In proceedings, 2nd World Conference on Integrated Design and Process Technology (IDPT) Austin, Texas, December 1-4 1996, as a keynote address to the track entitled "The Process Road from Requirements to System Architectures (and back)"

THE ROLE OF INDEPENDENT V&V IN UPSTREAM SOFTWARE DEVELOPMENT PROCESSES

Steve Easterbrook
NASA/WVU Software Research Lab
NASA IV&V Facility, 100 University Drive, Fairmont, WV 26554
steve@atlantis.ivv.nasa.gov

Abstract

This paper describes the role of Verification and Validation (V&V) during the requirements and high level design processes, and in particular the role of Independent V&V (IV&V). The job of IV&V during these phases is to ensure that the requirements are complete, consistent and valid, and to ensure that the high level design meets the requirements. This contrasts with the role of Quality Assurance (QA), which ensures that appropriate standards and process models are defined and applied. This paper describes the current state of practice for IV&V, concentrating on the process model used in NASA projects. We describe a case study, showing the processes by which problem reporting and tracking takes place, and how IV&V feeds into decision making by the development team. We then describe the problems faced in implementing IV&V. We conclude that despite a well defined process model, and tools to support it, IV&V is still beset by communication and coordination problems.

INTRODUCTION

Software errors can be very expensive. Earlier this year, a software error was responsible for the loss of the European Space Agency's Ariane 5, which was carrying a payload worth $500 million. The error was due, primarily, to a failure to ensure that new requirements (since Ariane 4) were fully propagated to all parts of the design [1]. A number of opportunities to detect the error were missed during the development process.

Schulmeyer [2] uses the term *defects* to describe latent errors that remain in the software after delivery. In the quest for zero-defect software, strategies are needed for cost effective error prevention, and error detection and removal. Most importantly, the earlier that errors are detected, the cheaper they are to correct. Boehm [3] showed that fixing a requirements error in the operations phase can be 100 times more expensive than fixing it in the requirements phase. The reasons are simple. Errors made in the requirements and high level design phases may affect many different parts of the program. Therefore reliance on testing alone is not cost effective for these types of error. In addition, software that is exhaustively tested against its requirements can still have defects, if the requirements were wrong in the first place.

The job of ensuring high quality software is generally termed *software assurance*. In practice, there are a number of logically distinct activities that together cover the assurance role [4]. It is normal to distinguish between Quality Assurance (QA), Verification and Validation (V&V) and Integration and Test (I&T). QA provides an overall check that the standards and procedures are defined and adhered to. Verification and Validation ensures that the software will satisfy its functional, performance and quality[1] requirements. I&T is responsible for thorough testing of the code, from unit testing, through integration, to acceptance testing by the customer. V&V should not be confused with I&T. The role of V&V is to perform analyses *throughout* the development process, to detect problems as early as possible, preferably before they show up in testing.

The theme of this track of the conference is "The Process Road from Requirements to Systems Architectures [and back]". The road metaphor is a useful one to help understand the differences between QA and IV&V. The process road needs policing, to prevent problems such as speeding (i.e. moving too fast through the requirements and high level design phases), and driving unsafe vehicles (i.e. faulty or missing requirements, poor choice of architecture, lack of traceability, etc.). The police are also needed when accidents occur, helping to clear up the mess, and sometimes making recommendations for preventative improvements. This job is divided up as follows. QA ensures that appropriate laws ("standards and procedures") are applied, and checks for infringements of these laws. V&V's job is to make sure that vehicles traveling on the road are roadworthy, are heading in the right direction, and will make it safely to their destinations.

This paper concentrates on the role of V&V in early development phases. We first define verification and validation, and explain the importance of independence. We then describe how IV&V is applied in NASA projects, and describe an example IV&V analysis activity. The remainder of the paper describes some of the problems in ensuring that IV&V is effective.

ROLE OF IV&V

The terms *Verification* and *Validation* are commonly used in software engineering to mean two different types of analysis. The usual definitions are:

Validation: Are we building the correct system?

Verification: Are we building the system correctly?

[1]The NASA software assurance standard, NASA-STD-2201-93, distinguishes activities concerned with quality requirements (reliability, maintainability, and so on) from those concerned with functional and performance requirements, calling the former software quality engineering, and the latter V&V. For simplicity, we treat all such activities as V&V in this paper.

In other words, validation is concerned with checking that the system will meet the customer's actual needs, while verification is concerned with whether the system is well-engineered. The distinction between the two terms is largely to do with the role of specifications. Validation is the process of checking whether the specification captures the customer's needs, while verification is the process of checking that the software meets the specification.

Verification includes all the activities associated with the producing high quality software: testing, inspection, design analysis, specification analysis, and so on. It is a relatively objective process, in that if the various products and documents are expressed precisely enough, no subjective judgments should be needed in order to verify software. In contrast, validation can be an extremely subjective process. It involves judgments of how well the (proposed) system addresses a real-world need. Validation includes activities such as requirements modeling, prototyping and user evaluation.

In a traditional phased software lifecycle, verification is often taken to mean checking that the products of each phase satisfy the requirements set in the previous phase. Validation is relegated to just the beginning and ending of the project: requirements analysis and acceptance testing. This view is common in many software engineering textbooks, and is misguided. It assumes that the customer's needs can be captured completely at the start of a project, and that those needs will not change while the software is being developed. In practice, requirements change throughout a project, partly in reaction to the project itself: the development of new software makes new things possible. Therefore both validation and verification are needed throughout the lifecycle.

For practical purposes, the distinction is not important. V&V is now regarded as a coherent discipline: "Software V&V is a systems engineering discipline which evaluates the software in a systems context, relative to all system elements of hardware, users, and other software" [5].

Independence

For *independent* V&V, the customer hires a separate contractor to analyze the products and process of the software development contractor. This analysis is performed in parallel with the development process, throughout the software lifecycle, and is additional to any in-house V&V performed by the developer. IV&V is applied in high-cost and safety-critical projects to overcome analysis bias and reduce development risk. The customer relies on the IV&V contractor as an informed, unbiased advocate to assess the status of a project's schedule, cost, and the viability of its product during development:

> *"[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. The IV&V agent should have no stake in the success or failure of the software."* [4].

IV&V employs many types of analysis, including analysis of requirements, design, code, performance, schedule, and cost, as well as testing.

In practice, different degrees of independence are possible. There are three dimensions along which independence of the IV&V agent can be measured: managerial, financial and technical [6]. Managerial independence refers to the separation of responsibility for IV&V from the organization developing the software. This separation should allow the IV&V agent to decide for itself how and where to focus its efforts. Financial independence refers to control of the IV&V budget. If this budget is separated from the development budget, it reduces the risk that IV&V funding will be diverted or reduced to exert influence on the IV&V agent. Technical independence refers to the fresh perspective gained from using personnel who are not involved in the development effort, and whose understanding of the software and its requirements is not influenced by the development team. Technical independence can also be increased by using different tools and techniques than those used by the development team.

As well as the degree of independence, the depth and coverage of analysis performed by an IV&V agent may vary depending on resources available, and the criticality of the software being developed.

Relationships

Figure 1 shows the relationships between customer, developer and IV&V agent in the ideal model for IV&V. In weaker forms of IV&V, the IV&V agent might report to the program management office within the customer ("modified IV&V"), or directly to the development contractor ("internal IV&V") [7].

The tensions between the IV&V and development contractors can affect their working relationship. The two contractors have potentially conflicting goals: the development contractor's goal is to produce the required system within cost and schedule constraints; the IV&V contractor's goal is to identify errors and risks. These goals are in conflict whenever problems identified by IV&V have cost or schedule implications. To prevent this conflict causing problems, it is important that the relationship is based on mutual respect. For example, if

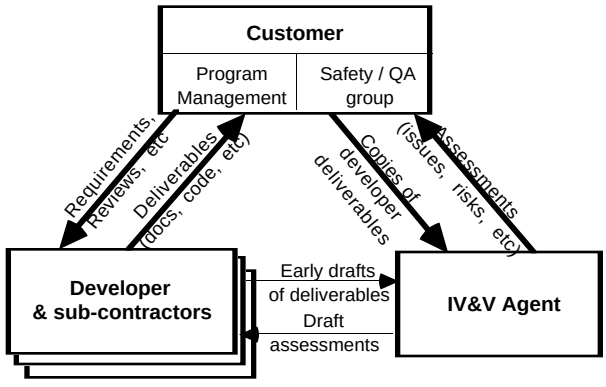

Figure 1: The relationships between IV&V agent, developer and customer.

the developer regards the IV&V agent as an ally in the effort to produce high-quality software, then the conflict can be avoided. If they regard the IV&V agent as an enemy put there to find fault with their work, the conflict becomes central to their relationship.

THE PRESCRIBED PROCESS

During the 1970's and 1980's, software came to play an important role in enabling NASA's development of larger and more complex spacecraft. Unfortunately, NASA's culture of prototype and test did not carry over well to the software domain [8]. A number of factors make software significantly different, and inherently less safe. Blum [9] lists: the complexity of software in relation to its size (software has no duplicated components), the lack of a fabrication stage (software is purely a design), and chaotic behavior (huge changes in behavior in response to minor changes in input). In addition to these, software is seen as more malleable than hardware: because of the lack of a fabrication stage, most engineers believe that it is easier to alter software than hardware in response to changing requirements.

Although software was not implicated in the Challenger accident in 1983, the subsequent inquiry offered a chance to assess all aspects of NASA's development processes. The Rogers commission identified a lack of independent oversight of development processes as a significant factor in the Challenger accident. There was no process for dealing with problems that arose in the engineering processes, and in particular, a lack of independent risk assessment. Risks were accepted in the face of schedule pressures, while the role of separate safety panels was reduced. Two subsequent NRC reports warned that software is under-represented in NASA's safety programs, and that "many of the same mistakes that contributed to the Challenger accident are now being repeated with respect to software" [6]. These reports recommended that NASA adopt software IV&V for shuttle and for all future manned missions. The adoption of IV&V was an attempt to redress the imbalance between safety and cost/schedule pressures, and to strengthen the oversight function.

Value of IV&V

The main value of IV&V is the fresh perspective it offers on questions of software safety and correctness. Like a doctor providing a second opinion on a life-threatening diagnosis, IV&V provides a second opinion to counter-balance that of the developer. Questions about safety and risk then become a dialogue rather than a monologue.

The cost of IV&V is typically a few percent of the entire development cost, while the benefit is a significantly reduced risk of loss of life, loss of a spacecraft, or loss of a mission. For example, IV&V for space shuttle software costs approximately $3.2 million per year. The cost of each mission is around $700 million, and the cost to replace a shuttle is estimated at $2 billion [6].

As well as reducing risk, IV&V has other benefits [10]. Errors are found earlier in the development process, and therefore are cheaper to fix. While this does not imply testing should be any less rigorous, savings can be made because the requirements specifications used to drive the testing process are clearer, and less effort may be needed for error-removal and re-testing. The delivered software should have fewer defects.

There is strong pressure from within NASA to measure the effectiveness of IV&V, in order to ensure value for money from its contractors. Although the general benefits of IV&V are known, there is no generally accepted means of measuring the effectiveness of a specific IV&V contract. A small number of quantitative studies have been conducted over the last decade [10]. These have focused on the collection of metrics on the number and severity of problems identified, and have estimated the potential cost of delayed detection of these problems. Such studies appear to have established that, in general, IV&V pays for itself several times over because of early detection of problems. However, these studies were fraught with methodological difficulties, and the benefits clearly depend to a very great extent on how IV&V is implemented.

The prescribed IV&V Process

The IV&V process adopted on recent NASA programs is consistent with that described by Lewis [7]. Ideally, IV&V begins before the development contract is awarded, reviewing the contract itself, and the decisions made during the bidding process. It then performs analysis activities throughout the devlopment process, producing regular briefings for the customer. The normal mode of working is to receive draft deliverables from the developer, perform various kinds of analysis on them, and report the results simultaneously to the developer and the customer.

The core of Lewis's process model for IV&V is a description of the activities to be performed at each phase of the development process. Figure 2 illustrates the portion of this model pertaining to the requirements phase. Without going into too much detail, it can be seen that the model encompasses activities such as checking for completeness, consistency, and testability of the requirements, performing requirements traceability analyses, evaluating the proposed development environment, assessing applicability of chosen standards and guidelines, monitoring schedule, staffing and other resource issues, and tracking issues and risks.

Similar process models are defined for each phase of the traditional software lifecycle. This set of process models is intended to be a complete account of all possible IV&V activities. Each individual project will then tailor the process by selecting those activities that are appropriate for the level of criticality and available resources.

CASE STUDY

We now describe an example IV&V activity. The IV&V team try to use methods and tools that are different from (but complimentary to) those applied by the development team, in order to maximize their technical independence. The example we describe here involves the use of a formal method to analyze a section of an informal requirements specification. While formal methods are not yet in

widespread use in IV&V, the activity we describe should give a good idea of the focus and scope of IV&V requirements analyses.

Our example concerns the analysis of specifications on the Space Station project. A Software Requirements Specification (SRS) is written by the relevant development contractor for each Computer Software Configuration Item (CSCI). These are written in natural language, and follow the format of DOD-STD-2167A. The IV&V contractor periodically receives copies of the SRS documents, in various stages of completion. These are analyzed for technical integrity by the IV&V contractor, in order to identify any requirements problems or risks. The kind of analysis performed will vary according to the level and the type of specification. If problems are identified, the IV&V contractor may recommend that either the requirements be rewritten, or the problem be tracked through subsequent phases.

We focus here on the analysis of the Fault Detection, Isolation and Recovery (FDIR) requirements for the Bus Controller, on the main 1553 communications bus on the space station. The bus controller monitors for communication errors on the bus, and if it detects errors in two consecutive processing frames, it may invoke an appropriate recovery response, depending on a number of conditions. The requirements for this function were specified in the SRS in English prose. An example :

(2.16.3.b) While acting as the bus controller, the C&C MDM CSCI shall reset the current SPD channel and pause the Bus FDIR logic on the SPD 1553 channel for TBD (6-12 seconds) seconds (until the channel is back on-line) upon detection of transaction errors of the selected messages to RTs whose 1553 FDIR is not inhbited in two consecutive processing frames within 100 millisec of detection of the second transaction error if; the SPD Channel's reset capability is not inhibited, the bus has not been switched in the major (10-second) frame, and either:
•1) the switch to the alternate bus is inhibited, or
•2) the bus has been switched in the last major frame.

The IV&V team might examine a number of properties of these requirements, including:

- Clarity - are the requirements clear and unambiguous?
- Consistency - are they free from contradiction?
- Traceability - do these requirements trace properly to higher level requirements, to the design, to test cases?
- Validity - do these requirements correctly describe the needs for automated failure detection and recovery? Do they accurately reflect the results of previous failure mode analyses performed on the bus architecture?
- Completeness - are all the failure modes and recovery actions covered? Does each requirement give a complete description of the conditions needed to decide whether the prescribed recovery action is the correct one?
- Testability - is there a way to verify whether these requirements are met in the implementation?

In this example, the requirements as stated were not clear, as the combinations of ‘ands’ and ‘ors’ in an English sentence were very hard to interpret correctly. The IV&V analysts therefore produced a logic table, to represent the combination of conditions described in these requirements, and recommended to the development team that a similar

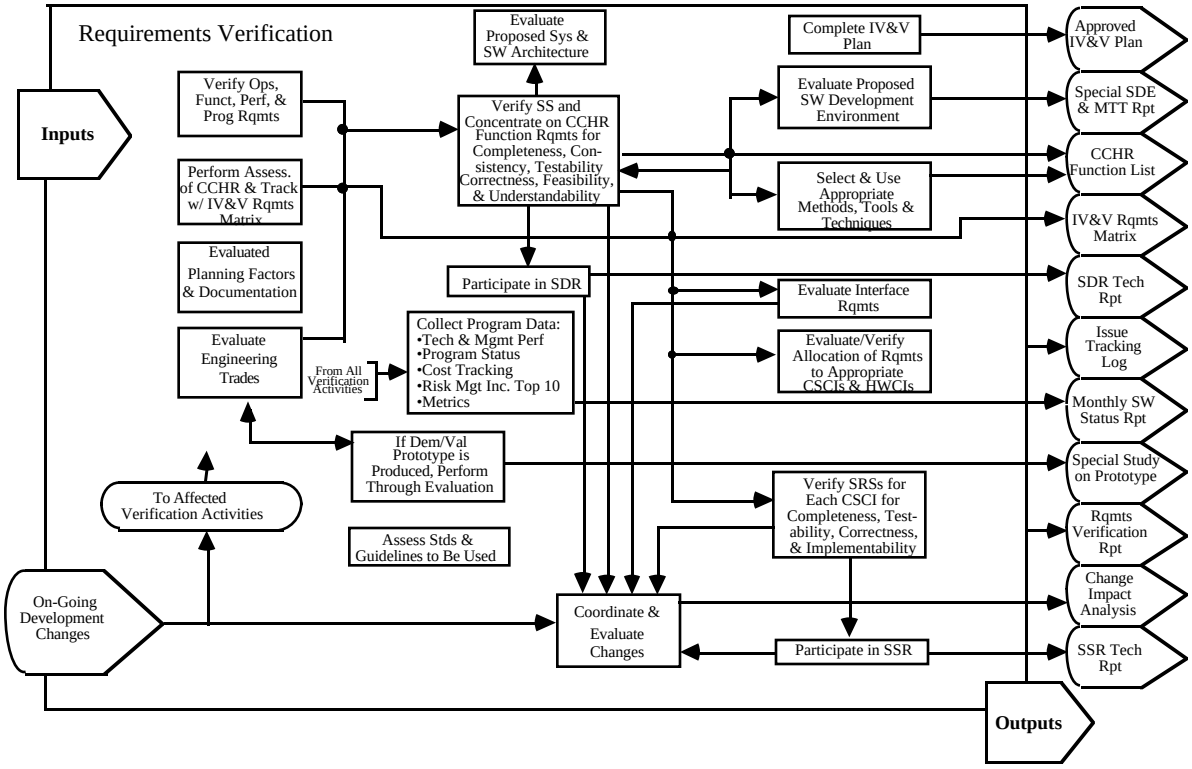

Figure 2: Lewis’s process model for IV&V in the Requirements Phase. Adapted from [7].

approach be adopted for all such requirements.

Having produced a clearer representation, the IV&V team then proceeded to identify properties of the specification that should hold if it is consistent and complete. A consistency property is that there should be no combination of conditions for which two different failure recovery actions are specified. A completeness property is that every possible combination of failure conditions should have some recovery action specified for it. These properties were tested by converting the tabular representations into a formal model (in this case SCR [11]), and using a tool to test for these properties. A significant number of consistency errors were found: there were combinations of conditions for which more than one recovery action was specified. These were traced to a problem with the ordering of the requirements. The correct functioning of the FDIR software depends on the tests described in these requirements being carried out in the order that the requirements are given. However, this is not stated explicitly. This finding confirmed an earlier informal observation by the IV&V team.

At this point the IV&V team would have gone on to check the validity of the requirements against a failure modes and effects analysis of the bus architecture. However, at this point in the case study the IV&V team found out that this section of the requirements was being substantially re-written. Hence they delayed further analysis until the new version became available.

This case study illustrates two interesting points about the work of an IV&V agent. First, the IV&V analysts often create their own representation (for example a formal model) of the developer's specification. However, these alternative representations are never given to the development team to use in place of the original specifications. This is to guard against the danger of the IV&V team being drawn into development work, and possibly losing their independence on subsequent analyses of these components.

Second, the IV&V team have their own discretion on how much analysis to perform. For example, the analysis does not stop when the first error is encountered. If there is an obvious fix to the error, it makes sense for the IV&V team to assume this fix will be made, and proceed with other types of analysis. However, there is a point beyond which further analysis adds little extra value; for example when major errors are encountered, or when, as in this case, a re-write is underway.

DIFFICULTIES IN IV&V

Having described the basic IV&V process, and illustrated it with a case study, we now discuss some of the difficulties faced by IV&V in carrying out their role. Our current research is investigating these problems, and seeking ways of overcoming them.

The following difficulties are inherent in the relationships between the developer, customer and IV&V agent. Some of these arise as a direct result of the conflicting goals of IV&V and developer; others are to do with resource pressures and the need for timely results:

- **resource allocation** - A complete, detailed analysis of the entire system is infeasible. Effort has to be allocated so as to maximize effectiveness. For example, a criticality and risk analysis might be performed to determine which components need the most scrutiny. Timing is also a factor; effort needs to be allocated at the right points in the development of a product (e.g. a document), so that the product is mature enough to be analyzed, but not so mature that it cannot be changed.
- **short timescales** - To be most effective, IV&V reports are needed as quickly as possible. There is always a delay between the delivery of an interim product to the IV&V team, and the completion of analysis of that product. During this time, the development process continues. Hence, if IV&V analysis takes too long, the results might be available too late to be useful. In general, the earlier an error is reported, the cheaper it is to correct, and the less reticent the developer is to fix it.
- **lack of access** - Contact between the development team and the IV&V team is difficult to manage. The IV&V team needs to maintain independence, whilst ensuring they obtain enough information from the developers to do their job. From the developers' point of view, interaction with the IV&V team represents a cost overhead, which can interfere with project deadlines. Inevitably, the IV&V contractor has less access to the development team than is ideal.
- **evolving products** - Documentation from the development team is usually made available to the IV&V contractor in draft form, to facilitate early analysis. The drawback is that documents may be revised while the IV&V team is analyzing them, making the results of the analysis irrelevant before it is finished.
- **reporting the right problems** - The IV&V contractor has, by necessity, considerable discretion over the kinds of analysis to perform on different products, and which problems to report. It is vital to the effective use of IV&V that the IV&V contractor prioritizes the problems it identifies. If too many trivial problems are reported, this may swamp the communication channels with the developer and the customer, and compromise the credibility of the IV&V agent.
- **lack of voice** - The IV&V contractor may have difficulty in getting its message across, especially if the development contractor disputes IV&V's assessment. Often, problems found by IV&V have cost and schedule implications, and in such circumstances the customer may be unwilling to listen. The effectiveness of IV&V then depends on having a credible advocate within the customer organization.

We discuss some of these problems in more detail below.

Coordination problems

In order to investigate these problems further, we developed a set of scenarios describing particular IV&V activities, and used these to explore where coordination problems occur. Easterbrook [12] describes these scenarios

in more detail. Here we present one of the scenarios, and discuss some of the problems it reveals.

Figure 3 shows a relatively formal process by which discrepancy reports originating from an IV&V analyst ("Carl") are entered into a database to be tracked, and a member of the development team ("Diane") is dispositioned to address the problem. In this case, the error is detected through an analysis similar to the one we described above for the case study.

From Carl's point of view, the process is as follows. Carl analyzes a section of the requirements document by generating a formal model of the section, and then running this through an automated consistency checking tool. The tool reports an inconsistency, which Carl traces back to a mistake in the original document. He writes a Discrepancy Report (DR). Let us call this DR#101. Three months later, a new draft of the specification is released. Carl checks to see which of his DRs have been addressed in this new draft. DR#101 is marked as having been worked on (by Diane), and is awaiting approval for closure. As the originator of the DR, Carl's signature is required. He updates his tabular representation to reflect the new draft, runs the new version of the model through the tool again, and confirms that the problem is now fixed. He therefore signs off the DR as closed.

The DR tracking tool avoids many potential communication problems, and ensures that closure is achieved for each reported problem. However, coordination problems can still occur.

For example, Carl could make mistakes in the translation from the text to the table - it is hard to confirm that the table is a faithful representation of the textual requirements. Similarly, Carl might not be able to trace the inconsistency back to the original requirements. He would then have great difficulty reporting the problem in a DR, unless he includes his tables, a description of the checking process, and some evidence that the tables are faithful to the original. This will make the DR rather cumbersome for a review panel to process.

Diane might not understand the problem. She might not be familiar with the tool that Carl uses. She might fail to correct the inconsistency, or might introduce more inconsistencies in this section. Carl may have problems updating his tables, perhaps because Diane (or someone else) has reorganized the section.

So, the error might still be there in the new draft. Worse still, it might be corrected in the new draft, and then reintroduced in subsequent versions, because some other change interferes with this one. Does this mean Carl has to update the table and re-run his checks again every time a changed draft is released? The problem here is one of representing relationships between the DR, the specification, and Carl's model. Tools that support traceability and configuration management help here, but do not guarantee that errors will not recur.

The scenario illustrates how expensive it can be to develop and maintain an alternative representation of an evolving specification. This may mean that this type of analysis gets delayed until the specification is relatively stable. This is undesirable.

Informal interactions

The process of tracking reported problems through a tool such as the DR tracker above is standard practice on most large projects. However, it is not the only means available for reporting and fixing problems, and there are good reasons why it will not be used for every problem. Consider an alternative scenario. Carl thinks there may be a mistake in the specification, but is not sure enough to

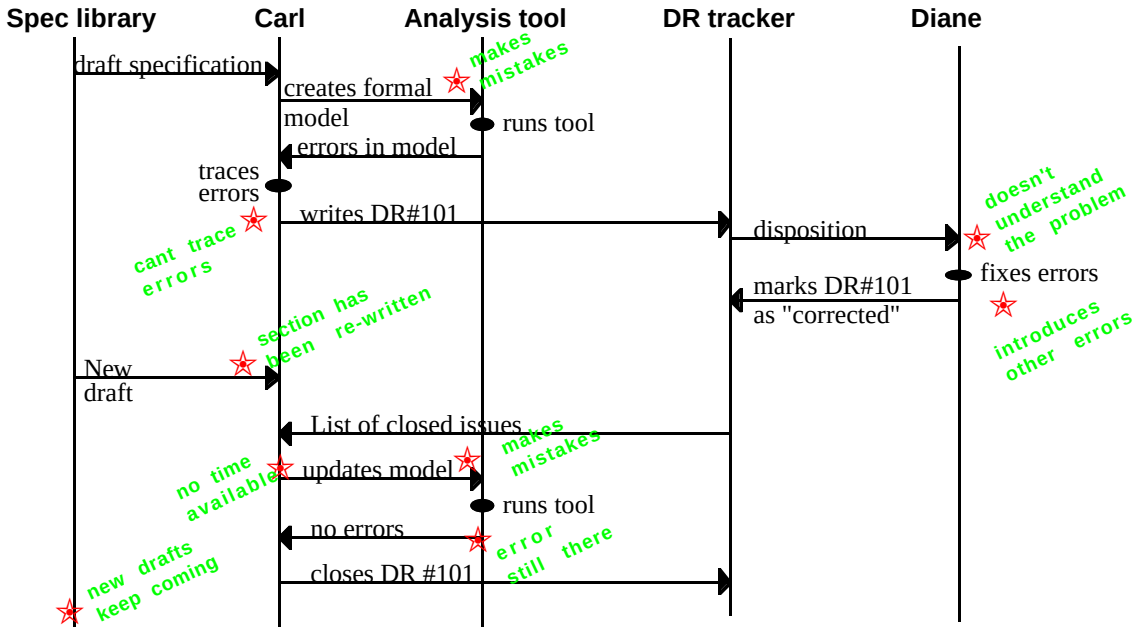

Figure 3. A possible scenario between an IV&V analyst (Carl) and a developer (Diane), annotated with potential problems

write a discrepancy report. So instead, he phones the author of the requirement in question ("Bob") and discusses it. In discussing it, they both realize it is a problem, and Bob says he can fix it immediately. Many minor problems are fixed in this way, by informal interaction between an IV&V analyst and his counterpart on the development team, without the overhead of writing discrepancy reports.

Unfortunately, there is a risk in using this alternative informal communication channel. If a face to face meeting is not readily available, it may take some effort to get in contact with the relevant member of the development team, during which time there is a danger that the problem might be forgotten. Communication could also fail because either Carl or Bob get transferred to another project or task.

The problem of other people making changes that interfere with this correction is even more acute here, as there is no record of the error that Carl and Bob identified, nor what change was made to address it. Other people have no visibility into the process that Carl and Bob went through.

The question of whether to formally track every single error discovered, no matter how minor, is a difficult one. As we have seen the tracking process removes some of the potential communication problems, but by no means all of them. It can be an expensive process to disposition discrepancy reports, follow up open issues, and ensure all are eventually signed off. It is possible to have several different issue tracking databases for a large project, with different levels of formality (and therefore cost) associated with them. However, this can introduce further problems in interactions between the different databases.

Conflict Resolution

In the scenarios above, we assumed that when Carl detects an error, the development team agree with his assessment. However, this is not always the case. The development team may disagree over whether a problem exists. Or they might agree that it exists, but refuse to fix it because of cost and schedule implications. Most importantly, it is possible that IV&V and developers disagree on whether it represents a safety risk.

A common reason for disagreements arising between the developer and IV&V team is that the resources are not available to address an issue raised by IV&V. In these cases, the IV&V team can act in the development team's interest, by taking the issues to higher levels of management, and pressing for more resources so that the development team can address the problem. The IV&V team may be more successful in securing the necessary resources than the development team would be if the request came directly from them.

In all cases of disagreement, there is a process for resolving the issue. Most of the problems identified by IV&V analysts are resolved by direct communication between the IV&V team and the development team. If they do not agree on the assessment of a particular problem, the IV&V team have to decide whether it is important enough to pursue further. If it is not, they still track it, to ensure

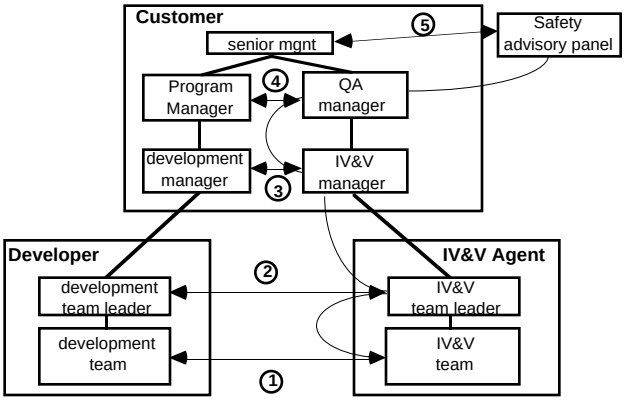

Figure 4: The confict resolution process.

it does not develop into a serious problem. If they do decide to pursue it, they will raise it with increasingly higher levels of management within the development and customer organizations (see figure 4), until they are satisfied that an appropriate response has been made, and that the relevant managers are aware of the problem.

Figure 4 shows a typical path that a controversial issue might take. If the IV&V team and development team cannot agree, then the IV&V team will alert their manager. If they decide the issue is important enough to pursue, then the managers within the IV&V contractor and development contractor will discuss it. If there is still no agreement at this level, then the IV&V contractor will request that the IV&V manager on the customer side raise it at the next Change Control Board (CCB) meeting. At this meeting, the IV&V team will present its case, describing the problem, the potential impacts, and recommendations. The development team then presents its view, including any steps they have taken to mitigate the risk. If the IV&V team are unhappy with the outcome, they can continue to pursue the issue at higher levels. Ultimately there are high level committees both inside and outside of NASA, such as the Aerospace Safety Advisory Panel (ASAP) who can continue to pursue the issue.

There are two main problems with this process. The first of these is the time it takes to pursue the issue through the various levels of management. At each stage there is a potential for delays of several months, whether intentional or not. In such cases, it is important that IV&V have a champion within the customer organization who can press for action to be taken as appropriate.

The second problem is one of presentation. Often the risk arising from a problem identified by IV&V is detailed and technical in nature, while the counter-argument is purely financial – it will cost money to mitigate the risk. Furthermore, higher levels of management may be unfamiliar with the technical details, especially when dealing with new technology. Within NASA, senior management often have little or no experience of software engineering, in which case the presentation by IV&V needs to include a substantial amount of tutorial material.

CONCLUSIONS

This paper has examined the role of IV&V in the software development process, concentrating especially on its role in requirements and design processes. IV&V provides an independent assessment of both developmental and operational risk. It helps to identify safety, reliability and performance concerns early in the software lifecycle, and has generally been demonstrated to save money though early identification of errors.

The role of IV&V is complementary to that of QA. Where QA focuses on checking that appropriate standards and process models are applied, IV&V focuses on the technical integrity of the software, through analysis of specifications, designs, code, and other documentation. Hence, IV&V will ensure that the requirements are complete, that a proposed system architecture will meet the requirements, and that traceability is demonstrated among requirements, designs and test cases.

An interesting emergent property of the IV&V process is that the IV&V agent can play a role as a process improvement agent, for a number of reasons. First, the recommendations made by IV&V in response to errors often address ways to prevent similar errors occurring in the future. Second, the IV&V team have some flexibility to apply new techniques and tools, especially where these plug perceived gaps in the analysis performed by the developer. If these new techniques and tools demonstrate their value in identifying errors, the development team may choose to adopt them themselves. Finally, the presence of an IV&V contractor provides an incentive for the developers to improve their own internal V&V practices, in order to catch errors before the IV&V contractor does.

Within the process model described for IV&V, there are still a number of problems. Some of these arise from the constraints imposed on the IV&V process. For example, the IV&V team need to devote resources to analyzing documents as soon as they are released, without necessarily knowing ahead of time what types of analysis will be needed. Other problems are due to the inherent conflicts between the goals of the developer and the IV&V contractor. For example, the developer may be reluctant to release draft documents to the IV&V team, for fear that the IV&V contractor will make them look bad by reporting large numbers of problems to the customer.

Although IV&V has been generally shown to have strong benefits, it is difficult to ensure that any particular IV&V effort will be effective. While an understanding of the processes involved is an important step, the biggest factor determining the effectiveness of an IV&V contract is the nature of the organizational relationship between the developer, customer and IV&V agent. Although the developer may be contractually obliged to work with the IV&V agent, this does not ensure a constructive relationship. Nor does it ensure that the customer will listen to the IV&V agent when the news is bad.

Research agenda

Our research to date has concentrated on observing the IV&V process, and identifying problem areas, as described in this paper. We are now focusing on potential solutions to the coordination problems illustrated by our scenarios. These problems are a result of the size of the documentation, and the fact that it continues to evolve. Tracing the effects of any small change to a specification is especially difficult. Where the IV&V contractor creates alternative representations for the developer's specifications, it can be difficult to ensure the alternative representations are faithful to the original. Furthermore, the models created by the IV&V team need to be updated whenever the documents on which they are based evolve. There is always a tension between the need to analyze drafts of the specifications in order to detect errors as early as possible, and the volatile nature of these early drafts.

We are also investigating the use of **l**ightweight formal methods for checking properties of partial specifications. The majority of work on formal methods assumes a commitment to the development of a complete and consistent formal specification. In contrast, when we use formal methods in the IV&V process, we are interested in developing just enough of a formal model to test particular properties. Problems here include the maintenance of fidelity between formal and informal representations of the same specification. Existing work on consistency checking does not help here, it generally assumes that consistency checks are being applied within a well-defined method, rather than between methods [13]. We are currently pursuing techniques based on behavioural analysis of formal specifications using model checkers.

ACKNOWLEDGMENTS

Thanks to Jack Callahan, Chuck Neppach, Dan McCaugherty, John Hinkle, and Gevony Laughlin for their input to this work, and to Edward Addy, George Saboloish, Butch Neal, and Frank Schneider for comments on earlier versions of this paper. This work is supported by NASA through cooperative agreement NCCW-0040.

REFERENCES

[1] J. L. Lions, *ARIANE 5 Flight 501 Failure: Report by the Enquiry Board*. European Space Agency, Paris 19 July 1996.

[2] G. G. Schulmeyer, The Move Towards Zero Defect Software. In *Handbook of Software Quality Assurance*, G. G. Schulmeyer and J. I. McManus, Eds., Second Edition. New York: Van Nostrand Reinhold, 1992.

[3] B. W. Boehm, *Software Engineering Economics*. Englewood Cliffs, NJ: Prentice-Hall, 1981.

[4] NASA, *Software Assurance Guidebook*. NASA Goddard Space Flight Center, Report SMAP-GB-A201, 1989.

[5] D. R. Wallace and R. U. Fujii, *Software Verification and Validation: Its Role in Computer Assurance and Its Relationship with Software Project Management Standards*. NIST Computer Systems Lab, Gaithersburg, MD, NIST Special Publication 500-165, 1989.

[6] N. G. Leveson, *An Assessment of Space Shuttle Flight Software Development Processes*. Washington DC: National Academy Press, 1993.

[7] R. O. Lewis, *Independent Verification and Validation: A Lifecycle Engineering Process for Quality Software*. J. Wiley & Sons, 1992.

[8] R. W. Butler and G. B. Finelli, The Infeasibility of Experimental Quantification of Life-Critical Software Reliability. *ACM SIGSOFT 91 conference on Software for Critical Systems*, New Orleans, Dec 4-6 1991.

[9] B. I. Blum, *Software Engineering: A Holistic View*. New York: Oxford University Press, 1992.

[10] Jet Propulsion Lab, *Cost-effectiveness of Software Independent Verification and Validation*. NASA JPL, Pasadena, CA, NASA RTOP report 1985.

[11] C. L. Heitmeyer, B. Labaw, and D. Kiskis, Consistency Checking of SCR-Style Requirements Specifications. *Second IEEE Symposium on Requirements Engineering*, York, UK, 1995.

[12] S. Easterbrook and J. Callahan, Independent Validation of Specifications: A coordination headache. *Proceedings, IEEE 5th Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'96)*, Stanford, CA, Jun 19-21 1996.

[13] S. Easterbrook and J. Callahan, Formal Methods for V&V of partial specifications: An experience report. *Proceedings, Third IEEE Symposium on Requirements Engineering (RE'97)*, Annapolis, MD, 5-8 Jan, 1997.