Verification and Validation Implementation at NASA

Dr. Linda H. Rosenberg
NASA, Goddard Space Flight Center

Any company that produces or relies on software knows the importance of high quality and reliability. Recently NASA focused on applying Independent Verification and Validation (IV&V) as part of a software improvement effort. This paper is not about IV&V, but about NASA's new IV&V implementation approach on all software development throughout the agency. This information is valuable to any project, organization, or company considering applying IV&V.

NASA is recognized as a leader in space technology, using cutting-edge science to probe galaxies never before seen by mankind. In keeping with this cutting-edge technology, much of the functionality previously done through hardware has transferred to software, including mission critical functions. But technology implementation is moving so fast, that at times quality assurance cannot keep up, although we try. The NASA Independent Verification and Validation (IV&V) Facility was established in 1993 in West Virginia and tasked to support NASA projects in achieving the highest levels of safety and cost effectiveness for mission-critical software. Despite this, NASA has experienced some failures recently that were traced, in part, to less than adequate implementation of mission software.

NASA determined the need for software IV&V after evaluating the causes of recent mission failures. These were due, in part, to software issues that should have been identified during development or testing. The NASA IV&V Facility was developed to be a center of excellence, but was underutilized. Projects that did use the facility had proven benefits. Thus NASA's focus for improvement includes increased software IV&V application.

NASA began by looking at available resources and projects that had applied IV&V to determine the benefits in the NASA environment. It was quickly determined that only a few projects were implementing IV&V; not all were taking advantage of the facility's expertise. There were very definite proven benefits to NASA when IV&V was applied. These ranged from cost savings to identifying mission critical errors not previously identified through testing. Applying IV&V on this software resulted in increased safety and mission reliability. However, an identified deficiency was that there was no consistent application of IV&V by the projects.

This paper discusses the approach taken to increase the use of IV&V within NASA. We will start by defining independent, verification, and validation. We will then discuss the written policy relative to the performance of software IV&V, and the criteria developed to help projects quantify the need for IV&V.

Independent Verification and Validation

A basic understanding of what constitutes IV&V begins with the definitions of verification and validation, then determines what is required for "I" – independence. The Institute of Electrical and Electronics Engineers (IEEE) 610.12-1990, Standard Glossary of Software Engineering Terminology, defines verification and validation [1]. Verification is defined as the process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase. Validation is defined as the process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements. Verification asks "Did we build the system right?" and validation asks "Did we build the right system?"

IEEE defines *independence* in IV&V as three parameters: technical independence, managerial independence, and financial independence. Personnel who are not involved in software development achieve technical independence. IV&V personnel use their expertise to assess development processes and products independent of the developer. They formulate their own understanding of potential problems and how the proposed system is solving them.

Managerial independence requires responsibility for the IV&V effort to be vested in an organization separate from the organization responsible for performing the system implementation. The IV&V effort independently selects the segments of the software and system to analyze and test, chooses the IV&V techniques, defines the schedule of IV&V activities, and selects the specific technical issues and problems to act upon. Most projects view V&V as sufficient and do not recognize the added value the independence brings.

Financial independence has been harder to attain. All work on the project, including quality assurance, is funded directly by the project; hence, IV&V is also funded directly by the project. In theory, the project team could remove IV&V funding if they are not satisfied with the findings. But with the implementation of the IV&V policy, projects are now required to work with the IV&V Facility to reach an agreement on the amount of IV&V and funding. The center director must agree to any changes to this along with strong justification by the project manager.

Implementation Approach

All software project managers, whether government or industry, never have time or money to spare, so when NASA identified the need for IV&V, it was met with a cry of, "Not on my project!" But the implementation of IV&V was part of a larger effort to improve the software developed at NASA, to achieve the highest levels of safety and cost effectiveness possible for mission critical software. The NASA IV&V Facility provides tailored technical, project management and financial analysis for NASA projects, industry, and other government agencies, by applying software engineering "best practices" to evaluate software risk and criticality throughout the system development life cycle. Through the facility, NASA has the means for implementing IV&V, but it is under utilized. Only a few very large NASA projects, such as Space Station, chose to apply IV&V through the facility.

To change this, the first step was to write a policy requiring projects to investigate the necessity of doing IV&V. It is an effective risk mitigation strategy, and since most NASA missions are cutting edge technology, they also are at high risk. Recognizing that cost must be balanced against potential benefits, the "amount" of risk incurred by a project had to be assessed. A policy was developed that included the process of determining the need for IV&V, the extent, and approach. The policy also states all IV&V will be done under the management of the NASA IV&V Facility, centralizing expertise and ensuring consistency.

The next step was to develop criteria determining when to consider IV&V. This required quantifying project risk for an initial assessment on which projects may require IV&V. Project risk is defined as a combination of the probability that an undesirable event will occur, and the consequence if the event does occur. The IV&V criteria were written using probability and consequence. Factors influencing software development were identified, and risk factors associated with them for a calculation of the probability. Consequences of failure were classified as Grave, Substantial, Marginal, and Insignificant. These are combined for a determination of the necessity of IV&V.

The final step, and the hardest in some respects, was to identify the projects that potentially required IV&V and to determine to what level IV&V was needed. Money is always an issue; there is never enough in any software development. So what was the cost, and what are the balancing benefits?

The approach to implement IV&V consistently and logically on all NASA software was broken into three steps:

1. Write a policy for the requirement of IV&V implementation.
2. Write the criteria for an initial determination of IV&V necessity.
3. Work with projects to implement IV&V.

Step 1: IV&V Implementation Policy

The policy was to clearly specify the process of determining when a program must apply IV&V under the management of the NASA IV&V Facility. One strength of the policy is the specification that the NASA IV&V Facility is responsible for the management of all software IV&V efforts within the agency. This creates a central repository of knowledge, tools, metrics, and lessons learned that can be used to improve the IV&V efforts on future projects throughout NASA.

The policy states that each project must produce, document, and implement a plan that addresses V&V performance; and if appropriate, IV&V through the software life cycle – from requirements through delivery and maintenance. The level of IV&V of software that is performed is based on the cost, size, complexity, life span, risk, and consequences of failure as defined using the criteria explained in the following section.

Step 2: IV&V Criteria

In order to accomplish the goal to increase the application of IV&V on all appropriate projects, it had to be determined what was meant by "appropriate" projects, hence the development of IV&V criteria. Looking back on the IV&V objective for risk mitigation, those projects with high risk had to be identified; but first, the criteria for what makes a software project high risk had to be defined and quantified. The quantification was the hardest part.

IV&V is intended to assist mitigating risk, hence the decision to do IV&V must be risk-based. NASA policy defines risk as the "combination of 1) the probability (qualitative or quantitative) that a program or project will experience an undesired event such as cost overrun, schedule slippage, safety mishap, or failure to achieve a needed breakthrough; and 2) the consequences, impact, or severity of the undesired event were it to occur [3]." The likelihood of occurrence and consequences of a given software failure cannot be calculated early in the software life cycle. However, there are realistic metrics available that give good general approximations of the consequences as well as the likelihood of failures.

Probability Evaluation

The probability of failure for software is difficult to determine at any phase in the software development life cycle. NASA has identified factors that impact development difficulty. These factors were then calibrated to determine the extent of risk for successful software development. While the indicators are not precise and are currently in Beta testing by NASA software development projects and the IV&V determination team, they are available to provide estimates that are adequate for assessing IV&V need.[1] There are nine factors: software team complexity, contractor support, organization complexity, schedule pressure, process maturity of software provider, degree of innovation, level of integration, requirement maturity, and software lines of code.

Five risk categories within each factor were identified based on input from software developers from all NASA centers. Values 1, 2, 4, 8, and 16 were assigned to each category. Finally, a weighting factor of 1 or 2 was identified for each factor. This information is shown in Table 1

To apply the criteria, the project manager identifies the category of risk for each factor and multiplies the appropriate value (1, 2, 4, 8, or 16) times the weighting factor of 1 or 2. The sum for all factors yields an initial numerical representation of the project software development risk. For example, a project might rate the following:

- Software team complexity – "up to 20 people at one location" = 4 * 2 = 8
- Contractor support – "with minor tasks" = 2 * 2 = 4
- Organizational complexity – "two locations but with same reporting chain" = 2 * 1 = 2
- Schedule pressure – "non-negotiable" = 16 * 2 = 32
- Process maturity – " CMM Level 1 but with a successful history" = 8 * 2 = 16
- Innovation – "between proven but new and cutting edge" = 8 * 1 = 8
- Integration – "almost stand alone" = 2 * 2 = 4
- Requirement maturity – "preliminary objectives" = 8 * 2 = 16
- Lines of code – "~ 300K" = 2 * 2 = 4
- TOTAL = 8+4+2+32+16+8+4 + 16+4= 94

This risk information must now be combined with the consequence evaluation.

Factors contributing to probability of software failure	Un-weighted probability of failure score					Weighting Factor	Likely-hood of failure rating
	1	2	4	8	16		
Software team complexity	Up to 5 people at one location	Up to 10 people at one location	Up to 20 people at one location or 10 people with external support	Up to 50 people at one location or 20 people with external support	More than 50 people at one location or 20 people with external support	X2	
Contractor Support	None	Contractor with minor tasks		Contractor with major tasks	Contractor with major tasks critical to project success	X2	
Organization Complexity*	One location	Two locations but same reporting chain	Multiple locations but same reporting chain	Multiple providers with prime sub relationship	Multiple providers with associate relationship	X1	
Schedule Pressure*	No deadline		Deadline is negotiable		Non-negotiable deadline	X2	
Process Maturity of Software Provider	Independent assessment of Capability Maturity Model (CMM) Level 4, 5	Independent assessment of CMM Level 3	Independent assessment of CMM Level 2	CMM Level 1 with record of repeated mission success	CMM Level 1 or equivalent	X2	
Degree of Innovation	Proven and accepted		Proven but new to the development organization		Cutting edge	X1	
Level of Integration	Simple - Stand alone				Extensive Integration Required	X2	
Requirement Maturity	Well defined objectives - No unknowns	Well defined objectives - Few unknowns		Preliminary objectives	Changing, ambiguous, or untestable objectives	X2	
Software Lines of Code*	Less than 50K		Over 500K		Over 1000K	X2	
Total							

Table 1: *Likelihood of Failure Based on Software Environment* (* indicates additional information is also required)

Consequence Evaluation

In general, the consequences of a software failure can be derived from the purpose of the software: i.e., what does the software control; what do we depend on it to do? NASA has many types of software, including flight software that is launched and contains mission critical functionality, ground system software that sends commands, scientific software for the experiments, and just about all other types of software imaginable. There are factors that can be used to categorize software based on its intended function as well as the level of effort expended to produce it: potential for loss of life, potential for serious injury, potential for catastrophic mission failure, potential for partial mission failure, potential for loss of equipment, potential for waste of software resource investment, potential for adverse visibility, and potential effect on routine operations.

Now the potential for failure had to be quantified. Four ratings were chosen: Grave, Substantial, Marginal and Insignificant. Each of the factors above were quantified for each rating. If *any* of the conditions are met, the software is considered to reside in that category. For example, the category Grave is defined as follows:

- Potential for loss of life - Yes.
- Potential for loss of equipment – Greater than $100,000,000.
- Potential for waste of resource investment – Greater than 200 work years on software.
- Potential for adverse visibility – International.

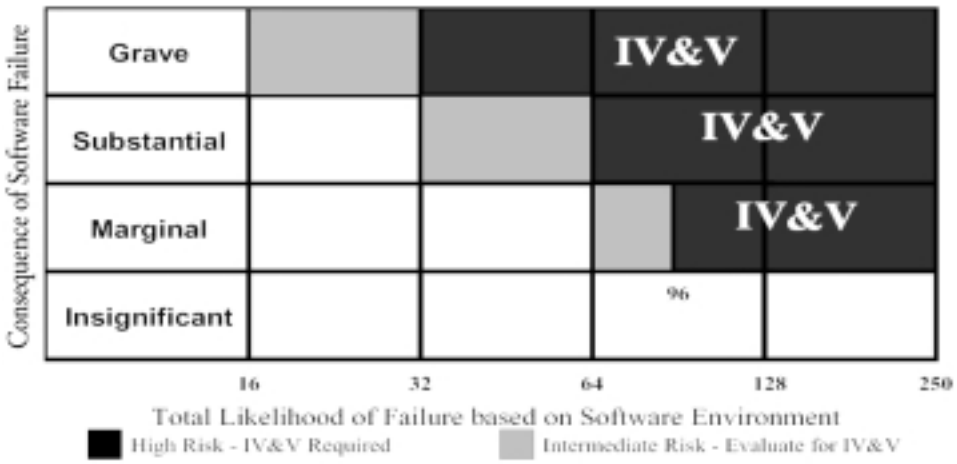
Combining the results of the "likelihood of failure rating" and the "consequences of failure" yields a risk assessment that can be used to identify the need for IV&V. Applying these criteria only determines that a project is an IV&V candidate – not the level of IV&V nor the resources associated with the IV&V effort. These must be determined as a result of discussions between the project and the IV&V Facility. Figure 1 shows a dark region of high risk where software consequences, likelihood of failure, or both are high. Projects having software that falls into this high-risk area shall undergo IV&V. The exception is those projects that have already done hardware/software integration. An IV&V would not be productive that late in the development cycle. The gray regions represent projects with intermediate risk. Projects having software that falls into these areas shall undergo an evaluation to determine if IV&V is warranted. Using the previous project example, a probability of 94 score means that they must perform IV&V if the consequence of failure is Substantial or Grave. Since the project's consequence was determined to be Grave, they must perform IV&V under the NASA IV&V Facility.

Step 3: Project Implementation

Prior to implementing IV&V, a company has to know about the software it develops. At NASA, software development is done at all 10 centers spread from California to Florida, Texas to Ohio, and Alabama to Maryland, and with universities and industries. In total, criteria data were received on approximately 100 projects. After discussions with projects to clarify the information and correct erroneous data, approximately 70 projects were identified as IV&V candidates.

Applying IV&V to 70 projects immediately, however, is impossible. The facility currently does not have the resources to accommodate this many projects and this

Figure 1: *Pre-Ship Test Risk Exposure*



much work at this time. Using the data from the criteria (consequence and probability) and discussions with the project managers, the following guidelines to focus the IV&V efforts were implemented:

- All projects currently receiving IV&V will continue.
- All projects classified Grave should be addressed first for IV&V with the highest priority applied to those closest to operational date as a general rule, with some attention applied to why it is classified as Grave.
- All projects classified as Substantial and needing IV&V based on the risk probability value greater than 32 that are in the requirements or design phase.
- All remaining projects classified as Substantial and needing IV&V based on the risk probability value greater than 32.
- All remaining projects classified as Marginal and needing IV&V based on the risk probability value greater than 96, prioritized based upon how close to starting they are.

Results

Applying IV&V on NASA projects has shown some very positive results and prevented costly errors. In one project, the IV&V activity identified design flaws in the command-and-control system that, if not corrected, would have resulted in a catastrophic hazard. This critical piece of software sends commands to hardware elements, and if not working properly, could fail to send emergency response commands leading to the loss of attitude control, rapid depressurization, and other hazardous conditions. Using a code analysis, IV&V identified an error that would eliminate the vital command link between the ground control system and the satellite. A special software patch was generated for the on-orbit software to correct the problem. IV&V developed policy criteria used by one program for non-flight software in integrated tests. This policy was key for insuring testing integrity while making it possible to keep tight development schedules. IV&V activities have benefited many different domains of NASA's software. In the manned-space flight domain, more than 4,000 problems were identified, 10 of the highest criticality, those that could result in loss of mission or loss of life. For experimental flight vehicles, IV&V identified more than 300 requirements and design problems. For ground systems, more than 250 legacy system requirements and mitigation problems were identified.These are just some of the benefits NASA has reaped from the formally applying IV&V. We currently are working on the return-on-investment calculations for these activities.

Conclusion

The results of NASA's investigation have shown that in this environment, IV&V has been cost effective. Companies large and small, in today's competitive world cannot afford software that is unreliable. To be effective, however, IV&V must be applied effectively, and the independence must not be lost. Project managers must understand the benefits above the cost, looking at the whole development, not just the current state.

NASA has recognized the value of IV&V and has taken steps to implement IV&V on all software projects where warranted. The decision to implement IV&V is no longer solely the decision of the project manager, but through an independent evaluation, the risk of the project is evaluated, and the need for IV&V is determined. Although the policy and criteria in this paper were written for NASA projects, they are applicable with minor modification to any software development.◆

Acknowledgements

Much of the work presented in this paper was a combined effort of NASA IV&V Facility personnel (Judy Bruner, John Hinkle, and Ken McGill), Goddard Space Flight Center personnel (Charles Vanek, John Dalton, Linda Rosenberg), and the members of the Software Working Group under Pat Schuller, Langley Research Center.

References

1. IEEE 610.12-1990, *IEEE Standard Glossary of Software Engineering Terminology*, Institute of Electrical and Electronics Engineers, Inc.
2. NASA IV&V Business Plan, Office of System Safety and Mission Assurance, NASA Goddard Space Flight Center, MD , June 2000, ivvplan.gsfc.nasa.gov
3. NASA Policy Guideline (NPG) 7120.5, NASA Program and Project Management Processes and Requirements Highlight Code, Office of Chief Engineer, NASA Headquarters, Washington, DC.

Note

1. Initial beta testing results indicate that some values and weights need to be adjusted. This will be done in the next release this year.

About the Author

Linda Rosenberg, Ph.D., serves as the chief scientist for Software Assurance at NASA's Goddard Space Flight Center, and is the former division chief of the Software Assurance Technology Office (SATO). Dr. Rosenberg is a recognized international expert in the areas of software assurance, software metrics, requirements and reliability. She serves on the Institute of Electrical and Electronics Engineers program committees for software reliability, software metrics, and software requirements. She is also an adjunct professor at the University of Maryland, Baltimore. Dr. Rosenberg holds a Ph.D. in computer science from the University of Maryland, a M.E.S. in computer science, and a batchelor's in mathematics.

Software Assurance Technology Office, Code 304
Goddard Space Flight Center, NASA
Greenbelt, MD 20771 USA
Voice: 301-286-0087
Fax: 301-286-1667
E-mail: Linda.Rosenberg@gsfc.nasa.gov

"Computers can figure out all kinds of problems, except the things in the world that just don't add up."

James Magary

"When we write programs that learn, it turns out that we do and they don't."

Alan J. Perlis