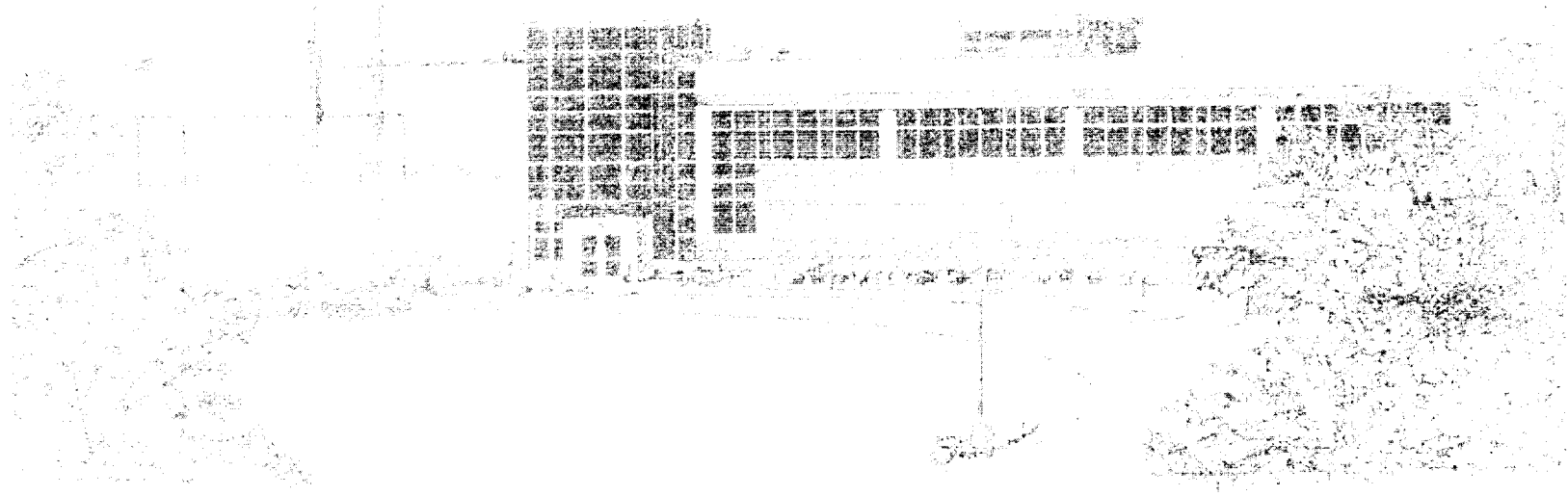
NASA-IVV-97-007
WVU-IVV-97-007
WVU-CS-TR-97-010

NASA/WVU Software IV & V Facility
Software Research Laboratory
Technical Report Series

A Case Study of IV&V Cost Effectiveness

By Ralph D. Neal, Dan McCaugherty, Tulasi Joshi, and John Callahan



National Aeronautics and Space Administration

West Virginia University

NASA-IVV-97-007

NASA IV&V Facility, Fairmont, West Virginia

A Case Study of IV&V Cost Effectiveness

Ralph D. Neal, Dan McCaugherty, Tulasi Joshi, and John Callahan

July 14, 1997

This technical report is a product of the National Aeronautics and Space Administration (NASA) Software Program, an agency wide program to promote continual improvement of software engineering within NASA. The goals and strategies of this program are documented in the NASA software strategic plan, July 13, 1995.

Additional information is available from the NASA Software IV&V Facility on the World Wide Web site http://www.ivv.nasa.gov/

This research was funded under cooperative Agreement #NCC 2-979 at the NASA/WVU Software Research Laboratory.

A CASE STUDY OF IV&V COST EFFECTIVENESS

Ralph D. Neal, West Virginia University
Dan McCaugherty, Intermetrics, Inc.
Tulasi Joshi, Fairmont State College
John Callahan, West Virginia University

I. Abstract

This paper looks at the independent verification and validation (IV&V) of NASA's Space Shuttle Day of Launch I-Load Update (DoLILU) project. IV&V is defined. The system's development life cycle is explained. Data collection and analysis are described. DoLILU Issue Tracking Reports (DITRs) authored by IV&V personnel are analyzed to determine the effectiveness of IV&V in finding errors before the code, testing, and integration phase of the software development life cycle. The study's findings are reported along with the limitations of the study and planned future research.1

Keywords: IV&V, software, verification, validation, cost, schedule, quality

II. Introduction

Management in government, private industries, and academia have been concerned about software (s\w) development that is feasible, within cost estimates, and on schedule with a high quality product that fulfills the requirements. These concerns are well founded. A study of 8380 application development efforts performed by the Standish Group [1994] found that cost overruns average about 189% of the original cost estimate and 31% of development efforts were ultimately canceled. Of the developments that were completed, only 16% delivered the initially specified functions while the remaining 53% delivered on the average about 61% of the initially specified functions. The leading causes for these development failures and cost overruns occurred early in the life cycle and included incomplete, unclear, and unstable requirements and specifications and lack of user input. Independent verification and validation (IV&V) is designed to mitigate these types of development risks. The Jet Propulsion Laboratory (JPL) reports that quality assurance (QA) and IV&V together assure the quality, reliability and maintainability of s\w products. [JPL, 1985].

IV&V is an important part of the assurance to which JPL alludes. The JPL report states that studies have shown that expenditure on software QA activities for a typical s\w project can add 5% to 50% to the cost of development [JPL, 1985], while other studies show that IV&V can add 10% to 15% to the cost of development. However, since maintenance costs are 40% to 80% of the life cycle costs, depending on which study you believe, the cost of the IV&V effort can be offset or exceeded by the savings and benefits that result from its use.

Of the techniques used to assure quality, IV&V often incurs more expense in the early life cycle phases. However, this initial expense pays dividends in the form of more problems being found earlier in the life cycle when they are cheaper to fix. Cost savings figures vary but Boehm [1981] and others have shown that the costs to correct an error in later life cycle phases may increase by a factor of four to ten in each subsequent phase. If IV&V can be shown to uncover errors earlier in the development life cycle than they would have otherwise been uncovered, the cost to correct them may be only one-fourth to one-tenth the cost to correct them without IV&V.

1 Funded in part by NASA Cooperative Agreement NCCW-0040

III. Independent Verification and Validation

Independent verification and validation (IV&V) consists of the analysis and testing of a software developer's requirements, design, and code by an independent agent. The agent inspects the software development process and products to ensure compliance with the requirements of the system and to promote product quality. The independent agent normally reports to the contracting agency directly rather than through the organization doing the development [NASA, 1995; Makowsky, 1994].

IV&V is conducted parallel to, but separate from, the system software development activities. The objective is to ensure that the software development project will satisfy requirements, design, implementation, operations, and maintenance objectives. The primary goal is to minimize the inherent risk of software systems development and maximize confidence in the result. The effectiveness of IV&V is lessened when IV&V is out of phase with the development process. This result can be minimized when the IV&V effort is an integral part of software development [Makowsky, 1994].

IV. The Software Life cycle

The classification of phases of the software development life cycle (SDLC) varies from one software development organization to another. For the purpose of this study, the (SDLC) has been divided into six phases that are generally consistent with: 1) the framework of Intermetrics, Inc. for its IV&V functional plan; 2) the choice of the 1985 JPL study on cost effectiveness of IV&V [JPL, 1985]; and 3) the waterfall model of Boehm [Boehm, 1981]. The six phases are defined as follows:

Phase 1: Requirements (Reqt)
Phase 2: Preliminary Design (PD)
Phase 3: Detailed Design (DD)
Phase 4: Implementation (coding and unit testing) (Imp)
Phase 5: Integration and System Test (IST)
Phase 6: Operation and Maintenance (Ops)

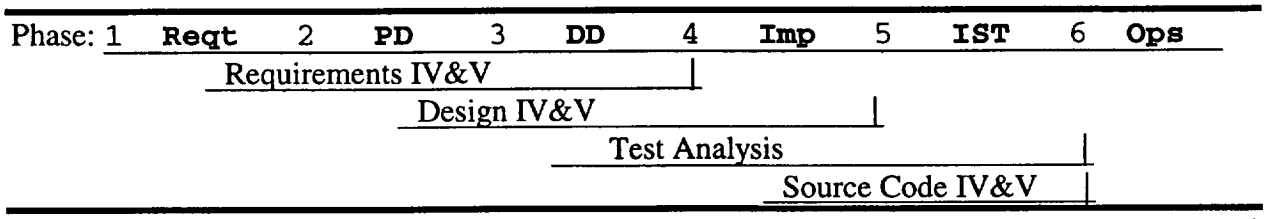
In this study, each computer software configuration item (CSCI) was developed on its own life cycle, i.e., within a larger system life cycle. The life cycles employed modeled the waterfall method close enough to be fitted within the above SDLC framework. It can be noted here that IV&V activities normally stop at the end of IST (phase 5).

V. Types of IV&V Activities

IV&V consists of the assessment activities: reviewing, analyzing, testing, and monitoring. Formal reviews and audits indicate milestone achievement sufficient to warrant progressing into the next phase of the development process. They serve as the basis for verification and validation [Makowsky, 1994].

While the IV&V activities vary along many dimensions from project to project and from contractor to contractor, the following text and graphic (Figure 1) represent a generic IV&V operation and IV&V as it was carried out for the DoLILU project. The open ended bars represent that the starting time for IV&V activities is not set; the closed end of the bars represent that the ending time for IV&V activities is set and coincides with the end of a SDLC phase. Requirement documents are examined to determine the degree of compliance by the software systems developer to the systems requirements. IV&V analysis of requirements normally starts while the development team is conducting the software requirements analysis (Phase 1) and finishes along with the software detailed design (Phase 3) and the critical design reviews that immediately follow the detailed design [Makowsky, 1994]. Design documents are evaluated to determine the degree of compliance with the system requirements. IV&V analysis of design normally starts during the preliminary design (Phase 2) and concludes with implementation (Phase 4) [Makowsky, 1994]. Source code is examined to determine the degree of compliance with the requirements and design. Source code examination normally starts during coding and unit testing (Phase 4) and concludes at the end of integration and system testing (Phase 5) [Makowsky, 1994]. As mentioned earlier, IV&V

Figure 1. The SDLC phases depicted on a timeline:



is normally not carried out during Ops (Phase 6).

VI. Justification of IV&V

NASA has used IV&V for software projects that require high reliability, such as in the case of the Shuttle Program and the Space Station Program. These programs are expensive and have a high degree of risk. In programs such as these where the potential social or financial loss resulting from an operational failure is so high, the cost of IV&V is considered a worthwhile investment because of the increased confidence, that results. Finding an error after the code has been put into operation is unacceptable when life may hang in the balance.

Aside from the justification for IV&V from an operational perspective, there are a number of studies and opinions that support the performance of IV&V in the successful completion of software development, e.g., SED-SES-IES-001, 1994; Makowsky, 1992; ESA PSS-05-10, February 1994; SMAP-GB-A201, September 1989; and NASA-GB-001-95, April 1995. Some of the benefits that have been pointed out in the literature are that IV&V:

supports early auditing of configuration management, especially for evolutionary development;
creates better documentation at all levels, i.e., requirements, design, code, and test;
reduces risk;
increases confidence in the software;
causes latent errors to become evident;
creates a better structure for management review;
reduces schedule slippage;
in general helps decision-makers make better decisions;
motivates the prime contractor;
helps the developer decide when testing has been completed;
and appears to be cost-effective.

The economic benefit that comes from IV&V is realized when errors are found earlier than they might otherwise have been found. Boehm estimated that the cost to fix an error for large projects increases 5 times if detected at phase 2, 10 times at phase 3, 20 times at phase 4, 40 times at phase 5 and 100 times at phase 6 relative to the cost to fix the error at phase 1 [Boehm, 1981; p.40]. Wolverton [1980] found the same geometric trend in the increase of costs to fix an error from one phase of the software life cycle to the next. Assuming unit cost at the requirements phase, Wolverton found that it cost 2.5 times as much to correct an error when detected at the design phase, 5 times as much at the code and unit test phase, and 36 times as much at the test and integration phase.

VII. OBJECTIVE

Intermetrics, Inc. was the NASA contractor for the performance of IV&V during the requirements definition, design, implementation, test, and acceptance life cycle phases of the Space Shuttle day of launch I-load update (DoLILU) phase I. DoLILU Issue Tracking Reports (DITRs) act as the nuts and bolts for the success of IV&V. DITRs facilitate the formal documentation and tracking of issues and errors. These issues may be specific errors, potential errors, general concerns, or observations that require resolution or clarification. The DITRs describe the issue, the recommended corrective action and the impact of not resolving the issue. A formal response to the recommendation is required by the organization responsible for correcting the problem. The DITRs provide a field for this response. NASA, IV&V, the developer, and quality assurance organizations review the response provided by the developer at assemblies of the DoLILU Working Group. The transmittal of DITRs to the responsible organization enables the timely resolution of issues and errors. DITRs identifying issues pertinent to maintaining the desired level of system safety are tracked and identified as such by IV&V and remain open until it is determined that an appropriate corrective action has been implemented [Intermetrics, 1993].

Figure 2 (reproduced from [Ramamoorthy, et al., 1984]) shows the percentage of errors introduced contrasted with the percentage of errors found by life cycle phase for the typical software development life cycle. The four columns, in each set, represent the phases of a four-phase life cycle model. These four phases correspond to the six phases of the model used in the DoLILU project thus: DoLILU phases two and three are combined into column two; DoLILU phase four is represented

by column three, DoLILU phase five is represented by column four, and DoLILU phase six is not represented. The DoLILU project did not track errors into phase six. Thus, the six phases of DoLILU are telescoped into four phases with the loss only of detail. The set of columns on the left represent the percentage of errors introduced into the software during each phase of the life cycle. The set of columns on the right represent the percentage of errors observed during each phase of the life cycle. So, thought the detail is not the same in the study by Ramamoorthy, et al., the trends of error introduction and error observation are clear and can be fit to the DoLILU life cycle.

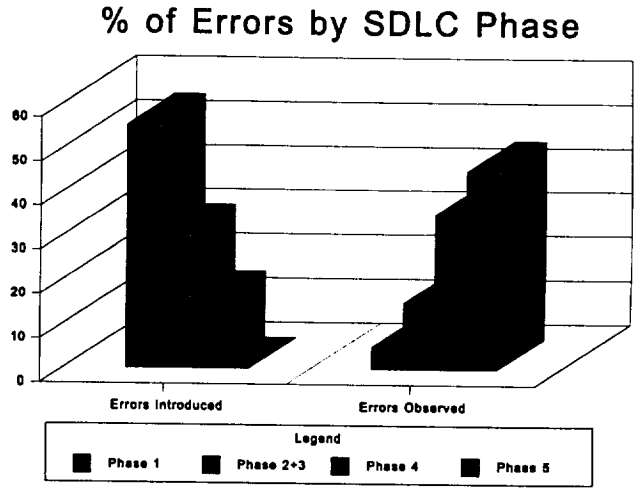


Figure 2. [Ramamoorthy, et al., 1984]

In general, errors are introduced in the earlier stages of the development life cycle and are found in the later stages of the development life cycle. A number of studies, as discussed earlier, have supported the notion that the application of IV&V is cost effective. Is this true in the case of DoLILU? These studies suggest that IV&V will not only pay for itself, but will save money, especially if its application detects errors at earlier stages of the software life cycle. The earlier in the SDLC that errors are observed, the cheaper they may be corrected [Boehm, 1981]. The objective of this study is to verify the cost effectiveness of IV&V as applied to DoLILU through a case study of Intermetrics' DoLILU Issue Tracking Reports (DITRs).

VIII. HYPOTHESIS

The IV&V for DoLILU reduced overall development life cycle cost by exposing errors as early as possible and by reducing the probability of latent errors.

IX. DATA COLLECTION

There were 109 DITRs in total. Nine of them were excluded from this study. Seven of them were not relevant to the software life cycle, and two of them were withdrawn because the issues involved were invalid. The Intermetrics DoLILU IV&V manager, now located at the NASA Software Systems and Technology Facility (NASA's IV&V facility) in Fairmont, West Virginia, prepared a large number of the DITRs, and identified the phases of the life cycle to which each DITR was applicable. To make this study compatible with studies mentioned earlier the software life cycle was divided into the six phases as previously outlined. Each error from each DITR was recorded separately. From the 100 DITRs used for this study, 695 errors were identified. Each error was identified with a DoLILU CSCI, and the phase in the life cycle where the error was observed. For example, a requirements related error identified during the code phase is assigned to phase 4 and not to phase 1.

X. ANALYSIS

There were nine DoLILU CSCIs associated with the 695 errors. These CSCIs were given the names DIBS, DIVDT, DOLSEND, DTS, ILUV, LTQS, RSOLNK, SVDS and SYSTEM. LTQS was associated with the highest number of errors accounting for almost 50% of the errors. DIVDT accounted for over 15%. Three CSCIs (ILUV, DTS, and DIBS) accounted for 7.6 TO 8.5% each, and the remaining four CSCIs accounted for less than 4% each (see Table 1). Out of the 695 errors detected, about 15% of them were found in phase 1, 6% in phase 2, 20% in phase 3, 3% in phase 5, and over half (57%) were found in phase 4.

When the nine CSCIs are plotted based on the percentage of errors found in or before phase 3, a bimodal distribution results (see Figure 3). Notice that the first band (0 to 20%) contains three CSCIs, the second band (21 to 40%) contains one CSCI, the fourth band (61 to 80%) contains one CSCI and the fifth band (81 to 100%) contains four CSCIs. There is a large unexplained gap in the middle. No CSCI shows an even distribution of observed errors across the beginning and ending phases of the SDLC. When a histogram is made of this data the bimodal distribution is even more evident (see figure 4).

Table 1	Phase 1	2	3	4	5	*	TOTAL	PERCENT
DIBS	13	1	44	1	0		59	8.49
DIVDT	**12S**	**1S**	**3S**	78	11		105	15.11
DOLSEND	**N**	**N**	**N**	20	1		21	3.02
DTS	1	36	0	16	0		53	7.63
ILUV	**N**	**N**	**N**	56	0		56	8.06
LTQS	**45S**	**1N**	**71S**	220	6		343	49.35
RSOLNK	4	0	8	0	0		12	1.73
SVDS	15	0	1	3	0		19	2.73
SYSTEM	17	0	9	1	0		27	3.88
TOTAL	107	39	136	395	18		695	
PERCENT	15.40	5.61	19.57	56.83	2.59			

* phase 6 error data not available
Full IV&V, i.e., all life cycle products were available for review (default).
Some IV&V, i.e., some life cycle products were available for review.
No IV&V, i.e., no life cycle products were available for review.

These results beg the question as to why IV&V was less effective for the four CSCIs in the 0-20% and 21-40% bands. Further examination of the specific development techniques indicates that, prior to the implementation phase, the four CSCIs in the lower percentage bands did not have the full compliment of standard SDLC products on which to perform IV&V, i.e. they did not follow the SDLC model to which this study is applied. We have, therefore, divided the CSCI developments into two groups, those that followed the SDLC, Group 1, and those that did not, Group 2. Table 2 provides the distribution of errors across the SDLC for these groups both in terms of the number of errors found in each phase and the percentage of errors found in each phase relativeto the total of errors from all phases. These data indicate that early IV&V leads to

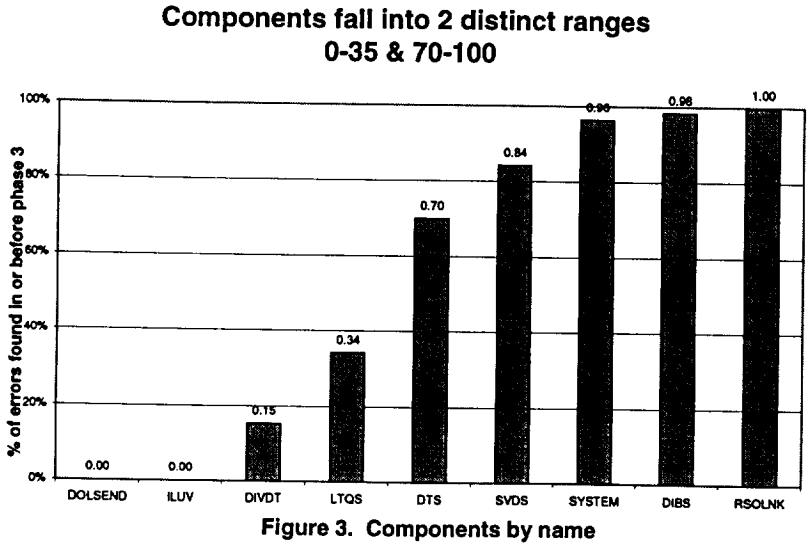


Figure 3. Components by name

early detection of errors and that the lack of early IV&V leads to more errors being found by IV&V at later, more expensive, phases of the development life cycle. The errors reported here do not include the errors found and eliminated by the developer, i.e., the errors reported here are only those found by IV&V. If IV&V was not employed, and therefore not finding errors, the number of errors found by the developer in phase 4 or later would be compounded by an amount proportional to the number of errors found by IV&V in the first three phases. If IV&V takes place in parallel with development, these errors are found earlier when they are cheaper to fix. If IV&V is not active until later in the life cycle, the errors that could have been discovered earlier are still there to be discovered and corrected later at a higher cost.

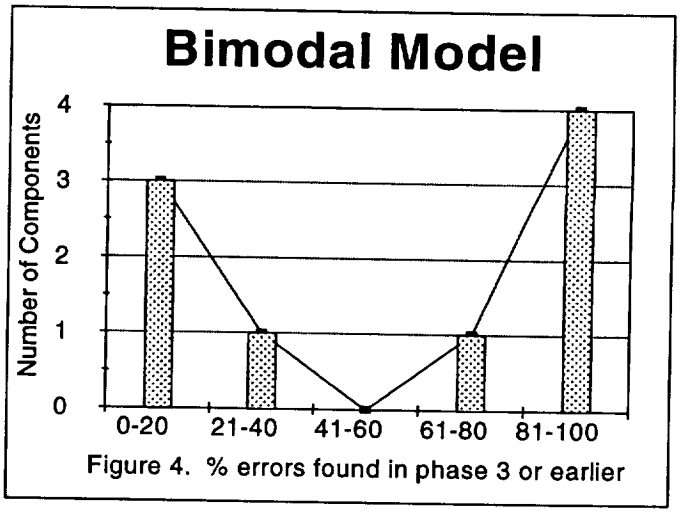


Figure 4. % errors found in phase 3 or earlier

Another way to view this data is to show the percentage of errors found in each phase for groups 1 and 2 relative to the percentages found by development efforts in general (according to figure 2). These percentages are plotted in figure 5. In this figure there is a substantial increase in the percentage of errors found by an IV&V agent in the requirements and design phases relative to normal development trends. This is a testament to how IV&V, when applied to developments following sound software engineering practices, results in the identification of errors early in the development rather than later. Additionally, both group 1 and 2 demonstrate a tremendous reduction in errors found during the integration and system test phase, Phase 5. Very few errors are found during phase 5 even if IV&V doesn't start in earnest until phase 4.

Table 2

		PHASE1	PHASE2	PHASE3	PHASE4	PHASE5	TOTAL
Group 1	count	50	37	62	21	0	170
% of group		29.41	21.76	36.47	12.35	0.00	
Group 2	count	57	2	74	374	18	525
% of group		10.86	0.38	14.10	71.24	3.43	

XI. CONCLUSION

Of special interest is the small number of errors found during phase 5 relative to typical error trends in general. This indicates that, given a chance to review the work, IV&V was effective in greatly reducing latent errors that are very expensive to correct.

From the data gathered on DoLILU, it seems that IV&V is most cost effective if it is performed from the very beginning of the software life cycle. Although it is less cost effective if it is performed only at later phases (especially after phase 3) IV&V is effective in reducing costly latent errors when applied during the code phase. Therefore, the hypothesis that "The IV&V for DoLILU reduced life-cycle cost by exposing errors as early as possible and by reducing the probability of latent errors." was accepted.

This study, however, has many limitations. The relative severity of the errors has not been discussed. Therefore, although it is reasonable to assume that the errors found by IV&V range from the insignificant to the critical and have the same distribution as all of the errors in the system, it would be helpful to be able to rank the errors by severity. It was not possible to determine actual costs either in a dollar figure or man hour/months because data were not available at the time of the study. Therefore, the savings from finding errors earlier in the life cycle are based on previous studies instead of being based on

empirical data from this study. We have no data on subsequent requirement changes or later review processes. The complexity and size of the CSCIs were not considered.

This was a first attempt to learn about the cost effectiveness of IV&V in the NASA setting. More studies are needed wherein the concerns mentioned above are controlled for. Studies have to be designed and implemented at the start of projects instead of relying on the data collected incidentally during a project. Typically IV&V is applied because of the potential catastrophic consequences of a software failure. Future studies are needed to further examine and quantify the cost effectiveness of IV&V to determine appropriate applications of IV&V for software engineering in general.

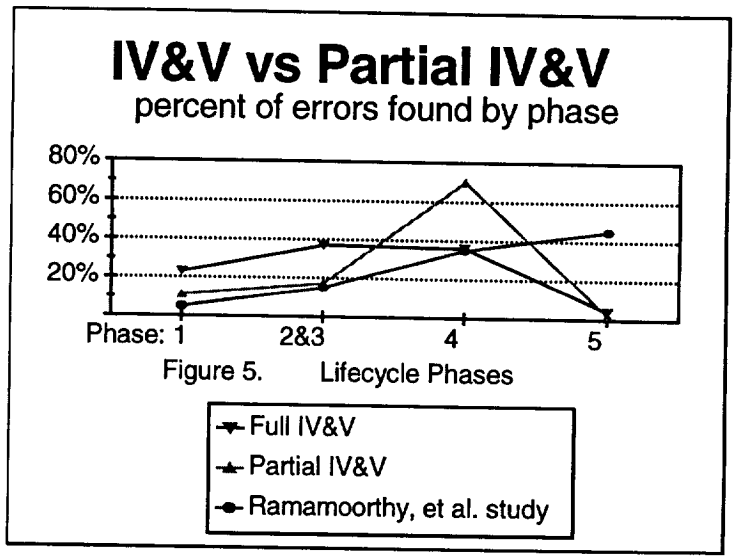


Figure 5. Lifecycle Phases

Bibliography

Boehm, Barry W., Software Engineering Economics, Prentice Hall, Inc., Englewood Cliffs, NJ, 1981.

Brooks, Fred, The Mythical Man-Month: Essays on Software Engineering, Addison-Wesley Publishing, Reading, MA,1975.

Farr, William H., A Survey of Software Reliability Modeling and Estimation, NSWC TR 82-171, September 1983.

Intermetrics, Inc., Independent Verification and Validation Plan for the Day of Launch Iloads Update System Phase II, 1993.

JPL, Cost-Effectiveness of Software Independent Verification and Validation, NASA RTOP #323-51-72, 1985.

Makowsky, Lawrence C., A Guide to Independent Verification and Validation of Computer Software, U.S. Army Belvoir RD&E Center, 1994.

NASA-RPT-004-95, Profile of Software at the National Aeronautics and Space Command, March 1995.

Putnum, Lawrence H., and Ware Myres, Measures for Excellence, Yourdon Press, Englewood Cliffs, NJ, 1992.

Ramamoorthy, et al., *Software Engineering*, IEEE Computer, October, 1984.

RTOP#323-51-72-NASA, Cost-Effectiveness of Software Independent Verification and Validation, October 1985.

SED-SES-IES-001, Software Engineering Directorate: Software Engineering Evaluation System (SEES), August 1994.

Shooman, M., *Software Reliability: Measurement and Models*, Proceedings of the 1975 Annual Reliability and Maintainability Symposium, 1975.

Standish Group, The, *Charting the Seas of Information Technology*, A Special COMPASS Report, The Standish Group International, Inc., Dennis, MA, 1994.

Wolverton, R.W., Airborne Systems Software Acquisition Engineering Guidebook: Software Cost Analysis and Estimating, U.S. Air Force ASD/EN, Wright-Patterson AFB, OH, 1980.

A CASE STUDY OF IV&V COST EFFECTIVENESS

Ralph D. Neal, West Virginia University
Dan McCaugherty, Intermetrics, Inc.
Tulasi Joshi, Fairmont State College
John Callahan, West Virginia University

I. Abstract

This paper looks at the independent verification and validation (IV&V) of NASA's Space Shuttle Day of Launch I-Load Update (DoLILU) project. IV&V is defined. The system's development life cycle is explained. Data collection and analysis are described. DoLILU Issue Tracking Reports (DITRs) authored by IV&V personnel are analyzed to determine the effectiveness of IV&V in finding errors before the code, testing, and integration phase of the software development life cycle. The study's's findings are reported along with the limitations of the study and planned future research.

Keywords: IV&V, software, verification, validation, cost, schedule, quality