The Role of Independent Verification and Validation in Maintaining a Safety Critical Evolutionary Software in a Complex Environment: The NASA Space Shuttle Program[1]

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

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

ABSTRACT

The National Aeronautics and Space Administration Space (NASA) Shuttle program is a multi-billion dollar activity scheduled to span over 40 years. Maintaining such software with requirements for high reliability and mission safety taxes current development methods. In this paper we present how Independent Verification and Validation (IV&V) activities have evolved in order to provide for these requirements. We also show how the IV&V activities for this program differ from those of more traditional software developments.

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

1 INTRODUCTION

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. The most common approach is to develop a product and then give it to an independent group, which will determine the correctness and reliability of the system. However, such an approach is suboptimal when multiple releases of a system are in concurrent development. If a traditional IV&V approach were used, then tracking issues across multiple releases would be almost impossible.

We present here an overview of the process used to ensure mission safety and reliability for NASA Space Shuttle software. The software has undergone over 22 releases since 1981 and is expected to evolve for at least another 20 years. Since releases occur on the order of once per year, and a release generally takes over two years to develop, multiple releases are almost always under concurrent development. We looked at the data collected by the Shuttle development organization and the Shuttle IV&V contractor in order to demonstrate that the process is under control and allows for concurrent developments, such as would occur in a product line architecture.

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/IVV facility in Fairmont, WV. The NASA Center Initiative which funded 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 software development process model used for the Shuttle, given the specifics of the system, software, and development environment and constraints. We then discuss the purpose of IV&V, and the roles, activities, and interactions with the development environment of IV&V for Shuttle development. The analysis in Section 3 shows how IV&V plays an important role in maintaining this system, being used across multiple releases.

2 SHUTTLE SOFTWARE DEVELOPMENT

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.

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

[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.

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 software subsystems: the Primary Avionics Software System (PASS) and 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. 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.

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. (See Table 1.) 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].

PASS Rel.	Total Modified KSLOC / OI	Modified functions KSLOC/OI	Modified core KSLOC/ OI	% Modified of Total KSLOC
I	55.7	29.4	26.3	12.4%
J	44.3	21.3	23.0	9.8%
K	44.0	34.4	9.6	9.8%
L	47.3	24.0	23.3	10.5%
M	50.7	10.4	40.3	11.3%
N	50.4	15.3	35.1	11.2%
O	21.9	7.3	14.6	4.9%
P	32.1	11.0	21.1	7.1%
Q	57.2	12.1	45.1	12.7%
Total	**403.6**	**165.2**	**238.4**	
Avg.	44.84	18.36	26.49	9.97%
StdDev	11.36	9.41	11.67	0.03

Table 1. Size of Modifications per OI

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).

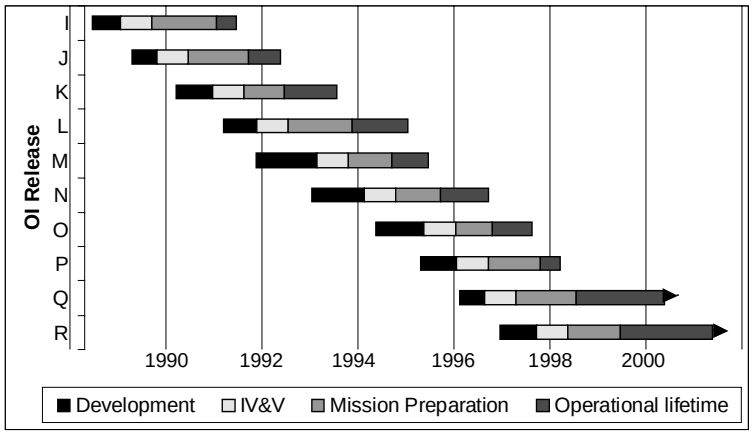


Figure 1. Lifecycle for each OI

Figure 1 shows the overlapping lifecycle for the 10 OIs completed since IV&V was instituted in 1988. Four phases are indicated for each OI: an initial development phase, a testing phase which includes an IV&V activity, a mission preparation phase which includes additional IV&V processes, and an operational lifetime executing on one or more Shuttle missions. As Figure 1 demonstrates, during most of this period, about 3 releases were in various phases of development (development, V&V or mission preparation), and up to four releases were active (either in execution or in development) during this period.

Because mission safety and reliability are the most important criteria for all missions and for each new software release, 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. As we later show, not-modifying source modules (because of the possible introduction of new defects) is a high priority requirement, which causes many non-critical changes to be delayed until absolutely necessary. Keeping track of these changes, as well as the underlying database to manage this process reliably, is at the heart of the IV&V process that has been developed.

Standard Models for IV&V

Verification and Validation (V&V) is a process common to all software development where the developer applies various processes (usually testing) to ensure that the new software agrees with its specification. Independent verification and validation (IV&V) is a V&V process where the V&V is performed by a group independent of the developer. The IEEE Standard for Software Verification and Validation [3] 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, while the managerial parameter is compromised. *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 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." [6]

An overall guiding principle in OI development is that changing any module, regardless of the reason, leaves the code open to error. Thus non-critical changes (e.g., a mistyped comment) are often not made until the module must be changed for other more important programmatic reasons. Thus pending changes often remain across multiple releases of the software. In fact, some changes, as we later show, have remained unresolved for over 3000 days (over 9 years).

Managing this set of pending changes over multiple releases is a critical issue management problem whose solution is needed to ensure reliability of the Shuttle code base. The issue tracking management is one of the important activities performed by the IV&V contractor. They use a tracking and reporting system, the Issue Tracking Reports (ITRs) as an eloquent mechanism for handling these pending changes. From 1988 through mid-1999 almost 800 such ITRs have been generated and are at the heart of the IV&V process for the Shuttle.

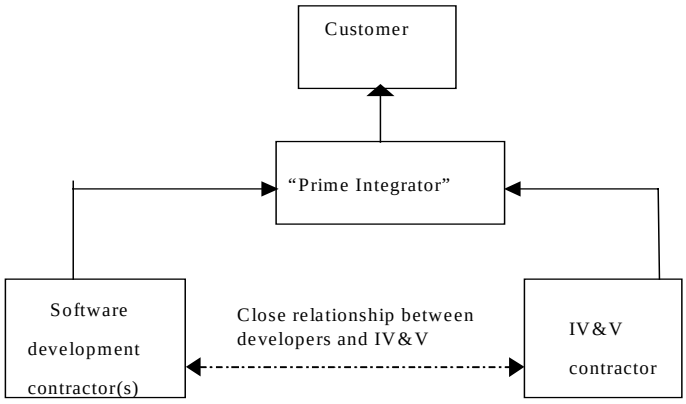


Figure 2. Modified Shuttle IV&V

For the Space Shuttle software, IV&V is a *modified* type. Figure 2 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.

Adapting IV&V for the Space Shuttle

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 3. More complete descriptions of this process are given in [2] and [4]. Briefly, the overall process is as follows:

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

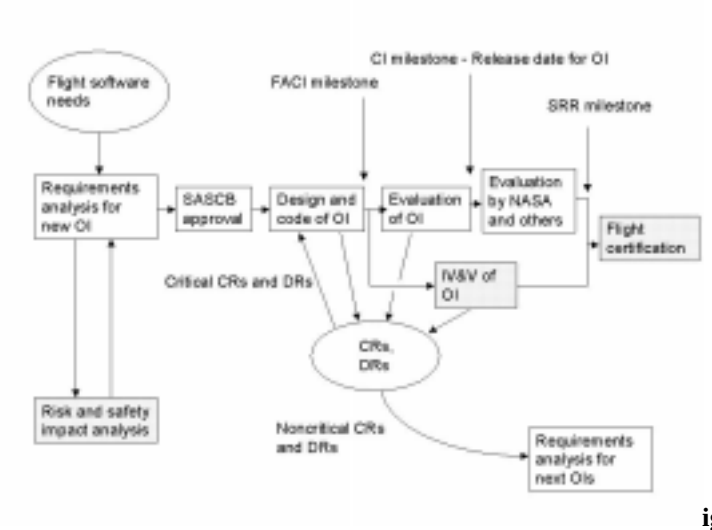
- After a discussion between the developer and the IV&V team, the issue is deemed not to be an error and the ITR is closed with no subsequent action. In some cases the source code implements a correct, but different, algorithm than what has been specified, and a decision is made to accept what has been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

Approximately one third of the ITRs represent documentation errors, e.g., the implemented software and the documentation do not agree. As with minor coding 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.

In Figure 3, rectangles represent the various processes for building a new OI, whereas ovals represent the main data that tracks the development. The shaded rectangles refer to the major IV&V activities.

- Flight Software Needs are identified by the flight software community.
- The flight software community, including the IV&V contractor, perform an analysis and a risk assessment on the needs, such that a set of requirements for a new software release is developed. As stated above, proposed changes are often strung out across several

OIs to minimize disruption of the software.



ig

Figure 3. Overview of Shuttle software development

- 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. Anomalies (i.e., Discrepancy Reports [DRs] and Change Requests [CRs]) are tracked once each new module becomes part of a software build. The key point at this stage is that CRs and DRs are tracked by the ITRs and become part of the traceability of defects across multiple OIs.
- After the developer adds all new functionality to the base software and makes the required corrections, the milestone called the First Article Configuration Inspection (FACI) is reached. At FACI the developers hand the product over to the independent V&V contractor and to developer's own V&V team. The development contractors have their V&V groups separate and managerially independent from the development groups. This is a more common form of IV&V, an *internal* IV&V according to [3].
- At the Configuration Inspection (CI) milestone, about 8 months later, the software is released to NASA, where it undergoes further evaluation before is ready for use on a mission. 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 mission preparation and undergoing more testing (e.g., operational testing), the software is certified for flight on the planned Shuttle mission at the Software Readiness Review (SRR) milestone, about 8 months after the previous CI milestone.

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

The three shaded areas in Figure 3 represent IV&V activities that extend across multiple OIs. These activities occur during three phases in the development process:

- **Requirements 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 an historical record (in terms of previous issues raised from earlier OIs) in judging the impact of any proposed change. This is also supported by the extensive domain expertise that the IV&V team members have.
- **Product evaluation**: The IV&V team analyzes the implemented code, evaluates the tests conducted by the developer, and proposes changes where warranted. 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**: At the end of an OI IV&V reviews all the DRs and CRs and certify that they were adequately implemented, corrected, and tested, that there are no issues relevant to safety that remained open, and there are no reactivated dormant code anomalies.

Ideally, IV&V would be performed on the entire system. In reality, due to budget and resource constraints, the IV&V contractor concentrates on software components used during the most critical phases of flight, e.g., ascent and descent. Other components could also occasionally be addressed if the program identifies them as critical issues. When IV&V was first instituted in 1988, there were 15 functions covered by IV&V; in 1992 the set was reduced to six functions (a subset of the initial 15). On these functions the IV&V effort may vary. The scope of IV&V can be: *comprehensive*, *focused*, or *limited* (limited is a subset of focused that is a subset of comprehensive). Limited addresses activities a-e, focused a-i, and comprehensive a-k in the following list:

a) Problem/change description
b) System impact analysis
c) Requirements analysis
d) Risk assessment
e) Disposition analysis

f) Code analysis
g) Level 6 and 7 test verification analysis
h) Documentation assessment
i) Safety assessment
j) Analysis of other system implementations
k) Complete test/verification analysis

The scope is determined by the criticality of the component, the risk and impact of the changes that have to be made to it, and the budget allocated. The contractor uses a tool named CARA [5] to help decide on the IV&V scope to be applied.

The IV&V contractor typically evaluates the CRs and DRs that are submitted to cover changes in the software. However they also often submit CRs and DRs themselves and use their specialized tools and expertise to perform a detailed evaluation of the software itself. Additional details of the shuttle IV&V process and an analysis of the data is given in [8].

3 MAINTAINING AN EVOLVING SOFTWARE SYSTEM

The ITRs tracked for the Shuttle are classified according to their criticality into one of the following five categories:

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).

From 1988 until mid-1999 about 780 ITRs have been entered in the issues tracking database. As Figure 4 demonstrates:

- Issues are found fairly uniformly across OIs, but
- The number of critical ITRs is quite low.

A total of 20 severity 1 and 2 ITRs were found attributable to these 10 OIs. As explained previously, many of the severity 3 through 5 ITRs are held until a later OI to avoid changing source programs needlessly.

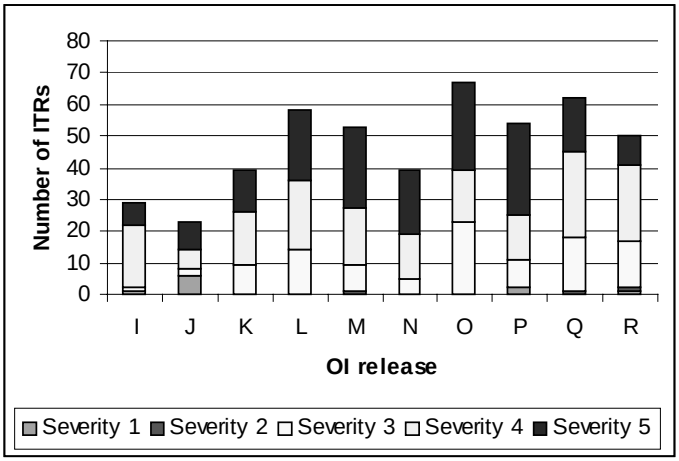


Figure 4. ITRs across OI releases

A measure of the complexity of issues tracking is the number of days an ITR remains open. The distribution of issues lifetime is given in Figure 5. Although most are handled within 40-60 days, many remain open for over a year.

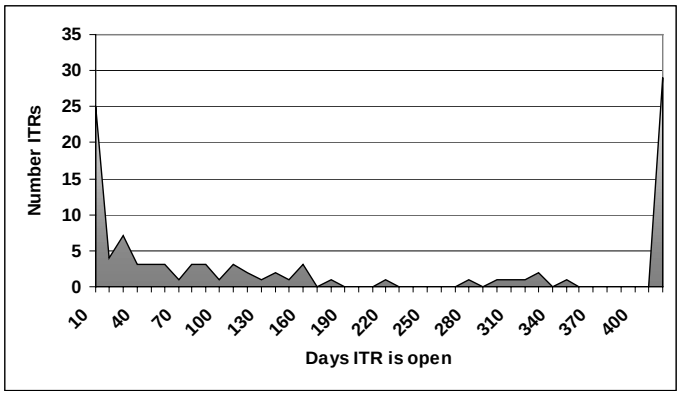


Figure 5. Days an ITR remained open

In figure 6, we reproduce the lifecycle given in Figure 1 with the lifetime of the long-lived severity 1 and 2 ITRs. Under each OI are the severity 1 and 2 ITRs attributed to that OI. Those that precede Release I were attributed to OIs that precede the introduction of IV&V in 1988.

Figure 6 indicates that 22 ITRs (18 severity 1 and four severity 2) are attributed to these releases. Note that 10 were found in code added to OIs that precede release I with three of the ITRs remaining open for up to 10 years. However, with the introduction of IV&V for OI I, only 12 were found, and none remained open for more than one additional OI.

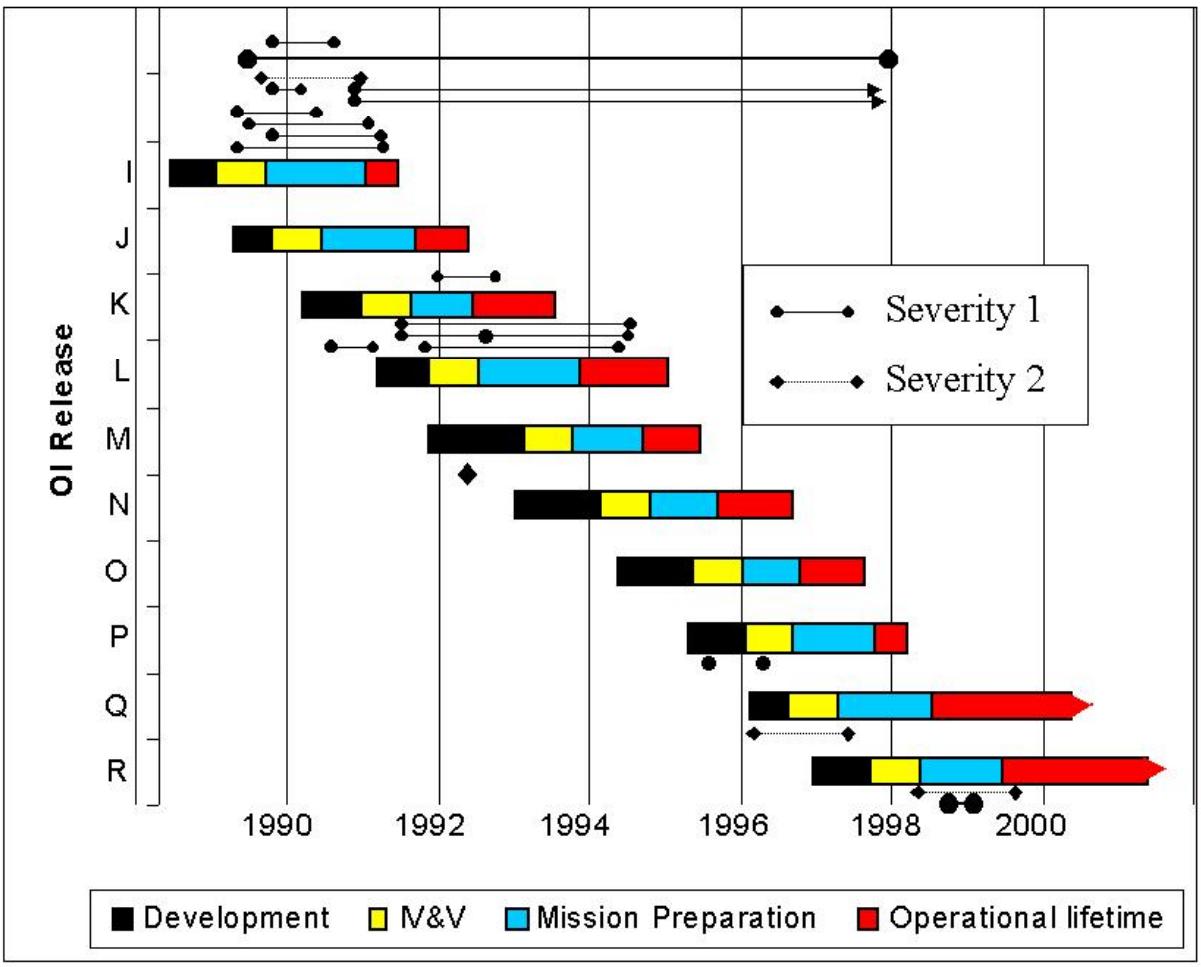


Figure 6. Severity 1 and 2 ITRs

Severity	Open ITRs	Closed ITRs
1	2	16
2	0	4
3	59	75
4	141	239
5	117	120
Total	319	454

Table 2. ITR summary

Note that an ITR doesn't necessarily indicate an error. Issues are often resolved with no changes. In the case of severity 1 ITRs, as Figure 7 indicates, only 1 severity ITR has ever flown on the shuttle since the introduction of IV&V, and that ITR was in dormant code.

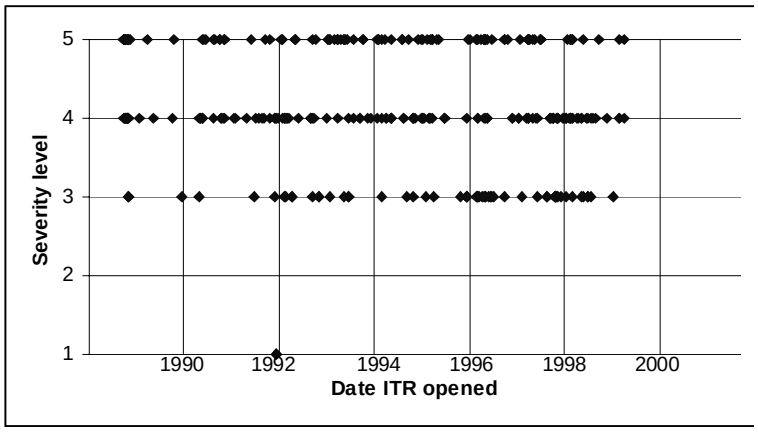


Figure 8. Open ITRs

A more meaningful chart would be the set of still open ITRs. This is given in Table 2 and graphed in Figure 8. (several ITRs are not listed, being part of a still incomplete

OI.) Note that they are all relatively harmless severity 3 through 5 ITRs, except for the two severity 1s that are also listed in Figure 6 above. (Both of these are in the BFS system, were not considered critical for mission operations, and have been closed after we obtained the data we have analyzed.) As stated earlier, most of these ITRs will remain open until the appropriate source module is edited because of new requirements. These ITRs all refer to non-critical issues and the danger of faulty corrections is taken very seriously by NASA management. Appropriate managing of this large list of issues across multiple releases is a major value of the ITR process.

4 CONCLUSIONS

The value of this study resides in capturing and describing a successfully implemented model for IV&V. It is a process that carefully weighs the value of IV&V against the high costs of providing verification to all work products in the development. The ability to manage a large database of issues across multiple releases of the software without losing integrity of the product was a major goal of the process. Shuttle software is highly reliable, and the number of defects that manage to "slip through the cracks" is down substantially from the pre-IV&V 1980s.

Recently the management of the NASA/IVV center was moved from the Ames Research Center to the Goddard Space Flight Center, with a goal of expanding IV&V activities to additional NASA projects across the agency. This data provides a baseline that is useful for setting up additional IV&V-like activities at NASA and elsewhere. We already know that absolute perfection in software is an unrealizable goal. With data such as this from a well-organized large complex development, we can set a standard that other organizations can try to achieve.

In the Shuttle process, there are several competing players - NASA as the customer, several vendors building the software and other contractors evaluating the software. A mechanism such as described here can be useful for providing the right measurements for oversight of the process without each organization losing its own proprietary interests. This analysis is also applicable 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.

5 ACKNOWLEDGEMENTS

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

REFERENCES

1. Nancy Eickelmann, 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. William Florac, Anita Carlson and Julie Barnard, Statistical process control: Analyzing a space Shuttle onboard software process, *IEEE Software* (July, 2000) 97-106.

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

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

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

6. 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*.

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. M. Zelkowitz and I. Rus, Understanding IV&V in a Safety Critical and Complex Evolutionary Environment: The NASA Space Shuttle Program, ACM/IEEE International Conference on Software Engineering, Toronto, Canada, (May, 2001).