Understanding IV&V in a Safety Critical and Complex Evolutionary Environment: The NASA Space Shuttle Program[1]

Marvin V. Zelkowitz
Fraunhofer Center for Experimental
Software Engineering, Maryland
and University of Maryland,
College Park, Maryland, USA
+1-301-403-8935
mvz@fc-md.umd.edu

Ioana Rus
Fraunhofer Center for Experimental
Software Engineering, Maryland
4321 Hartwick Road, Suite 500
College Park, Maryland 20740, USA
+1-301-403-8971
irus@fc-md.umd.edu

ABSTRACT

The National Aeronautics and Space Administration is an internationally recognized leader in space science and exploration. NASA recognizes the inherent risk associated with space exploration; however, NASA makes every reasonable effort to minimize that risk. To that end for the Space Shuttle program NASA instituted a software independent verification and validation (IV&V) process in 1988 to ensure that the Shuttle and its crew are not exposed to any unnecessary risks. Using data provided by both the Shuttle software developer and the IV&V contractor, in this paper we describe the overall IV&V process as used on the Space Shuttle program and provide an analysis of the use of metrics to document and control this process. Our findings reaffirm the value of IV&V and show the impact IV&V has on multiple releases of a large complex software system.

Keywords

Evolutionary software, Life and mission critical software, Software independent verification and validation, Metrics, Process characterization, Space Shuttle program, Software safety and reliability

1 INTRODUCTION

"Defect free software" is a highly sought goal, and the use of an independent group to provide verification and validation (IV&V) on a software system is often cited as a means to ensure a high quality software product.

However, this is an elusive goal.

We present here an overview of a multi-level complex process for the NASA Space Shuttle software IV&V. The major results of our study show that:

- In spite of the complexity of the software being produced, the resulting product is effective and safe.
- IV&V is able to provide an independent means to certify flight readiness of this software.
- Yet, in spite of the care in developing such software, defects are still an unavoidable consequence of today's software development process.

We learned that for the Space Shuttle program IV&V does not follow the common model, where an independent group takes the artifacts developed by another group and applies verification and validation (V&V) activities to them. It is a more complex process, where "independence" is more loosely defined and it is manifested in some, but not all aspects of the process. The activities performed by the IV&V contractor span across the whole lifecycle and are not limited to just product verification and validation; they also include risk analysis, requirements analysis, issues tracking, and process evaluation. In our initial attempt to develop an economic model for the return on investment for IV&V activities, we discovered that many of the benefits are qualitative, and therefore cannot be measured and expressed in dollar figures.

The NASA IV&V program for the Space Shuttle was instituted in 1988 and in 1997 management for IV&V was transferred to the NASA/IV&V facility in Fairmont, WV. The NASA Center Initiative which funds this research, began in 1999 as a comprehensive look at understanding the economic impact that the IV&V process has had on the Shuttle program.

In Section 2 of this paper we look at the classical definition of IV&V and the existing models that implement IV&V. Then in Section 3 we present the process model used for the Shuttle, given the specifics of the system, software, and development environment and constraints. We will discuss the purpose of IV&V, the roles, activities, and interactions with the development environment. The analysis in Section 4 shows that IV&V proved to be successful and beneficial in the context of this program.

[1] This work has been performed as NASA Subcontract No. 93-393B-FUSA from the NASA/IVV facility in Fairmont, WV to the Fraunhofer Center, Maryland.

0-7695-1050-7/01 $10.00 © 2001 IEEE

2 SOFTWARE INDEPENDENT VERIFICATION AND VALIDATION PROCESS TYPES

The IEEE Standard for Software Verification and Validation [9] identifies three parameters that define the *independence* of IV&V: technical, managerial, and financial. Depending on the independence along these three dimensions, there are many forms of IV&V, most prevalent being: classical, modified, internal, and embedded. *Classical* independence embodies all three parameters. *Modified* preserves technical and financial independence, with some compromise on managerial independence. *Internal* and *embedded* IV&V are performed by personnel from the developer's organization. Therefore all three independence aspects are compromised, the difference between the two being that for *internal*, the IV&V team reports to a different management level than does the development team.

According to the definition given by the NASA Safety and Mission Quality Office, IV&V is "a process whereby the products of the software development life cycle phases are independently reviewed, verified, and validated by an organization that is neither the developer nor the acquirer of the software, IV&V differs from V&V only in that it is performed by an independent organization." [10]

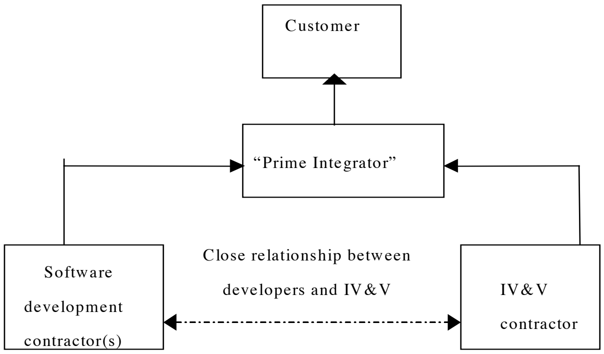
Customer
"Prime Integrator"
Software development contractor(s)
Close relationship between developers and IV&V
IV&V contractor

Figure 1. Modified IV&V

For the Space Shuttle software, IV&V is a *modified* type. Figure 1 shows the *modified* model of IV&V. The prime integrator (i.e., NASA) manages the entire software development. Development and IV&V are performed by separate companies that report to the prime integrator at the same level. The IV&V personnel is collocated with the developers and they have both informal and formal communication.

For the Shuttle software there is one more level of IV&V, an *internal* IV&V, used by the development contractors, who have their V&V groups separate and managerially independent from the development groups. In the current study we do not address this aspect. Our goal is to understand the impact that financially-independent IV&V has on Shuttle software development.

3 IV&V FOR SPACE SHUTTLE SOFTWARE

Space Shuttle Software Characteristics

The NASA Space Shuttle program uses four orbiter spacecraft. Software releases, called operational increments (OIs), are used for repeated missions on all four orbiters. There have been over 22 operational increments developed between 1981 and 1999.

Software OIs enjoy reuse across all four orbiters as well as repeated use for each orbiter. The core functionality of Shuttle software (common for all OIs) consists of 765 software modules with a total of 450K DSLOC (Delivered Source Line of Code). Each new release requires on average 18K DSLOC in modified mission-specific functionality and 26K DSLOC of new or modified core functionality. This represents an average of approximately 4% of new or modified system code (core functionality) with each release, thus providing for a stable base software system [1] [7] [8].

This is not a simple example of staged product evolution, where each new version of the product completely replaces the previous version. Rather there is a base system of core functionality that is reused and enhanced by extensions that differ from mission to mission. The Shuttle software could be viewed as a horizontal product line as it primarily enjoys forward interoperability of the software, but has been also applied with backward interoperability on a limited basis (e.g., an earlier increment could be used instead of a newer one in a coming mission).

Mission safety and reliability are the most important criteria for all missions and for each new software release. Because of this, changes to either the software or hardware are made with great care, such that they do not alter the achieved safety and the architectural integrity of the system.

The software is written in High-order Software Language for Shuttle (HAL/S), and executes on legacy hardware with limited memory: General Purpose Computers (GPCs) with a semiconductor memory of 256K 32-bit words. For each OI, new functionality is carefully weighed against the memory requirements of the existing functionality.

The Shuttle has two main flight control subsystems: the Primary Avionics Software System (PASS) and the Back-up Flight System (BFS), which provides backup capabilities for the critical phases of a mission. PASS and BFS have been developed independently by different contractors. A third contractor built the Space Shuttle Main Engine Controller (SSMEC), but that system was outside of the scope of our study.

The Shuttle uses five on-board computers - four running the PASS software for redundancy and one running the BFS version. However, the n-version risk mitigation strategy does not work as well for software failures as with hardware failures since a software failure in one computer is likely to fail in the others as well (e.g., the Ariane 5 failure in 1996 [4]).

In this complex environment, IV&V acts as a pair of extra eyes, to objectively ensure that the required functionality is implemented, given inherent hardware constraints, with minimum risk, preserving the architectural integrity and safety of this life and mission critical software.

Shuttle Development Process

NASA uses a complex development process, with numerous verification checks, to assure reliable development of each new OI. For the purposes of this paper, this is briefly described in Figure 2. More complete descriptions of this process are given in [2] and [3]. Briefly, the overall process is as follows:

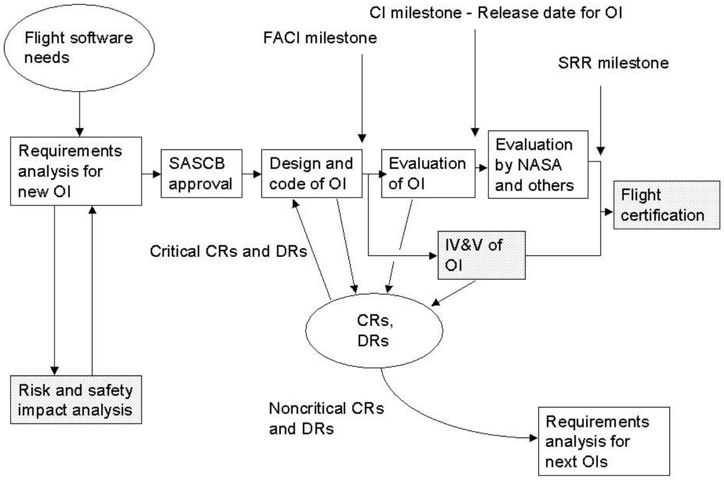

Figure 2. Overview of Shuttle software development

In the figure, rectangles represent the various processes for building a new OI, whereas ovals represent the main data that tracks the development. A fourth data item, the Issues Tracking Report, is used by the IV&V contractor to monitor this development and will be discussed later. The shaded rectangles refer to the major IV&V activities.

- Flight Software Needs come from NASA headquarters, the flight software community and from issues raised by the development and IV&V teams on previous OIs. Each new requirement generally has a champion, who is seeking to add or change that functionality to the existing software base.
- The flight software community, including the IV&V contractor, perform an analysis and a risk assessment on the new requirements such that a set of requirements for a new software release is developed. Multiple releases (i.e., operational increments) are often under consideration at one time, and proposed changes are often strung out across several OIs to minimize disruption of the software and schedule changes to be most advantageous to NASA.
- The Shuttle Avionics Software Control Board (SASCB) approves these requirements and a new operational increment is scheduled.
- The developer of the Shuttle software uses these requirements and upgrades a previous Shuttle software release to meet the new requirements. This typically takes about 8 months for initial development. Defects (e.g., Discrepancy Reports [DRs] and Change Requests [CRs]) detected by the developer are tracked once each new module becomes part of a software build. If critical, the CR or DR is implemented as part of the current OI; if not critical it is added to the list of new requirements to be evaluated for a further OI.
- After the developer adds all new functionality to the base software, the milestone called the FACI (First Article Configuration Inspection) is reached. The developer performs verification and validation testing, and the IV&V contractor begins to analyze these tests.
- At the Configuration Inspection (CI) milestone the software is released, where it undergoes further evaluation before the software is ready for use on a mission. At this time additional performance and functionality testing occurs by NASA, the software developer, and other groups to assure that all performance, reliability, and safety criteria are met. The CI milestone is called the *release date* for the software, even though the process can take another year before the software actually flies on the shuttle.
- After passing these evaluation criteria, the software is certified for flight on the planned Shuttle mission at the Software Readiness Review (SRR) milestone.

A new OI is released about once a year. Since a single OI can take up to 28 months to build, several OIs are under simultaneous development, and the IV&V process needs to keep track of potential problems that can cross OI boundaries. This is significant as CRs and DRs are intentionally delayed for implementation across multiple releases until a more advantageous time.

Shuttle IV&V Process

Depending on the NASA program goals, there are different goals for IV&V as well. For Shuttle the main objectives are safety, reliability, and mission completion. Therefore, the Shuttle IV&V program has four major goals:

1. Demonstrate the technical correctness of critical flight software, including safety and security concerns;
2. Assess the overall quality of the system and software products;
3. Ensure compliance with the development process standards;

4. Provide written evidence and traceability of this correctness so that software can be certified as flight ready.

An IV&V process requires that the evaluation group (the IV&V team) have technical, financial and managerial independence from the development group. Within the Shuttle program this is accomplished as follows:

1. **Technical** – IV&V prioritizes its own efforts and has its own (proprietary) set of analysis tools to determine which components to study;
2. **Financial** – The IV&V budget is independent from the developer's budget, although both are part of the overall Shuttle program budget.
3. **Managerial** – IV&V is performed by a different organization from the development organization. There is an independent reporting route to NASA program management. The IV&V independently decides:
   - Which areas of the system to analyze
   - What techniques to use for IV&V
   - The schedule of IV&V activities to be performed

The three shaded areas in Figure 2 represent IV&V activities. These activities occur during three phases in the development process:

- **Requirement analysis**: Risk analysis and risk reduction activities such as Hazard analysis, and Change impact analysis for safety, hardware and development resources lead to problem detection in the early development phases. The IV&V team represents a historical record (in terms of previous issues raised from earlier OIs) in judging the impact of any proposed change.
- **Product evaluation**: The IV&V team analyzes the implemented code and evaluates the tests conducted by the developer and proposes changes where warranted. This may involve informal negotiations in resolving issues, uncovering new issues that need to be resolved, and formal decision making. The IV&V team generally does not test the software, although it does in certain situations. Most of its activity is in evaluating the results of the developer's own testing process.

- **Flight certification**: IV&V has to sign-off at the end of the process to ensure traceability and disposition of all critical issues that were uncovered during development of that OI.

Process-oriented IV&V in any phase means process audit and improvement. The IV&V contractor interacts with Shuttle software development in four distinct ways:

1. **Issues tracking.** All open issues generated by IV&V are tracked by the Issue Tracking Reports (ITRs) discussed below.
2. **Flight software readiness assessments**, which evaluate critical software changes prior to the flight readiness review. The IV&V Certification of Flight Readiness statement is integral to the Shuttle Program's process for verifying that the upcoming mission can be safely and successfully executed.
3. **Special studies** investigate specific core functionality changes to the flight software (e.g. the Global Positioning System (GPS) receiver/processor, the Multifunctional Electronic Display System (MEDS) - "glass cockpit").
4. **Facilitates channels of communication** with the NASA Office of Safety and Mission Assurance (OSMA) by providing copies of status information and IV&V presentations.

Due to cost limitations, not all Shuttle software is subject to IV&V. Subsystems deemed mission critical are candidates for IV&V, and there are varying levels of IV&V effort for these subsystems. The IV&V contractor concentrates on software used during the most critical phases of flight, e.g., ascent and descent. Depending upon the criticality and risk of software changes that have been made, and the allocated budget and available resources, the contractor determines the level of IV&V effort needed on a component [5]; *comprehensive* (eleven activities), *focused* (a subset of nine out of the eleven activities), or *limited* (five out of the eleven activities).

As part of this evaluation process, IV&V issues are tracked by the IV&V contractor in both informal interactions with the developer and in a series of formal reports called Issues Tracking Reports (ITRs). During requirements analysis for an OI and thereafter (Figure 2), the ITRs are used by IV&V to track any further IV&V issues. Issue Tracking Reports (ITRs) are IV&V contractor documents for keeping track of all the actual and potential issues (anomalies) associated with any CRs and DRs within the scope of IV&V, across OIs. By tracking their progress, and certifying their disposition, the IV&V contractor provides a mechanism for NASA to certify the OI as being safe and flight ready. At the end of each OI all CRs and DRs are reviewed to ensure that there are no open issues relevant to safety and also that the changes did not activate issues in dormant code. This has proved to be a successful mechanism for allowing software to evolve safely across OIs. From 1988 through mid-1999 there were almost 800 such ITRs.

Once discovered, an issue is tracked until it is resolved and the ITR is closed. Issues can be dispositioned in several ways:

- After a discussion between the developer and the IV&V team, the issue is deemed not to be an error and the ITR is closed with no subsequent action. In some cases the source code implements a correct, but different, algorithm than what has been specified, and a decision is made to accept what has been developed.

- If the problem is serious (e.g., mission safety is at risk), a discrepancy report (DR) is created. At this point the ITR is closed and the developer's DR tracking mechanism assures that the problem will be fixed.
- For a relatively minor error that will not affect the safety of the current mission, a change request (CR) is generated. CRs will be scheduled for implementation for a subsequent OI. This represents almost half of the ITRs that have been generated. With multiple OIs under concurrent development, an ITR will often cause a change to the requirements of the following OI in the schedule. Such changes are delayed until a later OI because of the danger of spurious changes. Since all such changes need extensive testing for all changed modules, such non-critical changes are not made until needed at a later date.

Approximately one third of the ITRs represent documentation errors, i.e., the implemented software and the documentation do not agree. As with minor errors, documentation changes are not made to the software until the module in question is later changed due to new functional requirements. In such cases the ITR is kept open until the module is later modified.

4 CHARACTERIZING SHUTTLE IV&V DATA

Much of the value of IV&V resides in the various risk mitigation tasks performed during the *definition* phase of an OI. The first issue tracking report is dated 1988, and by mid-1999 a total of 777 ITRs were collected. The IV&V team rates ITRs in severity from 1 to 5 with the following meaning:

Severity 1. A problem can cause loss of control, explosion, or other hazardous effect.

Severity 2. A problem can cause inability to achieve mission objectives, e.g., launch, mission duration, payload deployment.

Severity 3. A problem is visible to the user (crew), which is not a safety or mission issue. It is usually waived and a CR for a later OI is opened.

Severity 4. A problem is not visible to the user (crew). It is an insignificant violation of the requirements. This includes documentation and paperwork errors (e.g. typo's), intent of requirements met, insignificant waivers.

Severity 5. An issue is not visible to user (crew) and is not a flight, training, simulation or ground issue. This includes programming standards, maintenance issues, and philosophical issues (e.g. improper HAL/S parameter name prefix, inefficient code that meets requirements).

Severity 1 and 2 ITRs are the most critical and need to be addressed during the development of that OI. Severity 3 ITRs, if workarounds are possible, are often resolved as change requests (CRs) for a later OI and are not changed or are documented as *user notes*. Many severity 3 ITRs represent issues that are not safety related that are present in that OI or could appear in a later OI. CRs are written to ensure that the later OI does not develop any problems. Severity 4 and 5 ITRs are generally CRs that will be corrected when the appropriate documents are updated.

Some of the severity 1 and 2 ITRs represent issues that are outside of the operating environment, of the software so cannot occur, even though theoretically possible. That is, the condition that can cause the software failure cannot occur in practice. Such errors are classified as 1N or 2N and are generally grouped with the severity 3 errors.

A Characterization of the ITRs

Severity	1	2	1N,2N,3	4	5	Total
PASS	7	6	85	219	142	**459**
Both	3	0	13	43	20	**79**
BFS	8	1	41	115	70	**235**
Unknown	0	0	0	3	1	**4**
SUM	**18**	**7**	**139**	**380**	**233**	**777**

Table 1. Summary of ITRs collected

Table 1 summarizes the set of ITRs that have been collected**.** *Both* refers to issues that affect both the BFS and PASS software. For four severity 4 and 5 ITRs it was difficult to determine which system they affected. Although the PASS was the primary avionics system (538 ITRs or 69.2%), 314 ITRs (40.4% of the ITRs) concern the BFS. About 10% concern both systems.

Severity	1	2	1N,2N,3	4	5	Total
OPEN	**2**	**0**	**59**	**138**	**113**	**312**
PASS or both	0	0	44	104	84	232
BFS	2	0	15	34	29	80
CLOSED	**16**	**7**	**80**	**239**	**119**	**461**
PASS or both	10	6	54	158	78	306
BFS	6	1	26	81	41	155

Table 2. Open and closed ITRs

As shown in Table 2, of the 773 ITRs for which we have a disposition[2], 461 (59.6%) of the ITRs are closed and 312 (or 40.3%) of the ITRs are still open in mid-1999. Although 40% of the ITRs are still open, all of the severity 1 and 2 PASS ITRs are closed and only two of the severity 1 BFS ITRs were still open at the time of our initial analysis. In

[2] We do not have details on the four severity 4 and 5 "unknown" ITRs from Table 1

both open ITR cases, which date from the early 1990s, the BFS requirements differ from the PASS requirements in the instance of aborting a mission. Those were not specified in BFS requirements and so were implemented differently in both the PASS and BFS systems. The ITRs indicate a possible resolution to the problems, but the ITRs have not been marked closed by the time of our analysis. [We just found out from the IV&V leader that they have now been closed.]

For the following analysis we limited our study to the severity 1, 2, and 3 ITRs, since severity 4 and 5 ITRs are of lesser impact and do not affect mission performance. There were 164 severity 1 through 3 ITRs. We classified these ITRs into one of the following categories:

- **No change** – ITR was resolved as not being a defect. The ITR is closed with no corrective action. This happens when the description of a requirement, software module, or module test case is incomplete, but the artifact is developed correctly. However, the reason for no action is recorded.
- **Change** – A software defect was found and corrected. In some cases, a formal discrepancy report (DR) is created indicating a problem to be fixed by the developer. The ITR is closed when the code is changed if no DR was created.
- **CR** – A change to the software for later implementation is planned. This may involve creation of a change request (CR) document for later SASCB approval.
- **Process** – The ITR reflects verification and validation activity of the developer that is unclear, e.g., a certain condition in the requirements is not part of the given test case. In such cases either the test is performed or the developer explains that the condition is actually tested.

	No chg	Chg	CR	Process	Total
Severity 1	8	2	7	1	18
Severity 2	2	4	1	0	7
Sev. 1N,2N,3	54	19	47	19	139
Total	64	25	55	20	164

Table 3. Disposition of ITRs

In Table 3 we present the disposition of severity 1, 2, and 3 ITRs. Of the 164, 64 were closed with no action and 20 process ITRs required the developer to reconsider certain tests. Thus 64+20 = 84 of 164 ITRs or 51.2% of the ITRs represent no changes to the developed software, whereas 48.8% (80 ITRs) do reflect changes proposed by the IV&V process.

It is often stated that the earlier a defect is found, the easier it is to repair. Looking at the creation dates for each ITR sheds light on this. In Table 4 we divide the ITRs into those found during the requirements and development phases (Early ITRs) and those found after software release (CI milestone) (Late ITRs)[3]. More than half (83 of 159) of the issues were discovered prior to release of the new OI.

	Early ITRs	Late ITRs
Severity 1	9	8
Severity 2	6	1
Severity 1N, 2N,3	68	67
Total	83	76

Table 4. Early ITRs

Table 4 represents all ITRs, including those resolved with no change to the software. By looking at only the 80 ITRs from Table 3 that represent proposed changes (the CR and Chg columns), the data from Table 4 is reduced to that shown in Table 5. About 62% of the severity 1 and 2 issues (8 out of 13) were found during the requirements and development phases and 59% of all major defects (47 out of 80) were found during these early phases. In addition, five severity 1 defects were detected in the OI software after the CI milestone, thus representing a real risk of a later mission failure if the defects were not found.

	Early ITRs	Late ITRs
Severity 1	3	5
Severity 2	5	0
Severity 1N,2N,3	39	28
Total	47	33

Table 5. ITRs representing proposed changes

OI	1-2 E	1-2 L	3 E	3 L	1-3 E	1-3 L	DR E	DR L
I				2		2	32	13
J				1		1	139	52
K		3		2		5	110	25
L			2	2	2	2	133	32
M				1		1	114	10
N			2		2		81	11
O			5	5	5	5	89	28
P	1		6		7		60	17
Q	1		5	3	6	3	67	12
R		1	2	7	2	8	61	9
Total	**2**	**4**	**22**	**23**	**24**	**27**	**886**	**209**

Table 6. Defects found by OI

Table 6 gives the distribution of 51 severity 1 through 3 ITRs that were identified with a particular OI. We limited this table to those OIs that were fully evaluated by IV&V after 1988. Each pair of columns in the table represents the early (E) and late (L) ITRs for that severity level. The rightmost two columns in the table represent the 1095 early and late DRs (defects) found by the developer of the PASS subsystem for each OI.

[3] Five of the ITRs were not identified with a specific OI, so the phase of discovery could not be determined.

5 IV&V EFFECTIVENESS

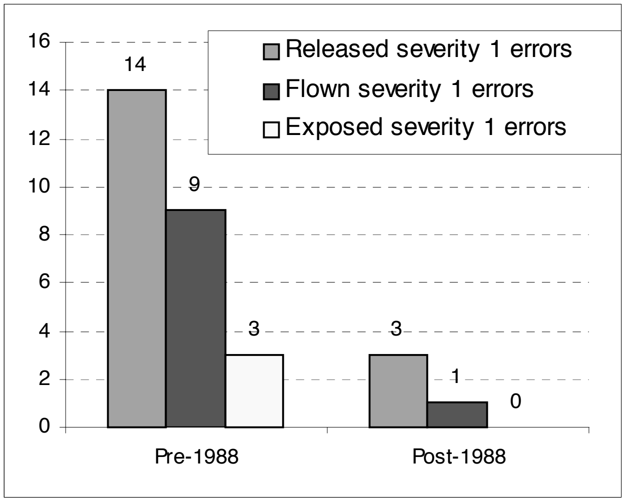

Figure 3. Severity 1 defects

The previous set of tables provides one set of quantified measures of the benefits of the IV&V process. However, we are interested in understanding the overall value of IV&V in this complex environment. A more operational measure of IV&V effectiveness is determining if any critical errors have been present on Shuttle missions. This is summarized in Figure 3.

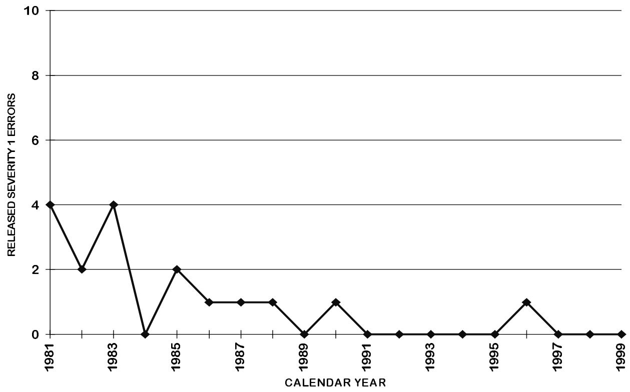

Figure 4. Released severity 1 errors by year

A total of 17 severity 1 errors have been found in released Shuttle software (as specified in Section 3, for the Shuttle program, *released* software dates from the CI milestone, that is over a year prior to launch). Of these, 10 defects have flown on missions. However, of these, 9 flew on pre-IV&V software, where 3 were in dormant code that did not execute. Since IV&V was instituted, only three severity 1 errors have been found on released software, and only one of these was on an actual mission. In this one case, it was in dormant code that did not execute. In Figure 4 we show the occurrence of severity 1 errors by year. The drop in such errors since 1990 is quite apparent from this figure.

A measure of defect detection that is often used needs to be reexamined in the light of the space Shuttle experience. The number of days a problem report remains open is a measure of the effectiveness of the defect detection process. From the 103 closed serious ITRs (severity 1 through 3) from Table 2, we computed the number of days that each remained open. This is displayed as Figure 5. The average number of days an ITR was open was 284, and ranged from 1 to 3049. The closure rate was fairly constant between 10 and 180 days. 24% of the ITRs were closed in 10 days or less (25 out of 103) and slightly more than 25% (29 out of 103) took more than 420 days.

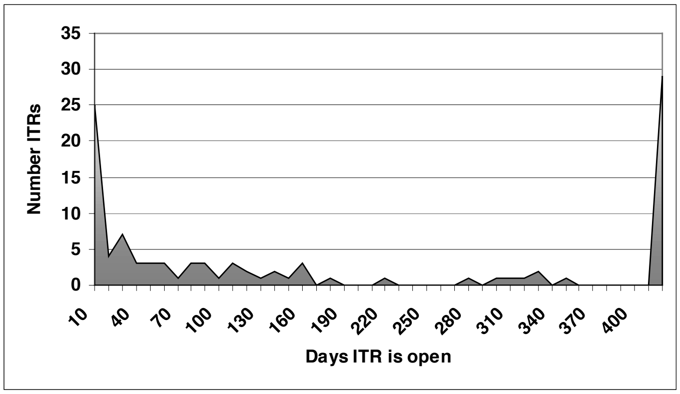

Figure 5. Days an ITR remained open

In a traditional development, an average days-open time of 284 days would be seen as a poor defect removal process. However, the goal of defect tracking in the Shuttle is to certify flight readiness of a future OI where for safety concerns changes are only made in software where necessary. A major benefit of the IV&V process is that the ITRs keep track of multiple issues over several operational increments. Non-critical issues may not be resolved for several years until that section of the software is modified. The number of days an ITR remains open is not of critical importance, but the fact that the process can keep track of all of these over a span of some 3000 days is vital to the success of the activity.

A final indirect measure of IV&V effectiveness can be implied by looking at the Early Detection rate (EDR) of each OI. The EDR is reported to NASA as a measure of how early all software defects are discovered for each OI. It is computed simply as (# Early errors) / (#Total number of errors). Thus it starts at 1.00 and drops below 1 as later testing errors are found.

NASA uses a date of 400 days (about 1 year) before release (CI milestone) as the date for early errors. Plotting the EDR for a typical OI results in a graph much like Figure 6.

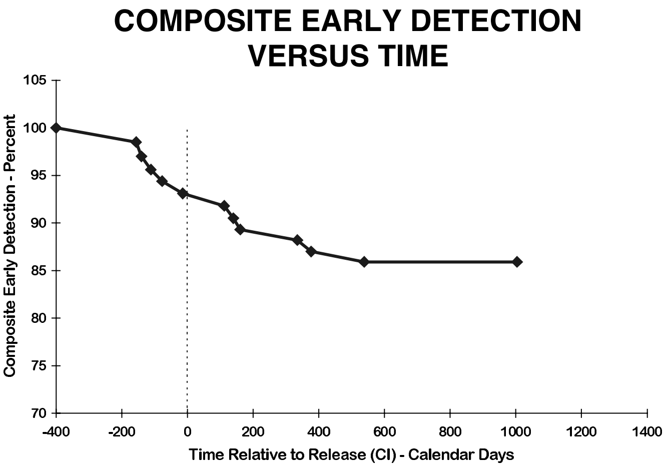

Figure 6. Early Detection Rate

The final value of the EDR for the OIs analyzed here are given by Table 7. The overall EDR rate for all OIs is .81 (Meaning 81% of all defects were found more than 400 days before software release). While the earlier four have EDRs of about .7 to .8, the last 4 have EDRs that ranged from about .75 to almost .9. While many things go into the error detection process, IV&V has had an impact.

OI	EDR
I	0.71
J	0.73
K	0.81
L	0.81
M	0.92
N	0.88
O	0.77
P	0.78
Q	0.86
R	0.87
Total	**0.81**

Table 7. Composite EDR for multiple OIs

6 CONCLUSIONS

In this paper we present an overview of NASA Space Shuttle software IV&V process and an analysis of the issues tracking reports (ITRs) produced during the IV&V process.

In our analysis we found that an ITR does not necessarily reflect a single issue in the development of that OI. Some represent a single discrepancy in one of the artifacts, whereas others may represent the results of an inspection reflecting 10 to 20 such issues. This study is only a first approximation in our study of IV&V for this program.

Our original goal was to develop an economic model on the value of IV&V in terms of a return on investment to NASA based upon the expenditures for the IV&V process. The annual budget for IV&V the last three years has been approximately $3-3.5 million, which is approximately 1/30 of the budget allocated annually for the complete software development and assurance process. In our attempt to develop this model for the return on investment of IV&V, we discovered that many of the benefits are qualitative and cannot be measured and expressed simply in dollar figures. Since many of the benefits of IV&V are in discovering issues early (e.g., 24 of 51 severity 1 through 3 errors from Table 6), it is hard to quantify money not spent on defect detection and correction at a later stage.

The value of the IV&V interaction with the rest of the software community and the domain expertise brought in by the IV&V personnel, manifested throughout the development of an OI and also across multiple OIs is hard to quantify. Not to mention that in case a critical defect manifests the mission could be compromised, or the orbiter lost. The cost of losing an orbiter is about $2 billion [3], not to mention the loss of life that cannot be estimated.

Therefore, our basic findings are the following:

- There is a demonstrated value of IV&V to the NASA Space Shuttle program. From Table 5 there were 80 serious issues which were found by the IV&V process on Shuttle software over a 10-year period, including 13 possible defects that if not resolved, could endanger a mission.
- Many of these defects were found during the requirements analysis phase. Almost one half (24 out of 51 from Table 6) of the serious issues were found during requirements and development. We also cannot determine the additional savings that resulted from the risk mitigation strategy employed during definition and requirements analysis that prevented defects from even surfacing. We can only guess at the quantitative savings that resulted from this early detection of defects.
- The IV&V process uncovered some additional shortcomings in the overall testing process. Many of the 29 ITRs (the 20 of Table 3 plus nine severity 4 ITRs) indicated insufficient testing of various modules. These could have resulted in additional defects on later OIs had they not been discovered.
- Only 1 severity 1 defect has flown on a Shuttle flight since IV&V was instituted, and that error was in dormant code that did not execute. This contrasts to 9 severity 1 errors on Shuttle flights prior to the IV&V process.

Final Comments

It is important to state that this analysis is not meant to be a criticism of either the developer or the IV&V team. The precise reading of documentation by the IV&V contractor led to numerous issues that could affect the current or future Shuttle missions. Many requirements problems were discovered via this route. The dual product and process evaluation by the developer and the IV&V team allows for increased safety and reliability of the product. It clearly shows in some cases (e.g., the 64 ITRs that were resolved with no changes) that a fresh look at the software (via the IV&V process) demonstrated that the documentation was unclear and a restatement of the specification resolved the issue. The independence of the 2 groups shows those different approaches to evaluating software leads to an increased defect discovery process.

It is clear from data, such as that presented in Figure 3, that elimination of all defects is still beyond the realm of current software practices. Although some defects still manage to slip by, NASA's IV&V process clearly shows a vast improvement in defect avoidance and the production of robust software since 1988. The NASA IV&V process is definitely complex, but appears to be thorough.

The question to be asked is how well can organizations do who do not have the resources of an agency like NASA? The danger of someone reading this paper is that they may deem IV&V as too complex or too expensive to install. The *real* danger is that they do not install such a process, and a correspondingly important system later fails.

This analysis is helpful to other organizations outsourcing software (especially critical software), where IV&V can balance stakeholders' interests, mitigate risks, improve communication and visibility, track changes and anomalies, and provide QA for both product and contractor's process.

Most IV&V processes have been organized around developing a correct system – from requirements to delivery. However, as this paper demonstrates, the NASA space Shuttle program is a multi-year ongoing development where IV&V is an integral part of a multi-release process. Understanding the interactions among developers and evaluation teams for such large complex systems is important for achieving reliability in such large critical systems in the future.

ACKNOWLEDGEMENTS

We would like to acknowledge the cooperation of the NASA IVV Center in Fairmont, WV, AverStar, Inc., and United Space Alliance for their support in providing the data that was used in this analysis.

REFERENCES

1. Eickelmann, Nancy, I. Rus, and M. Zelkowitz, Preliminary Case Study Findings of the Space Shuttle Software Evolution as a Product Line Process, ISAW-4 workshop at ICSE 2000, Limerick Ireland, June 2000.

2. Florac, William, A. Carlson, and J. Barnard, Statistical process control: Analyzing a space Shuttle onboard software process, *IEEE Software* (July, 2000) 97-106.

3. Leveson, Nancy et al., An Assessment of the Space Shuttle Flight Software Development Process, National Academy Press, Washington DC, 1993.

4. Lions, J.L. et al. Report by the Inquiry Board on the Ariane 5 Flight 501 Failure, <http://www.esrin.esa.it/htdocs/tidc/Press/Press96/ariane5rep.html>, 1996.

5. McCaugherty, Dan, The criticality and risk assessment (CARA) method, NASA Workshop on Risk Management, Farmington, PA, October, 1998.

6. NASA, Business plan for the effective utilization of independent verification and validation to reduce risk in NASA missions, NASA Goddard Space Flight Center, May 31, 2000.

7. Schneidewind, Norman F., How to evaluate legacy system maintenance, *IEEE Software,* (July 1998) 34-42.

8. Schneidewind, Norman F., Measuring and evaluating maintenance process using reliability, risk, and test metrics, *IEEE Trans. on Software Engineering* 25, No. 6, (1999) 769-781.

9. IEEE Standard for Software Verification and Validation, Std.1012-1998, Annex C.

10. NASA headquarters Safety and Mission Quality Office (Code Q) letter of 13 January 1992; *Clarification of NASA's Independent Verification and Validation (IV&V) Perspective*.

11. Zelkowitz, M. V., Yeh R. T., Hamlet R. G., Gannon J. D., Basili V. R., Software engineering practices in the United States and Japan, IEEE Computer 17, No. 6 (1984) 57-66.