Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE @ DIRECT®

The Journal of Systems and Software 70 (2004) 143–154

The Journal of Systems and Software

www.elsevier.com/locate/jss

Defect evolution in a product line environment ☆

Marvin V. Zelkowitz [a,b,*], Ioana Rus [a]

[a] Fraunhofer Center for Experimental Software Engineering, Maryland 4321 Hartwick Road, Suite 500, College Park, MD 20740, USA
[b] Department of Computer Science, University of Maryland, College Park, MD 20742, USA

Received in revised form 23 October 2002; accepted 7 November 2002

Abstract

One mechanism used for monitoring the development of the Space Shuttle flight control software, in order to minimize any risks to the missions, is the independent verification and validation (IV&V) process. 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 over multiple releases of this software. Our findings reaffirm the value of IV&V, show the impact of IV&V on multiple releases of a large complex software system, and indicate that some of the traditional measures of defect detection and repair are not applicable in a multiple-release environment such as this one.
© 2003 Elsevier Inc. All rights reserved.

Keywords: Evolutionary software; Life and mission critical software; Software independent verification and validation; Metrics; Product line development; Process characterization; Space Shuttle program; Software safety and reliability

1. Introduction

From the 1970s until 1999, the National Aeronautics and Space Administration (NASA) released 22 versions of a large complex software system that pilots the Space Shuttle. Each version contains changes made to implement specific functionality for the next scheduled mission, and the collected set of releases can describe a product line development.

Removing software defects is a primary goal in software development, but for Shuttle mission safety is of utmost importance, so the removal of non-critical defects by modifying the source program must be weighed against the increased risk of mission failure due to that correction. This leads to questions about what is the appropriate measure of defect detection and removal in this environment. The use of a separate group to provide independent verification and validation (IV&V) on a software system is often cited as a means to ensure a high quality software product. In this paper we present an overview of this multi-level complex process for the NASA Space Shuttle software IV&V. [1] We then discuss the differences in defect detection and removal between a more traditional custom software environment and this multiple release product.

The NASA IV&V program for the Space Shuttle was instituted in 1988; in 1997 management for IV&V was transferred to the NASA/IV&V facility in Fairmont, WV (NASA, 2000). 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 V&V; they also include risk analysis, requirements analysis, issues tracking, and process evaluation.

☆ This work was performed as part of NASA Subcontract No. 93-393B-FUSA from the NASA IVV facility in Fairmont, WV to the Fraunhofer Center, Maryland and National Science Foundation grants CCR9706151 and CCR0086078 to the University of Maryland.

* Corresponding author. Tel.: +1-301-4038965; fax: +1-301-4053691.

E-mail addresses: marv@zelkowitz.com (M.V. Zelkowitz), irus@fc-md.umd.edu (I. Rus).

[1] See also Zelkowitz and Rus (2001a,b).

0164-1212/$ - see front matter © 2003 Elsevier Inc. All rights reserved.
doi:10.1016/S0164-1212(03)00013-X

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. In Section 5 we discuss differences between this environment and the more traditional software development environment.

2. Software independent verification and validation

According to 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." (NASA headquarters Safety and Mission Quality Office, 1992).

The IEEE Standard for Software Verification and Validation (1998) 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 in these two cases 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.

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

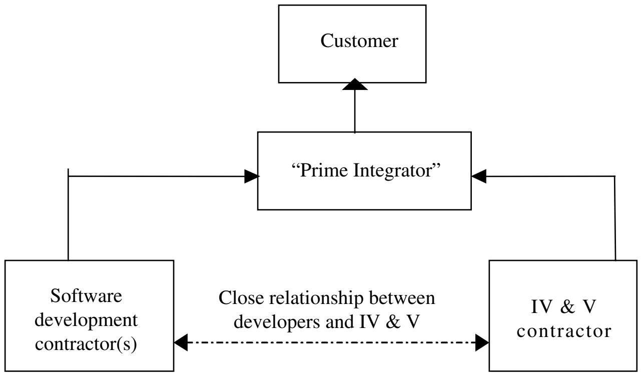
Customer
"Prime Integrator"
Software development contractor(s)
Close relationship between developers and IV & V
IV & V contractor

Fig. 1. Modified IV&V (from Leveson et al., 1993).

For the Shuttle software there is an additional level of IV&V, an *internal* IV&V, used by the development contractors, who have their V&V groups separate and, to some degree, 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, performed by a different contractor than the developer, has on Shuttle software development.

3. IV&V for Space Shuttle software

3.1. 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 OIs 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 6% of new or modified system code (core functionality) with each release, thus providing for a stable base software system (Eickelmann et al., 2000; Schneidewind, 1998; Schneidewind, 1999) (Table 1).

This is not a simple example of staged product evolution, where each new version of the product completely replaces the previous version. Rather it is more of a product line architecture where a base system of core functionality 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).

Fig. 2 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, which includes IV&V activities in understanding the impact of the changed requirements on the existing software base; a validation phase, which includes IV&V activities to evaluate the validation process performed

Table 1
Size of modifications per OI

OI	Total modified KSLOC/ OI	Modified functions KSLOC/ OI	Modified core KSLOC/ OI	% Modified of core KSLOC	% Modified of total KSLOC
I	55.7	29.4	26.3	6.5	12.4
J	44.3	21.3	23.0	5.1	9.8
K	44.0	34.4	9.6	2.1	9.8
L	47.3	24.0	23.3	5.2	10.5
M	50.7	10.4	40.3	9.0	11.3
N	50.4	15.3	35.	7.8	11.2
O	21.9	7.3	14.6	3.3	4.9
P	32.1	11.0	21.1	4.7	7.1
Q	57.2	12.1	45.1	10.0	12.7
Total	403.6	165.2	238.4		
Average	44.84	18.36	26.49	6.0	9.97
Standard Deviation	11.36	9.41	11.67		

KSLOC = Thousands of DSLOC.

by the software developer; a mission preparation phase which includes additional IV&V processes monitoring the evolution of the software; and an operational lifetime executing on one or more Shuttle missions. As Fig. 2 shows, during most of this period there was concurrent release development, about three OIs being in various phases of development (development, validation, or mission preparation), and up to four releases were active (either in execution or in development) during this period.

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. Different contractors developed PASS and BFS independently. 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. However, using redundancy as a risk mitigation strategy does not work as well for software failures as for hardware failures since a software failure in one computer is likely to manifest in the others as well (e.g., the Ariane 5 failure in 1996 (Lions et al., 1996)). For this reason critical defects must be eliminated from software, and IV&V is one of the means for doing so.

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. Therefore, an overall guiding principle in OI development is that changing any module, regardless of the reason, might leave the code open to new errors. 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 some changes often remain pending across multiple releases of the software. In fact, some proposed changes, as we later show, have remained unresolved for over 3000 days (over 9 years).

3.2. Shuttle software 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 process is briefly described in Fig. 3. Rectangles represent the various processes for building a new OI, whereas ovals represent the main data that tracks development issues. Another set of records, the issues tracking reports (ITRs), is used by the IV&V contractor to monitor any potential problems that are uncovered during development and will be discussed later. The shaded rectangles refer to the major IV&V activities. More complete descriptions of this process are given in

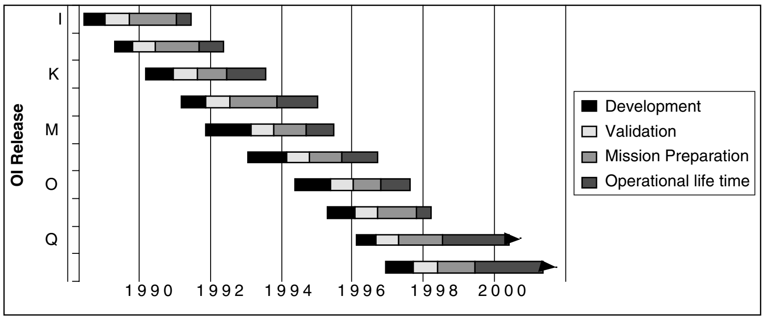

Fig. 2. Lifecycle for each OI.

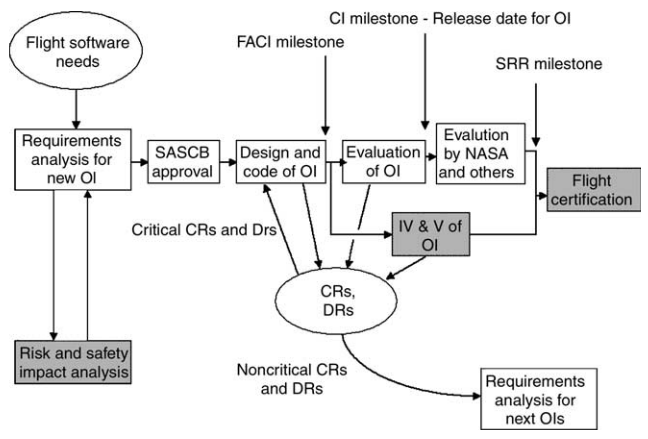

Fig. 3. Overview of Shuttle software development for an OI.

Florac et al. (2000) and Leveson et al. (1993). Briefly, the overall process is as follows:

- 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, performs 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., OIs) 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 OI is scheduled.
- The developer of the Shuttle software uses these requirements and modifies 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 future 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 V&V testing, and the IV&V contractor begins to analyze these tests.
- At the configuration inspection (CI) milestone the software is released by the developer to NASA, but it undergoes further evaluation before it 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 some CRs and DRs are intentionally delayed for implementation across multiple releases until a more advantageous time.

3.3. 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; and the schedule of IV&V activities to be performed.

The three shaded areas in Fig. 3 represent IV&V activities. 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 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, 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 in certain situations this might happen. 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.

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 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 and the multifunctional electronic display system (MEDS) also known as the "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 the 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 (McCaugherty, 1998).

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 from the set of 15. On these functions the IV&V effort and scope may vary. The scope of IV&V can be: *limited, focused,* or *comprehensive*. *Limited* comprises activities (a)–(e), *focused* comprises activities (a)–(i), and *comprehensive* comprises activities (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 testing (i.e., developer's functional and performance verification testing) 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 proprietary tool, CARA, (McCaugherty, 1998) to help decide on the level of IV&V scope to apply.

The IV&V contractor usually evaluates the CRs and DRs that are submitted to cover changes in the software. However they also often submit CRs and DRs themselves and in some cases 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 Schneidewind (1999).

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. During requirements analysis for an OI and thereafter (Fig. 3) the ITRs are used by IV&V to track any further IV&V issues. 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.

The ITRs represent potential anomalies with the Shuttle software systems, but do not necessarily represent errors or defects in the code. They simply represent those issues needing further clarification. Some will represent defects, which must be corrected, whereas many others are closed with no further action necessary.

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. For example, 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 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 an error that will not affect the safety of the current mission, a 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 a subsequent OI. Such changes are delayed until a later OI because of the danger of spurious modifications. Since all such changes need extensive testing for all changed modules, such non-critical changes are not made until needed at a later date.
- Note that a CR does not necessarily represent a change in a future OI. During the requirements process for that new OI, this CR will be evaluated along with all other proposed changes by the SASCB and either accepted, rejected, or delayed until a still-future OI.
- 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 problem identification performed during the *definition* phase of an OI. 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, and 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 programming style 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 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 due to some other required changes.

Some of the severity 1 and 2 ITRs represent issues that are outside of the operating environment of the software so they have a very small probability of occurrence, even though theoretically possible. Such issues are classified as 1N or 2N and are generally grouped with the severity 3 anomalies.

4.1. A characterization of the ITRs

Table 2 summarizes the set of ITRs that have been collected. For four of the severity 4 and 5 ITRs it was difficult to determine which system they affected. Although the PASS is the primary avionics system (538 ITRs or 69.2%), 314 ITRs (40.4% of the ITRs) concern the BFS. About 10% concern both systems.

As shown in Table 3, of the 773 ITRs for which we have a disposition,[2] 461 (59.6%) of the ITRs were closed and 312 (or 40.3%) of the ITRs were still open in mid-1999. However, all of the severity 1 and 2 PASS ITRs were closed and only two of the severity 1 BFS ITRs were still open at the time of our initial analysis. In 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.

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

Table 2
Summary of ITRs collected, by severity and location, 1988–1999

Severity	1	2	1N,2N,3	4	5	Total
PASS	7	6	85	219	142	459
PASS/BFS	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 3
Closed and still open ITRs in 1999

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

The ITRs indicate a possible resolution to the problems, but the ITRs have not been marked closed by the time of our analysis.[3] 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 1–3 ITRs. We classified these ITRs according to their disposition (resolution) 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 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 CR document for later SASCB approval.
- *Process*—The ITR reflects V&V 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.

In Table 4 we present the disposition or proposed disposition of severity 1, 2, and 3 ITRs. Of the 164, 64 result in no action and 20 require process ITRs for 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.

[3] Since we performed this analysis, these 2 ITRs have been closed.

Table 4
Disposition of ITRs, 1988–1999

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

Table 5
Early vs. late ITRs

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

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 5 we divide the ITRs into those found during the requirements and development phases (early ITRs) and those found after software release, that is after the CI milestone (late ITRs).[4] More than half (83 of 159) of the issues were discovered prior to release of the new OI.

Table 5 represents all ITRs, including those resolved with no change to the software. By looking at only the 80 ITRs from Table 4 that represent proposed changes (the CR and Chg columns), the data from Table 5 is reduced to that shown in Table 6. 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.

Table 7 gives the distribution of 51 severity 1–3 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. By comparison, 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. Typically IV&V ITRs numbered around 2–5% of the early defects that were not found by the developer, but from 15% to almost 25% of the late DRs for a particular release.

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

Table 6
ITRs representing proposed changes

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

Table 7
Early and late defects found, by OI

OI	ITRs by severity level				DRs	
	1–2 E	1–2 L	3 E	3 L	E	L
I				2	32	13
J				1	139	52
K		3		2	110	25
L			2	2	133	32
M				1	114	10
N			2		81	11
O			5	5	89	28
P	1		6		60	17
Q	1		5	3	67	12
R		1	2	7	61	9
Total	2	4	22	23	886	209

From 1988 until mid-1999 777 ITRs have been entered in the issues tracking database. Of those occurring in the 10 releases that we studied in detail, Fig. 4 shows:

- Issues are found fairly uniformly across OIs.
- Although the number of critical ITRs is quite low, they still exist; IV&V uncovered some of them that otherwise might have been not detected.
- A total of 20 severity 1 and 2 ITRs were found attributable to these 10 OIs. As explained previously, many of the severity 3–5 ITRs are held until a later OI to avoid changing source programs needlessly.

A more operational measure of IV&V effectiveness is determining if any critical errors have been present on Shuttle missions (Fig. 5). 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, nine flew on pre-IV&V software (i.e., before 1988), where three 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.

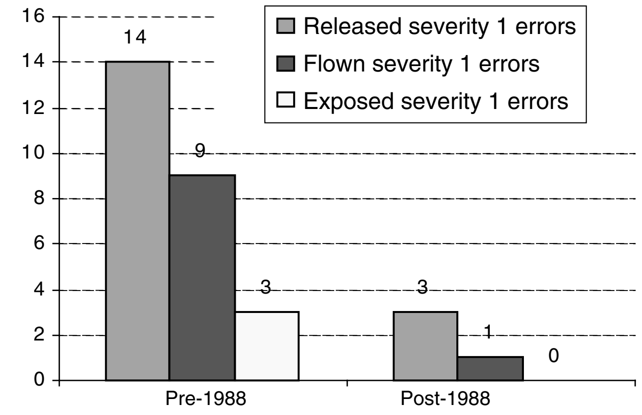

Fig. 5. Severity 1 defects.

IV&V effectiveness can also 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 (number of 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 baselining early errors. Plotting the EDR for a typical OI results in a graph much like Fig. 6. A flat line means that most errors were found early, a decaying line means that more errors were found later.

The final value of the EDR for the OIs analyzed here is given by Table 8. The overall EDR rate for all OIs is 0.81 (meaning 81% of all defects were found more than

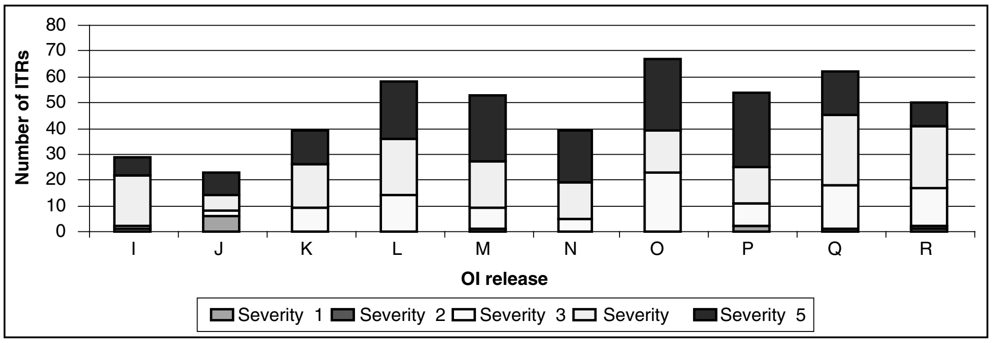

Fig. 4. ITRs across OI releases.

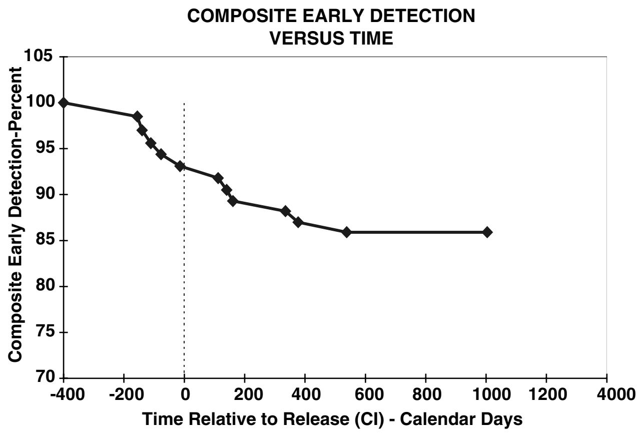

Fig. 6. Early detection rate.

Table 8
Composite EDR for multiple OIs

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

400 days before software release). In Table 8 the first four OIs have EDRs of about 0.7–0.8 and the last four have EDRs that ranged from about 0.75 to almost 0.9. While other software development improvements might have affected the error detection process, the IV&V process has definitely had an impact.

5. Product line defect detection

The previous section provides a set of quantified measures of the benefits of the IV&V process for defect detection and elimination. However, we also want to understand the propagation of defects across multiple releases of the software. In Fig. 7, we reproduce the lifecycle given in Fig. 1 with the lifetime of the severity 1 and 2 ITRs. Under each OI there are the severity 1 (circles) and 2 (diamonds) ITRs attributed to that OI. Those that precede release I were attributed to OIs that precede the introduction of IV&V in 1988 and are drawn above the bar corresponding to OI I. The bar connecting two dots represents the elapsed time from creation to closure of that ITR. A single dot represents an ITR that was opened and closed within a relatively short time period.

Fig. 7 indicates that 22 ITRs (18 severity 1 and four severity 2) are attributed to the graphed 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, after the introduction of IV&V for OI I, only 12 severity 1 and 2 anomalies were found, and none remained open for more than one additional OI. (The one severity 2 OI was identified during the requirements process before development of OI Q began.)

A more meaningful chart would be the set of currently open ITRs. This was summarized earlier in Table 3. For the 312 still open ITRs in 1999, Fig. 8 plots the date each one was first uncovered. (Several ITRs are not listed, being part of an OI still incomplete at the time of analysis.) Note that they are all relatively harmless severity 3–5 ITRs, except for the two severity 1s that were discussed earlier with Table 3 (and Footnote 3).

By plotting the open ITR data of Fig. 8 according to the OI where it was created, we get an overview of how ITRs evolve over time. In Fig. 9, for each OI, we give the number of ITRs detected in that OI, the number of ITRs

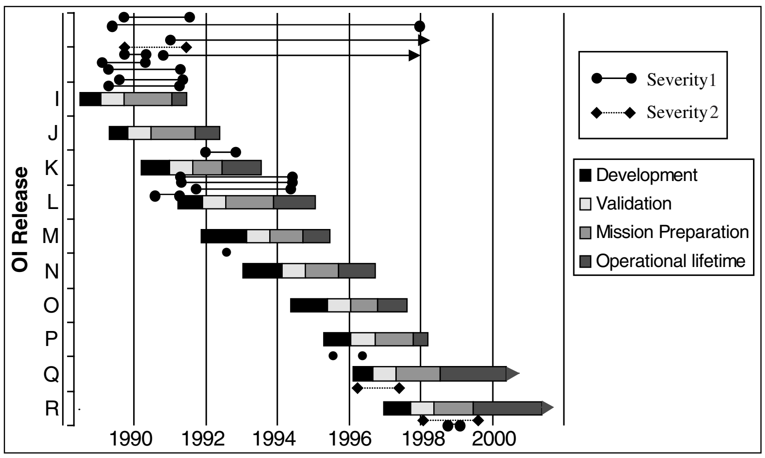

Fig. 7. Severity 1 and 2 ITRs.

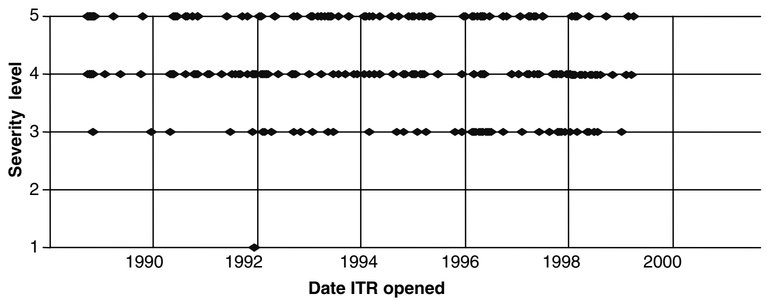

Fig. 8. Open ITRs.

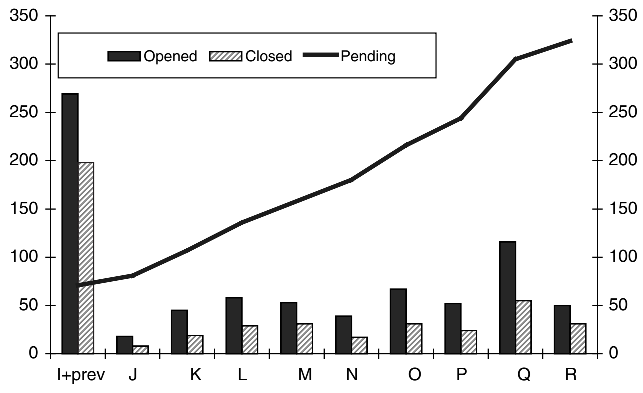

Fig. 9. Growth in open ITRs.

closed eventually that were initially opened in that OI and the number of ITRs still pending from all previous OIs.

It clearly shows that the number of ITRs remaining open due to each OI is an increasing function. However, recall from Table 3, that except for two, severity 1 and 2 ITRs have all been closed and over 250 of the remaining open ITRs are severity 4 and 5.

A measure of defect detection that is often used in software development needs to be reexamined in the light of the Space Shuttle experience. Usually the number of days a problem report remains open is a measure of the effectiveness of the defect removal process. But this may not be useful from a product line perspective. Recall that Fig. 9 shows that the set of open ITRs increases over time. From the closed serious ITRs (severity 1–3, Table 3), we computed the number of days that each remained open. This is displayed as Fig. 10. 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 and slightly more than 25% took more than 420 days.

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

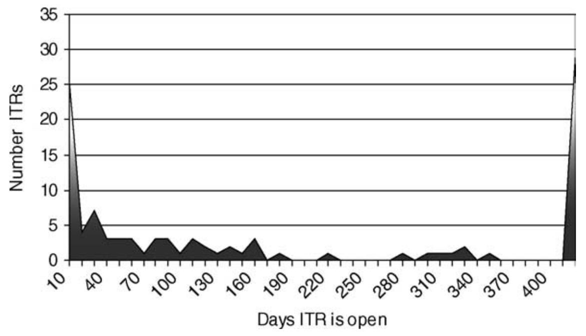

Fig. 10. Days an ITR remained open.

where necessary. The IV&V ITR process keeps track of multiple issues over several OIs.

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.

6. Conclusions

In this paper we present an overview of the NASA Space Shuttle software IV&V process and an analysis of the ITRs produced during the IV&V process. 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 [5] 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

[5] The annual budget for IV&V the last 3 years has been approximately $3–3.5 million, which is approximately 3% of the budget allocated annually for the complete software development and assurance process.

cracks" has declined substantially from the pre-IV&V 1980s.

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 10–20 such issues. This study is only a first approximation of IV&V for this program.

Therefore, relative to IV&V on the Space Shuttle program we found the following:

- There is a demonstrated value of IV&V to the NASA Space Shuttle program. From Table 6 we can see that 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.
- Quantitative IV&V studies are unfortunately quite rare. Our results are similar to a small controlled study performed at NASA Langley Research Center (Arthur et al., 1998), and although we could not attach a cost savings for this activity, previous studies have shown a return on investment of between 1.3 and 1.8 on similar NASA software (Rogers et al., 2000).
- Many of these defects were found during the requirements analysis phase. Almost one half (24 out of 51 from Table 7) 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 4 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 one 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 nine severity 1 errors on Shuttle flights prior to the introduction of the IV&V process.

Important note: 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 some documentation was unclear and a restatement of the specification resolved the issue. The independence of the two groups shows those different approaches to evaluating software leads to an increased defect discovery process.

Most IV&V processes have been organized around developing a correct and risk free 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 complex systems is important for achieving reliability in such critical systems in the future.

Because of safety considerations, defects are not necessarily fixed when found in the Shuttle's product line development model. The development process must adapt and be able to track such issues across multiple versions of the software.

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.

It is clear from the data presented here, that finding all defects in NASA's highly engineered development process 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 Shuttle software since 1988.

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.

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

Arthur, J.D., Groener, M.K., Hayhurst, K.J., Holloway, C.M., 1998. Adding value to the software development process: a study in independent verification and validation. Technical Report 98–15, Virginia Tech., Blacksburg, VA.

Eickelmann, N., Rus, I., Zelkowitz, M., 2000. Preliminary case study findings of the Space Shuttle software evolution as a product line process, ISAW-4 workshop at ICSE 2000. Limerick, Ireland.

Florac, W., Carlson, A., Barnard, J., 2000. Statistical process control: analyzing a Space Shuttle onboard software process. IEEE Software (July), 97–106.

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

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

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

McCaugherty, D., 19998. The criticality and risk assessment (CARA) method. NASA Workshop on Risk Management, Farmington, PA, October 1998.

NASA, 2000. 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.

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.

Rogers, R., McCaugherty, D., Martin, F., 2000. A case study of IV&V return on investment. In: Third Ann. Systems Eng. and Supportability Conference, October 2000.

Schneidewind, N.F., 1998. How to evaluate legacy system maintenance. IEEE Software (July), 34–42.

Schneidewind, N.F., 1999. Measuring and evaluating maintenance process using reliability, risk, and test metrics. IEEE Transactions on Software Engineering 25 (6), 769–781.

Zelkowitz, M.V., Rus, I., 2001(a). Understanding IV&V in a safety critical and complex evolutionary environment: the NASA Space Shuttle program. In: IEEE Computer Society and ACM International Conf. on Soft. Eng., Toronto, CA, May 2001. pp. 349–357.

Zelkowitz, M.V., Rus, I., 2001(b). The role of independent verification and validation in maintaining a safety critical evolutionary software in a complex environment: the NASA Space Shuttle program. In: International Conference on Software Maintenance, Florence, Italy, November 2001. pp. 118–126.

Ioana Rus is a scientist at Fraunhofer Center for Empirical Software Engineering, Maryland. Her research interests focus on software process modeling and simulation, process improvement, measurement, and empirical studies in software development and evolution. Her work also addresses models and methods for software dependability engineering, as well as experience and knowledge management in software engineering. She has a Ph.D. in Computer Science and Engineering and is a member of the IEEE Computer Society and ACM. She has worked as software engineer, assistant professor, and researcher.

Marvin V. Zelkowitz is a professor of Computer Science at the University of Maryland holding a joint appointment with the University's Institute for Advanced Computer Studies. He is also Chief Scientist of the Fraunhofer Center for Experimental Software Engineering, Maryland, an applied research and technology transfer organization located near the campus of the University of Maryland in College Park. Prof. Zelkowitz received the MS and Ph.D. degrees from Cornell University in Computer Science in 1969 and 1971, respectively and the BS in mathematics from Rensselaer Polytechnic Institute in 1967. He is a fellow of the IEEE, a Golden Core member of the IEEE Computer Society, and a member of ACM. He is the series editor of Academic Press' "Advances in Computers" book series, and on the editorial boards of Empirical Software Engineering and Computer Languages.