# A PRACTICAL EXPERIENCE WITH INDEPENDENT VERIFICATION AND VALIDATION

Gerald Page*, Frank E. McGarry**, and David N. Card*

* Computer Sciences Corporation, Silver Spring, Maryland 20910
** National Aeronautics and Space Administration, Greenbelt, Maryland 20771

## ABSTRACT

One approach to reducing software cost and increasing reliability is the use of an independent verification and validation (IV&V) methodology. The Software Engineering Laboratory (SEL) applied the IV&V methodology to two medium-sized flight dynamics software development projects. Then, to measure the effectiveness of the IV&V approach, the SEL compared these two projects with two similar past projects, using measures like productivity, reliability, and maintainability. Results indicated that the use of the IV&V methodology did not help the overall process nor improve the product in these cases.

## INTRODUCTION

Independent verification and validation (IV&V) is the systematic evaluation of software by an independent organization, i.e., not the development organization. During the software development process, the IV&V team provides an objective appraisal of the development process and product to

- Detect problems earlier and consequently solve them earlier
- Ensure that all defined requirements are addressed at each development stage
- Provide managers with better visibility of the development process

These improvements should lead to more reliable software with better cost and schedule control.[1] However, the cost effectiveness of the IV&V methodology, when compared to more traditional development approaches, has not been fully demonstrated.

This paper describes an attempt to assess the benefits and limitations of the application of IV&V in one particular development environment.

## SOFTWARE ENGINEERING LABORATORY

The environment of the IV&V experiment was the National Aeronautics and Space Administration/ Goddard Space Flight Center's (NASA/GSFC's) flight dynamics area. There, software is developed for such spacecraft problems as attitude determination and control, mission planning, and maneuver control. This software development environment has been studied for 7 years by the Software Engineering Laboratory[2] (SEL), a research project sponsored by NASA/GSFC and supported by the Computer Sciences Department at the University of Maryland and by Computer Sciences Corporation.

The SEL's goals are to understand and improve the overall software development process. To this end, the SEL conducts experiments with production software projects and measures the effect of the techniques applied, identifying and adopting beneficial methodologies for future projects.

During the past 7 years, the SEL has studied more than 45 software development projects totaling more than 2 million lines of source code. Most flight dynamics projects are developed on a group of IBM mainframe computers using FORTRAN and assembler programming languages. The applications projects monitored by the SEL are largely scientific and mathematical in nature with moderate reliability requirements but with severe development time constraints that are imposed by a fixed spacecraft launch date. The development process typically takes between 18 and 24 months from the beginning of preliminary design to the end of software acceptance testing. Depending on mission characteristics, the size of a system ranges from 30,000 to 120,000 lines of source code, with an average of 30 percent reused from previous similar projects.

An IV&V methodology was applied to two typical flight dynamics development projects in an attempt to determine the benefits of the approach. Each project was in development for 2 years and was approximately 65,000 lines of source code in size.

## THE IV&V METHODOLOGY

The major functions of an IV&V team are (1) to ensure that the product of each phase of the software life cycle is consistent with the product of the previous phase (i.e., verification) and (2) to ensure that the product of each phase accurately responds to the original software requirements

0730-3157/84/0000/0453$01.00 © 1984 IEEE

(i.e., validation). The benefits of using this methodology are claimed to be

Earlier detection of errors
Increased software reliability
Improved software maintainability

The additional product review activities of the IV&V team, especially in the early life cycle phases, should produce more complete and earlier error detection, as well as a more easily maintained product. Increased visibility and reduced cost are potential secondary benefits. The IV&V team provides an independent and impartial assessment of project status, thus increasing its visibility to management. Life cycle cost is reduced if fewer errors are propagated through to maintenance and operations, especially in environments where maintenance is a large part of the total life cycle cost. In addition, the system development cost should be reduced by earlier error detection and correction.

A study[3] of large TRW, GTE, and IBM software development projects suggests that the difficulty of correcting an error increases the longer the error remains in the system. Figure 1 (from that study) shows that the cost of correcting an error doubles for each life cycle phase that it remains in the system. Data collected by the Software Engineering Laboratory (SEL) support the existence of a trend of increasing cost. By finding errors soon after they enter the system, the IV&V team should reduce the cost of subsequent error correction during development.

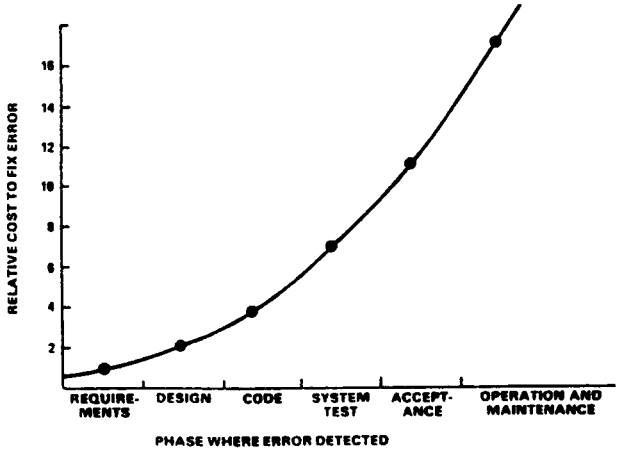


Figure 1. Cost of Correcting Software Errors (TRW, GTE, IBM Data[3])

## THE IV&V EXPERIMENT

The goal of this study[4] was to determine which of the potential benefits (previously described) would be obtained by applying the IV&V methodology to flight dynamics projects. To perform this evaluation, the SEL developed an IV&V plan and applied it to two typical attitude projects. The results of these projects were compared to the results of two previous attitude projects (that did not use IV&V) with respect to seven performance measures.

## PROJECTS STUDIED

Each of the experimental projects was initially estimated to require 5 to 7 staff-years of effort to produce approximately 55,000 lines of source code, exclusive of the IV&V effort. The IV&V team was expected to expend an additional 15 percent of the development team's effort.

The two past projects with which the IV&V projects were compared actually required 9.1 and 9.2 staff-years of effort to produce 85,000 and 90,000 lines of source code, respectively. These projects were completed and operational in 1979. Both IV&V projects were completed and operational in 1981. All four projects were developed on the same IBM mainframe computers in FORTRAN. Table 1 describes some other characteristics of the projects.

Table 1. Characteristics of Projects Studied

| CHARACTERISTICS | PROJECTS | | | |
|---|---|---|---|---|
| | IV&V 1* | IV&V 2* | PAST 1 | PAST 2 |
| SIZE (THOUSANDS OF DELIVERED LINES OF SOURCE CODE) | 66 | 67 | 85 | 90 |
| EFFORT (STAFF-MONTHS) | 9.8 | 9.8 | 9.1 | 9.2 |
| DURATION (WEEKS) | 88 | 94 | 81 | 70 |
| AVERAGE STAFF (FULL-TIME EQUIVALENT) | 5.2 | 4.9 | 5.3 | 6.1 |
| APPLICATION EXPERIENCE (AVERAGE YEARS) | 6.2 | 7.2 | 5.9 | 7.0 |

*EXCLUDES IV&V TEAM.

Data were collected both manually and automatically from all four projects during and after development. Errors detected and hours charged were reported on forms filled out by the development and IV&V teams. These data were quality assured and stored in a computer data base for easy retrieval.

## FUNCTIONS OF THE IV&V TEAM

The SEL developed an IV&V plan based on the relevant software engineering literature.[5] An IV&V team was selected and trained to

Validate requirements to ensure completeness and correctness

Verify design to ensure that it is a complete and correct translation of the requirements

Perform independent system testing to ensure that the software satisfies the requirements

Verify consistency from requirements to software to operation

Fix nothing

Report all discrepancies and other findings

The application and overall experience of the IV&V technical staff was similar to that of the average

development team; the managers, however, had more experience. The IV&V team shared resources with developers and operations personnel. The IV&V team was expected to use about 15 percent of the development resources to perform these functions.

## PERFORMANCE MEASURES

One approach[6] to assessing the benefit of using an IV&V team is to count the number of faults found by the team, to determine the probable cost of those errors had they become latent errors through later stages of the life cycle, and, from this, to compute a dollar benefit of using the team. This procedure ignores the possibility that some (perhaps all) of the faults found by IV&V might have been found by the development team in the absence of the IV&V team.

The approach adopted by the SEL was to define relevant performance measures and then to compare the overall performance of a development team working with an IV&V team to the performance of a development team working alone. The following seven measures were defined, covering every phase of the software life cycle:

Operational Reliability--Errors discovered during operation per thousand lines of source code developed

Maintenance Cost--Effort (staff-hours) per error corrected during maintenance

Requirements Specification Quality--Errors per thousand lines of source code (found during code and test) that were attributed to the requirements specification

System Design Quality--Errors per thousand lines of source code (found during code and test) that were attributed to the system design

Software Implementation Quality--Errors per thousand lines of source code detected during acceptance testing

Testing Effort--Percent of development effort required for system and acceptance testing

Development Cost--Effort (staff-months) per thousand lines of source code developed

These measures indicate the degree to which earlier error detection was achieved, reliability was increased, and maintainability was improved (the purported benefits of IV&V). Values of these measures for two IV&V-assisted projects were compared with values from two similar (but non-IV&V) past projects. This enabled the overall effectiveness of the IV&V methodology to be evaluated as well as its effect on each life cycle phase.

## EXPERIMENTAL RESULTS

The IV&V process for the experimental projects lasted 14 to 16 months and required an additional effort of 16 to 18 percent of the development effort (close to the initial 15 percent estimate). IV&V staffing averaged 1.1 persons and peaked at 3.0 persons (full-time equivalents). Six different individuals were involved in the IV&V team, including technical managers.

Both development teams expended 9.8 staff-years of effort. One project produced 66,000 and the other 67,000 lines of source code. Both size and cost substantially exceeded the initial estimates. Table 2 shows the values obtained for the seven performance measures from the IV&V and past projects.

Table 2. Comparison of Performance Results

| PERFORMANCE MEASURE | IV&V PROJECTS | PAST PROJECTS |
|---|---|---|
| OPERATIONAL RELIABILITY (ERRORS PER KSLOC) | 0.9 | 0.6 |
| MAINTENANCE COST (STAFF-HOURS PER ERROR) | 14 | 13 |
| REQUIREMENTS SPECIFICATION (ERRORS PER KSLOC) | 0.6 | 0.5 |
| SYSTEM DESIGN (ERRORS PER KSLOC) | 2.1 | 2.2 |
| ACCEPTANCE TESTING (ERRORS PER KSLOC) | 1.9 | 1.9 |
| TESTING EFFORT (%) | 34 | 36 |
| DEVELOPMENT COST (STAFF-MONTHS PER KSLOC) | 2.2 | 1.4 |

NOTE: KSLOC = THOUSANDS OF DEVELOPED LINES OF SOURCE CODE.

The following sections discuss the performance results obtained from the experiment in more detail.

## OPERATIONAL RELIABILITY

The additional testing and verification effort provided by the IV&V team should increase the operational reliability of the delivered software. However, the error rates achieved by the two IV&V projects were greater than the error rates for the two past projects. The average error rate for the two past projects was 0.6 error per thousand lines of developed code; both IV&V projects had higher error rates (an average of 0.9). The use of an IV&V methodology did not improve the quality of the software put into operation.

## MAINTENANCE COST

Early detection of errors by the IV&V team should reduce the cost of correcting errors during maintenance. That is, relatively few requirements, design, or serious coding errors should remain in the system when it enters the maintenance and operations phase. Those errors found should be trivial and few. However, the average effort required to correct an error (during maintenance) for the IV&V projects (14 hours) was about the same as that for the non-IV&V projects (13 hours). Given that the operational reliability of the IV&V projects was not any better, the relative maintenance cost of the IV&V projects was greater than that of the non-IV&V projects.

## REQUIREMENTS SPECIFICATION QUALITY

One effect of an IV&V team should be to reduce the number of requirements errors that are propagated through to the coding phase, yet the requirements error rates for the IV&V and non-IV&V projects were about equal (0.6 and 0.5 error per thousand lines of source code, respectively). However, very few requirements remained unspecified in the later stages of development. Hence, there were very few late surprises in terms of requirements problems as compared with the past projects. The use of an IV&V methodology did decrease the impact of requirements errors, ambiguities, and misinterpretations. Nevertheless, because requirements errors are not a major problem in this environment (few requirements errors occur), the effect on the overall development process was minor.

## SYSTEM DESIGN QUALITY

Another effect of an IV&V team should be to reduce the number of design errors that are propagated through to the coding phase. In this experiment, the design error rate for the IV&V projects (2.1 errors per thousand source lines of code) was approximately equal to that of the past projects (2.2 errors per thousand source lines of code). The use of an IV&V methodology did not produce any reduction in design errors reaching coding.

## SOFTWARE IMPLEMENTATION QUALITY

The additional system testing and review performed by the IV&V team should result in fewer development errors being uncovered during acceptance testing. Both IV&V and non-IV&V projects demonstrated an acceptance testing error rate of 1.9 errors per thousand lines of source code. The use of an IV&V methodology produced no change in the reliability of the software entering acceptance testing.

## TESTING EFFORT

Early error discovery should facilitate system and acceptance testing, thereby reducing the effort required. Since testing can account for 40 percent of the development cost in this environment, there is the potential for substantial cost savings in this area. In this experiment, the IV&V projects demonstrated only a marginal reduction in the effort required for system and acceptance testing relative to the past projects (from 35.9 to 34.3 percent).

## DEVELOPMENT COST

It was expected that the cost of using the IV&V methodology would be confined to the additional cost of operating the IV&V team. However, the two IV&V projects (2.6 staff-months per thousand lines of source code) were about 85 percent more expensive than the two past projects (1.4 staff-months per thousand lines of source code). Since the operational reliability of the software was not any better, an 85-percent increase in cost for the same product is a very expensive penalty to pay. The cost of the development part of the IV&V projects alone (2.2 staff-months per thousand lines of source code) was approximately 55 percent higher than the past development cost. The cost of using IV&V is high.

## CONCLUSIONS AND RECOMMENDATIONS

In summary, the performance measures indicate that the first application of an IV&V methodology in the flight dynamics environment

Did not produce a more reliable product
Did not detect errors earlier
Did not improve maintainability

The overall conclusion is that IV&V is not cost effective for NASA/GSFC's flight dynamics projects, especially since other software techniques have been shown to improve software quality in this environment at little or no net cost.[7]

Despite these results, it may be possible to better integrate the IV&V methodology into the software development process to make IV&V more cost effective in the flight dynamics environment for

The right size effort
The right reliability requirement

Most ground-based flight dynamics projects require 8 $\pm$ 4 staff-years of effort. An IV&V methodology may be cost effective for larger projects. For onboard (flight) systems with a more stringent reliability requirement, an IV&V methodology may be cost effective for 5- to 6-staff-year efforts. In both these cases, an IV&V effort of approximately 15 percent of the development effort should be sufficient in the flight dynamics environment.

Software developers should keep in mind, however, that no software engineering methodology can replace technical and managerial expertise. It may be best to regard IV&V as an insurance policy[8]: an additional premium that should be paid when the consequences of failure are great.

## REFERENCES


1. S. B. Mohanty, "On Software Verification & Validation," Proceedings of the Software Verification and Validation Symposium, June 1981

2. Software Engineering Laboratory, SEL-81-104, The Software Engineering Laboratory, D. N. Card, F. E. McGarry, G. Page, et al., February 1982

3. B. W. Boehm, "Software Engineering R&D Trends and Defense Needs," Research Directions in Software Technology. MIT Press: Massachusetts, 1979

4. Software Engineering Laboratory, SEL-81-110, Evaluation of an Independent Verification and Validation (IV&V) Methodology for Flight Dynamics, G. Page and F. E. McGarry, to be published

5. Michael S. Deutsch, "Verification and Validation," Software Engineering. Prentice-Hall, Inc.: New Jersey, 1979

6. M. S. Fujii, "Five Major IV&V Projects: A Quantitative Analysis," Proceedings of the Software Verification and Validation Symposium, June 1981

7. D. N. Card, F. E. McGarry, and G. Page, "Evaluating Software Engineering Technologies in the SEL," Proceedings of the Eighth Annual Software Engineering Workshop, November 1983

8. J. B. Munson, "Acquisition Management of Embedded Computer Systems and the Role of IV&V," Proceedings of the Software Verification and Validation Symposium, June 1981