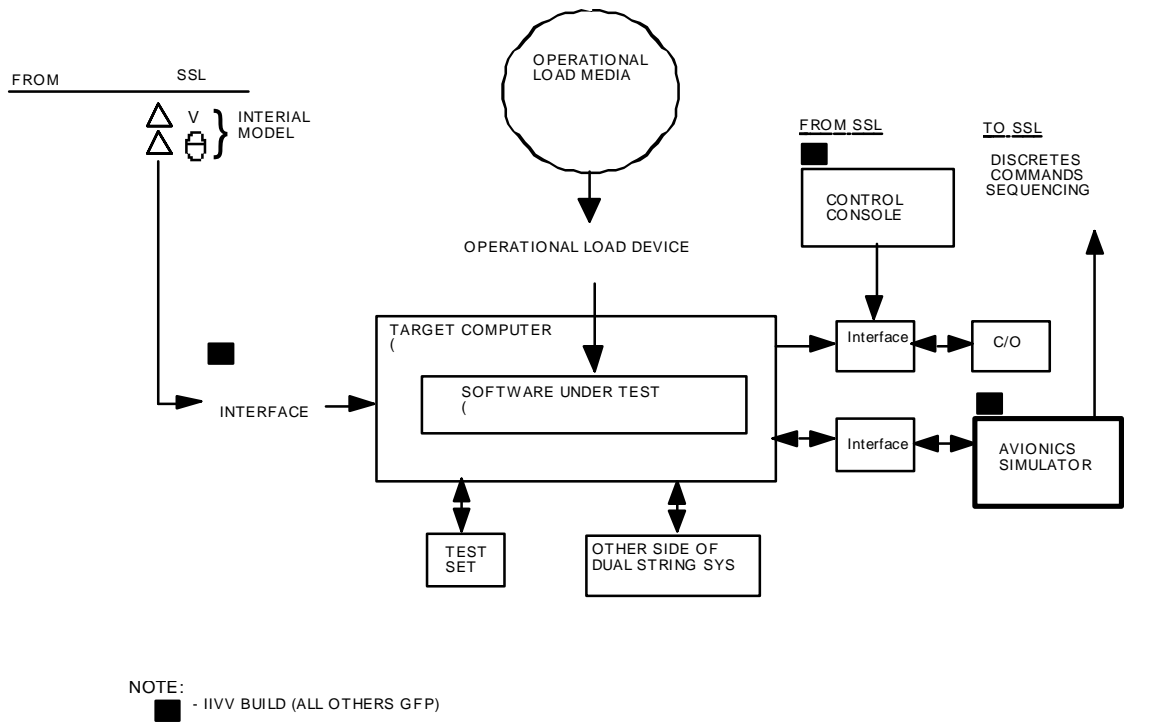


Figure 3 - Real-time Hardware-based Validation Concept

### When to consider IV&V

IV&V is not right for every program. Comprehensive, fully independent, IV&V is justified when the risk from software failure is high. Risk can be from cost or safety factors. IV&V must also be considered from the view point of the customer and management. If customers and management are willing to do active engineering with software, where other people can provide “outside eyes” of independence, then less rigorous IV&V may be acceptable. IV&V certainly can find errors, but if the cost of the errors being found do not offset the cost of the IV&V, then benefits can be questioned. IV&V can be considered in medical, air safety, nuclear, space, government, or other high risk software programs.

There are now companies and IV&V centers like NASA’s IV&V center that can provide resources and expertise on IV&V. While not for everyone, these centers can provide knowledge bases even to more standard test approaches.

### Experience and Data Found Over a 20 Year Life Cycle

IV&V is common on many critical software DOD programs, but there are different levels of IV&V. The range is from review, audit, and/or inspection to full and complete test programs with hardware-in-the-loop test beds. The latter seem better at finding errors and improving products, but represent a cost factor that can be equal to the cost of developer-based testing.

IV&V should be focused on negative testing, in which finding errors is the goal. It should do this by having all levels of testing (unit to full software qualification), multiple tools (simulations, instrumentor, oracles, etc.), and knowledgeable people. IV&V should be viewed as serving in place of the customer performing detailed acceptance testing as defined in ISO 9001.

The keys to exercising a successful IV&V program are tools, people, knowledge about the test domain, defining processes with metrics, and an interest in improving each of these. While other papers that we have written addressed tools, people, and process, we would like to note here that continuous improvement efforts have contributed both to IV&V success and longevity. Our IV&V area, with the support of the customer, has continuously and in small affordable steps upgraded tools, people, and

processes in line with Software Engineering Institute concepts. Trade studies and upgrade plans were prepared at many points along the line of improvement. We have moved from large mainframes to micro computers and workstations, and from custom-built tools to commercial products and new languages. Test processes and tool improvement is measured and evaluated against planned expectations. Not every improvement or tool has been successful, but we learn from our failures as well as successes.

Another experience we have found is that test automation works but up front costs and risk issues must be considered. During automation efforts, we first established what our manual processes were. If a manual process was repeated often enough to justify the cost of automation, then we budgeted development effort to test tools. The trade off point between manual and automation was basically to calculate the total cost of manual efforts and then do the same for automated efforts. If automation had less cost and time or there was some major technical advantage, then we would elect to automate. Automation costs include the up front effort to buy, integrate, and then set up tests. It is important to note that not everything we do is automated, but when repetition or some risk factor, like the chance for human error, indicate automation is viable, we develop it using a software development plan, just like any software effort.

### **Data From IV&V Detected Errors**

Over the years, we have found that IV&V finds the most errors and results in the best products when done with full testing and analysis. Part-time or review and audit types of IV&V, which we have done, are not as comprehensive and have missed critical errors. They provide some level of product improvement, and so their costs may be worthwhile, but to obtain the higher levels of error detection, the program must spend the money for full IV&V testing and analysis. Quality is not free, but the lack of quality can cost in the long run too. It is a trade off that must be considered during program planning by management and customers. To do this, we consider the following statistics from the IV&V problem report database:

- Over 1600 error reports have been filed by IV&V in one area, the number represented about 60% of all errors found by IV&V since many errors were reported informally (outside of the tracking database via memos or meeting action items) on pre-release engineering products to save costs by analyzing products while they were still under development. This was an important cost reduction factor.
- 30 percent of all software problems were found by IV&V on products we tested.
- 20 percent of IV&V defined problems resulted in code or requirement changes, and included errors that could have resulted in degraded mission performance.
- An Air Force review estimated that at least one mission over the last twenty years had been impacted by IV&V, which more than paid for the costs of IV&V.
- Over 80 percent of problems found by IV&V are in documentation. While not impacting code, finding these kinds of errors can reduce maintenance costs and help in process improvement related to concepts such as the Capability Maturity Model and ISO.
- The kinds of IV&V logic errors found include: configuration management problems, code standards violations, incorrect data parameters, logic faults in the code, design faults, and incorrect requirements that impact the design and code. These are equally distributed over the remaining 20 percent of error reports.
- Over 75 percent of the total errors were found during initial development. However, after almost 17 years of operations and maintenance on one program, some 30 error reports were filed during a major system upgrade. Two of these reports impacted flight products. As the software matured, the shift in problem reports was from code and requirement errors to documentation problems.

- Problems are found by inspection, test, and analysis using both manual and automated approaches. Problems are equally distributed for the verification, validation, and scientific validation areas

## Lessons Learned

The following represent some of the main lessons learned by the IV&V program during the past twenty years of testing:

- The IV&V team should be involved at the earliest possible time in the software development cycle. For established programs, the developer should provide extensive technical information and documentation of the system design.
- The IV&V scope must be established and agreed to by all involved parties. Scope would include statement of work, reporting structure, interface requirements for formal deliveries and support, and final deliverable software procedures.
- It is best and most independent if a separate budget controlled by the IV&V contractor is established so that changes in scope and deliverable products can be negotiated.
- Establish a customer for the IV&V effort external to the software development group. In order for the IV&V effort to have an influence on the software design, findings and recommendations should be reported to an interested third party with organizational responsibility for the software development group.
- A formal data exchange list should be created between the developer and the IV&V organization, along with a format to control interchange of documentation and products.
- Establish combined schedules for all tasks and maintain active review and notification of schedule impacts.
- Accomplish active planning for all tasks including review of schedules, metrics (trending and projection), and deliveries.
- Establish a forum for exchange of technical and schedule information on a regular basis. Weekly seems to work best.
- There should be an independent development of a requirements traceability matrix to ensure all requirements are testable and to track changes and requirement closures.
- Complete configuration management of all delivered products should be implemented by the IV&V team (know what you are testing).
- Increasingly with the aging of a project, a primary focus of testing should be to validate the interfaces between software and hardware elements. Compatibility problems are hard to find and often remain even after extensive testing and IV&V. Good interface definition helps, but systems of any complexity hide interface problems.
- There is a hidden cost associated with using reuse and heritage software. Typically, the requirements documentation and the interfaces can have problems that complicate their use.

- The cost and time to develop the test software, process, and tools can often be underestimated. This can result in failure of these types of efforts. Also, part of the cost that is often overlooked is the training of people to do the work.
- For critical or safety related code, it is important that no lines of code exist for which there is no corresponding requirements information or objective evaluation criteria (design level). A requirement that only states what to do and is not clear is also incomplete. To develop tests and test requirements, it is important that the software requirement be stated in such a way that a test or IV&V engineer can infer how the code is supposed to function.
- If present, it is important that software quality assurance group be staffed with personnel who are experts in software development engineering, since they need to make decisions about the validity of tests, tools, processes, and IV&V efforts.
- Establish a common problem reporting and tracking database with formal use and access procedures between all users.

### **Summary**

IV&V has contributed to mission success within the Lockheed Martin Denver Astronautics group, but we have found that IV&V has to be done correctly. A good working relationship with the developer and customer has to be maintained. IV&V had to define a plan that used methods, tools, and techniques to test the products and find errors. Once the IV&V plans were defined, these had to be managed and controlled to produce results. This includes obtaining and training the technical staff to do the job right. And, finally, over time the plans and associated elements had to be optimized to ever improve technical and budget performance. The role of independence in testing may be changing but we should not forget some of the lessons learned for IV&V.

### **References**

- D. Wallace and R. Fujii, "Software Verification and Validation: An Overview," IEEE Software, May 1989
- F. McGarry and G. Page, "Performance Evaluation of an Independent Software Verification and Integration Process,": Tech. Report SEL 81-110, NASA Goddard Space Flight Center, Greenbelt, MD., Sept. 1982.
- W. Howden, "Program Testing versus Proofs of Correctness," Journal of Software Testing Verification and Reliability, Vol 1, Issue No. 1, 1991.
- M. Deutsch and R. Willis, "Chapter 2: Overview of the Book", Software Quality Engineering A Total Technical and Management Approach, Prentice Hall, Inc. 1988, p33.
- Hagar, Burba, Wittekind, and Bell, "World Wide Web Based Tools to Support Testing", 1997 Qual Week.