In: Proceedings of the Fifth Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'96), Stanford, California, June 19-21, 1996. IEEE Computer Society Press. Pp232-237.

Independent Validation of Specifications: A coordination headache

Steve Easterbrook
steve@atlantis.ivv.nasa.gov

John Callahan
callahan@cerc.wvu.edu

Software Research Lab
NASA/WVU IV&V Facility
Fairmont, WV 26554

Abstract

Large, complex projects face significant barriers to coordination and communication due to continuous, rapid changes during a project's lifecycle. Such changes must be tracked, analyzed, and reconciled to ensure high-quality in the end-product, otherwise problems may be get lost or ignored in the overall complexity. We report on "work-in-progress" in the study of coordination problems between two independent, separate groups: software development and software analysis. We have begun to construct a taxonomy of coordination problem, which we illustrate with two scenarios. We briefly describe current attempts to introduce incremental improvements to coordination problems in such projects via World-Wide-Web tools. Based on actual project experiences, we plan to deploy such tools in a non-intrusive fashion to improve coordination and communication between software development groups.

1. Introduction

This paper describes some of the coordination problems faced in the verification and validation (V&V) of large software systems. In particular, we examine the process of Independent V&V (or IV&V), a practice used on large, safety-critical software systems in the defense and aerospace industries. IV&V faces many of the same coordination problems as any large development project, and introduces many problems of its own. In particular, IV&V requires two organizations with conflicting goals to cooperate to satisfy a single customer. Not only is there a lack of suitable tools to support the coordination needed, but often even the infrastructure in which such tools would be used is missing. We will illustrate some of the problems faced by IV&V through two scenarios, and describe our initial work on the development of web-based tools that address the problems.

Although we focus on IV&V, many of the coordination problems are found in any large project. In many projects, coordination problems surface merely as tensions and frustrations, perhaps leading to schedule and budget problems. In an IV&V process, the coordination crosses organizational boundaries. This makes it harder for the people involved to find ways to work around any lack of cooperation. Hence, the problems are more visible, and easier to study.

Within the IV&V process, we concentrate on the analysis of requirements specifications. This is a deliberate choice: independent analysis of requirements specifications has the potential for the biggest impact of any IV&V activity. The quality of the analysis of design and implementation and the rigor of testing depend to some extent on the availability of good quality specifications.

In this paper, we identify the interdependence of large volumes of documentation as a key problem in software specification. Any change to one part of a specification may have many small impacts ("ripples of influence" [6]) throughout the documentation, which may be hard to track down. Certainly this kind of problem was an important factor in the fatal decision to launch Challenger [1]. The mechanical problem that caused the blast was well known before the accident; however, failures in tracking the problem and in keeping all documentation consistent led to faulty decision making by NASA managers.

In this paper, we outline the IV&V process and identify a number of coordination problems that we have observed in actual IV&V processes. Specifically, we draw on our informal observations of and interactions with IV&V personnel working on the Space Station and the Earth Sciences Data and Information System (ESDIS) projects. We present two scenarios, which reveal the extent to which coordination problems can reduce the effectiveness of IV&V. We then describe our initial work with the World Wide Web to provide infrastructure and tools to overcome some of these problems.

2. Coordination Problems in IV&V

Independent Verification and Validation (IV&V) is a process in which 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 [2]. As the IV&V agent has no stake in the success or failure of the software, it can provide an unbiased assessment of the status of a project's schedule, cost, and the viability of its product during development. IV&V provides an early warning of problems, and helps to identify and manage both technical and programmatic risks.

An example of an IV&V activity is the analysis of specifications on the Space Station project. An SRS is written by a development contractor for each Software Configuration Item (CSCI). These are 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, and analyses them for technical integrity, to identify requirements problems or risks. The kind of analysis performed will vary according to the level and the type of specification, and will cover issues such as clarity, testability, traceability, consistency and completeness. If problems are identified, the IV&V contractor may recommend either that the requirements be rewritten, or that the problem be tracked through subsequent phases.

There are a number of tensions that interfere with the smooth running of an IV&V contract. The development contractor and IV&V contractor have 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. The extent to which this conflict causes problems depends very much on the attitude of the project managers within the development contractor. If they regard 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 the relationship between developer and IV&V.

Within this context, there are a number of factors that affect the relationship:

- Timing - IV&V needs to report problems as early as possible. In general, the earlier a problem is detected, the less it costs to fix. Also, by the time IV&V produces its reports, the reports might be irrelevant: the product may have changed, or errors may have already been fixed. Delayed reporting by the IV&V agent can sour the relationship with the developer.
- Access - In order to do its job, IV&V needs timely access to a large amount of project data and documentation. Preferably, it needs to receive early drafts of documents, as well as regular updates. Developers are naturally unhappy about releasing drafts to an IV&V agent: draft documents are full of known problems. The development contractor needs to be able to trust the IV&V contractor to handle draft documents sensitively.
- Evolution - IV&V analysis and development effort proceed in parallel, so that effectively IV&V is dealing with a moving target. Even if the IV&V agent can obtain regular updates of a product, these updates only represent snapshots, and do not capture the evolution of the product. Hence, problems may recur or disappear, without any explanation.
- Appropriateness - When reporting problems to the customer, the IV&V contractor needs to be careful to match the report with the status of the product it applies to. For instance, if a document is an early draft, then it should not be treated as a finished product: the IV&V agent need not report problems that can be reasonably expected to be fixed by the developer in due course.

Central to the work of IV&V is the problem of issue tracking. Problems that are important enough to be reported to the customer need to be tracked so that responses can be assessed. Minor problems need to be tracked in case they become major. While formal issue tracking processes (and tools that support these processes) help, they do not address the need to trace dependencies and decision rationale.

Many of these problems are common to all large projects: in an IV&V project the problems are more obvious because they cross an organizational boundary.

3. Scenarios

This section presents two scenarios, drawn from our observations of IV&V processes. In each case, we present a normal process, and then discuss potential communication and coordination problems. Our aim is to use these scenarios to understand how better to support the process.

3.1. Scenario 1

An IV&V analyst (let us call her Alice) is reading a specification, and notices a requirement that begins *"Within 110ms of ..."*. The figure of 110ms seems wrong, but she cannot check it immediately as it will require a calculation using data available elsewhere.

A typical, informal process is illustrated in figure 1. Briefly, Alice makes a note to herself to come back and

check the figure, which reaches the top of her to-do list after a couple of weeks. She then makes the calculation, and confirms that the figure is indeed wrong. She phones the author (let us call him Bob) of the section of the document in which this requirement occurs, and leaves him a voice mail message explaining the problem.

Bob gets the message, checks the relevant section, and concludes that Alice's assessment is correct. He phones Alice to say he will fix it before the next draft of the document. Alice makes a note to remind herself to check that the fix has been made when she gets the next release. Two months later, she receives the new release, runs through her list of things to check, and finds that the figure has been corrected.

Now look at what could have gone wrong. First, any of the communications in figure 1 could have failed to occur. For example, Alice might have failed to make a note to herself, she might not be able to trace the author of the document, Bob might never return her call, and so on. Communication might also fail because either Alice or Bob leave the company, or move to a different project, at any point during the scenario. Figure 1 is annotated with these possible communication failures.

Second, there are a number of places where coordination problems may occur. For example, Bob may disagree with Alice's assessment of the problem. This might be because either he or Alice made a mistake in their calculations, because Bob misunderstands Alice's message, or because he looks at the wrong section. A decision must be made over whether the issue is important enough to pursue further: Alice may decide to write a formal discrepancy report.

Other changes to the specification may also cause problems. For example, someone else might make conflicting changes to the same section of the specification, they may delete the whole section, they might change the figures on which Alice's calculation was based, or they may undo Bob's change. This is especially a problem if other people have no visibility into the process that Alice and Bob went through.

Finally, if everything else goes well, the figure might still be wrong in the new draft. This could be because Bob changed it but still got it wrong, because he changed the wrong figure, because he never got round to it, or because the same problem recurs throughout the specification and Bob did not track down all of them.

Some of the problems illustrated in this scenario could be addressed with an issue tracking system, and our next scenario shows the use of such a system. However, for this first scenario, an issue tracking system might be too formal. If Alice can resolve the problem by picking up the phone, she may be reluctant to incur the overhead of entering the problem into an issue tracking system.

The scenario also illustrates the inter-relatedness of different parts of the specification. When Alice checks the figure, she uses data from elsewhere in the documentation. She knows what data to use from her familiarity with the specifications: nowhere is it recorded that the data she uses is related to the figure she is checking. Alice may or may not be the first person to notice the relationship. In either case, once the relationship is noticed, it ought to be recorded for future reference. With a little intelligence

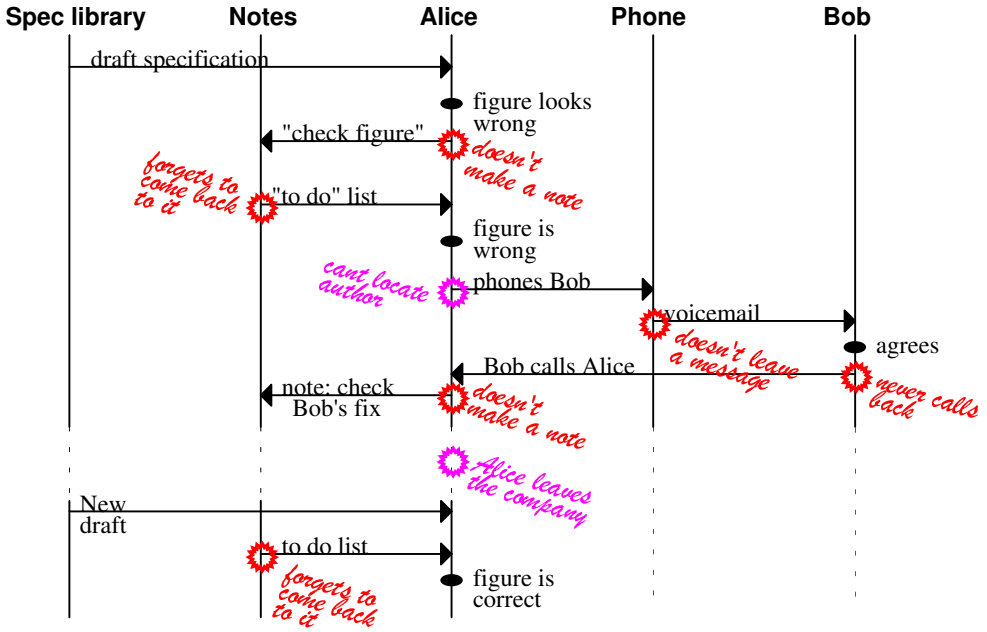

Figure 1. The first scenario, annotated with potential communication problems

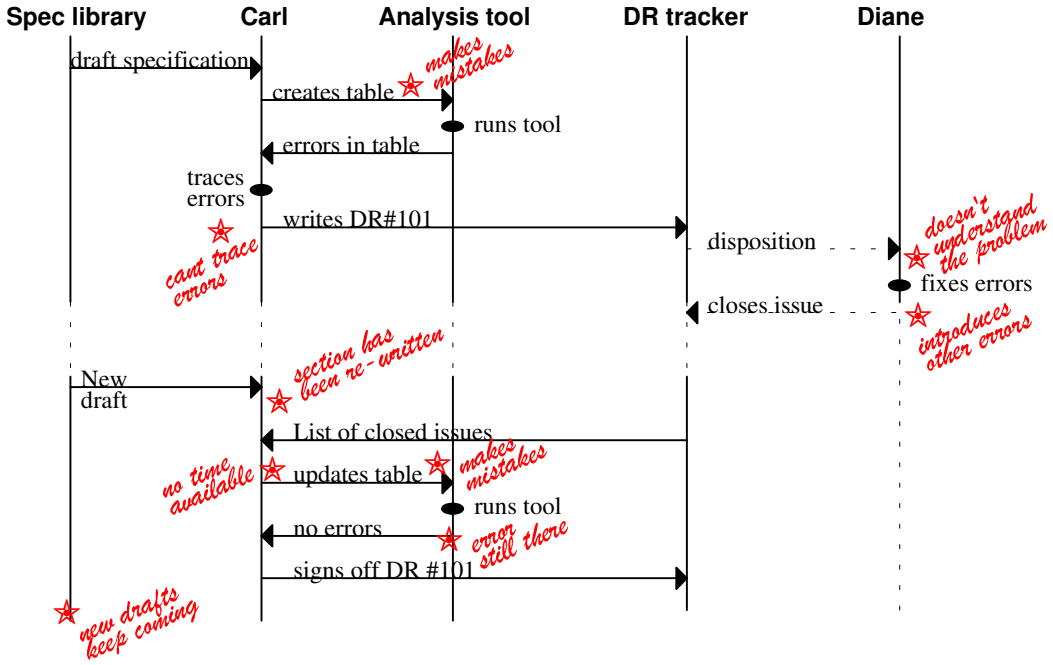

Figure 2. The second scenario, annotated with potential coordination problems

built into support tools, it should be possible to detect that Alice uses some parts of a spec to check others, and hence record a dependency relationship. Section 4 discusses our current project to introduce such support tools incrementally into ongoing projects.

3.2. Scenario 2

The second scenario (Figure 2) illustrates a more formal process, in which fewer communication problems can occur. An IV&V analyst (let us call him Carl) is analyzing a section of the requirements document by generating a tabular version of the section, and then running the resulting tables 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 the DR database, 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 originator of the DR, Carl's signature is required before the issue can be closed. He updates his tabular representation to reflect the new draft, runs the new version of the table through the tool again, and confirms that the problem is now fixed. He therefore signs off the DR as closed.

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

For example, Carl could have made 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. Carl might not have the time to update the tables and so just performs a visual check, in which case he might not notice if the correction introduced any new errors.

Finally, the process might need to be repeated indefinitely. Diane's changes might not have corrected the problem, or other people might make further changes after the DR is signed off. 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 to do with the relationship between alternative representations of the same information. Currently, such a relationship exists only in Carl's head - there is no representation of the

relationship anywhere else. Either of the two chunks of specification may evolve, but there is no way to trace the "ripples of influence" of any changes. Hence, there is no opportunity for tool support to reason about how changes to one side affect the other.

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.

Notice that the relationship is bi-directional. Although the table is generated and updated from the text, Carl needs to be able to trace problems from the table back into the text. It is also highly likely that Carl may want to alter the table to see what possible fixes there are, and then see what effect this has on the text.

4. The Web as an enabling technology

The problems identified in the scenarios show how time-consuming and costly it can be to track changes, especially where there are many dependencies throughout the specifications. A full solution to these problems would require all dependencies between different parts of a specification to be explicitly represented. Such a solution requires significant advances in the capture and representation of dependencies between specification elements. Partial solutions exist for individual methods (e.g. the consistency checking for SCR [3]). A general solution for multiple methods is still a long way off.

To explore such a general solution, we have adopted an incremental, empirical approach. We need to put into place the infrastructure for recording data about each chunk of specification, including annotations and relationships with other chunks. However, we also need to integrate this infrastructure with the existing project support systems on the projects we wish to study, to minimize the disruption caused. We will not be able to proceed with our empirical study unless each step is relatively painless for the project.

The infrastructure we need to put in place must satisfy two major criteria. It must be adaptable enough to fit in with a wide range of existing project support tools on different platforms, using heterogeneous networks, and accessing existing project data in a variety of different formats. Second, it must provide the ability to record and track arbitrary relationships between chunks of specifications [4].

Initial experiments indicate that the World-Wide-Web will satisfy most of the first criteria. In particular:

- It removes dependence upon any one platform, and allows us to design tools that can immediately run on any platform without reconfiguration
- It hides networking details, so that remote access to documents is as straightforward as local access.
- It is extensible via Java, so that tool development is feasible.
- It provides a basic hypertext functionality.
- Most project documentation on the projects we wish to study is already available in electronic format, and can easily be converted to HTML.

Despite these benefits, a number of problems remain. Firstly, the hypertext model provided by HTML is far too simplistic. Links are unidirectional, and encode no semantic information. In general, links are hardwired into documents, although Javascript makes dynamic links feasible. Navigational aids are still relatively primitive, limited to path tracing and keyword searching; no tools yet exist for integrating graphical views of hypertext structure. Documents themselves are static: the web is a passive browsing mechanism - browsers cannot edit or annotate documents. Editing of documents normally takes place offline.

Another serious shortcoming of the Web is the inability to represent the structure of non-textual documents (e.g. diagrams). Hence manipulation of structured documents can only be achieved by using extensions to web browsers.

5. The WHERE project

The Web-based Hypertext Environment for Requirements Engineering (WHERE) project is an experiment in extending the capabilities of the Web, using viewpoints [7] instead of web pages as the basic building block. viewpoints combine the idea of an actor, role or agent in the development process, and the idea of a view or perspective that an actor maintains. In software terms, viewpoints are loosely coupled, locally managed, coarse-grained objects which encapsulate a partial specification in a suitable representation scheme, and partial knowledge of the process of development.

The Viewpoints framework divides a specification into manageable chunks ('viewpoints'), each of which has the following attributes:

- a representation style, the scheme and notation by which the viewpoint expresses what it can see;
- a domain, which defines the area of concern addressed by the viewpoint;
- a work plan, which comprises the set of actions by which the specification can be built, and a process model to guide application of these actions;
- a work record, which contains an annotated history of actions performed on the viewpoint.

Each viewpoint has a defined owner. The owner is responsible for developing a viewpoint specification using

the notation defined in the style slot, following the strategy defined by the work plan, for a particular problem domain. A development history is maintained in the work record. This framework encourages multiple representations, and is a deliberate move away from attempts to develop monolithic specification languages. It is also independent from any particular software development method.

The WHERE project will implement this framework using the Web. The core functionality will be provided by three Java applets:

The Viewpoint Editor is a configurable specification editor. The editor is configured from one of three templates: depending on whether the notation to be edited is graphical, tabular, or textual. Each template takes a syntax description as a parameter, to provide a syntax-directed editor for the given notation. Note that the editor will not need to be able to generate large specifications, as a specification is broken down into individual viewpoints. For example, if a specification method calls for three different types of table, each table will be represented as a different viewpoint, and three different configurations of the tabular viewpoint editor will be needed, one for each table type.

The Viewpoint Reviewer allows a user to browse and annotate viewpoints created by other people. If the viewpoint to be loaded was developed outside the WHERE environment, the Reviewer will provide a rudimentary parsing of the viewpoint, so that annotations can be attached to different parts of its structure. The reviewer will not allow a user to edit the viewpoint, and will store annotations and meta-viewpoint data as a separate 'view' of the viewpoint.

The consistency checker allows the user to define and check relationships between viewpoints. Each viewpoint can have three types of relationships with other viewpoints: method-defined, user-defined, and emergent. Method-defined relationships describe relationships that *should* hold between two viewpoints of a particular type. User-defined relationships are entered by users to record and track non-standard relationships between particular viewpoints. Emergent relationships are recorded as the result of certain actions on viewpoints, where the action reveals that a relationship must exist.

The tools can be used to provide support for the entire requirements specification process, or as a partial aid to some aspects of it. The partial mode allows a gradual introduction of the tool into existing projects. For example, existing specification documents can be loaded into the viewpoint reviewer and annotated. New viewpoints can be created using the viewpoint editor, with defined relationships to the existing documentation. A typical use would be to model a portion of an existing specification in a new notation, while explicitly recording relationships with the existing specification.

6. Conclusions

Our approach to empirical investigation and incremental tool deployment is based largely on the Experience Factory model [5] that has been successfully employed on other NASA software efforts. Based on the experiences with IV&V analysts using our tools, we will continue to refine and add features.

Technologies like the World-Wide-Web and Java are essential enablers that will allow software developers to manage the volume of change of large, complex software projects. We have demonstrated briefly how failures in coordination and communication can wreak havoc on such projects *even in situations where potential problems are identified early in the project's lifecycle*. The use of an independent analysis team, such as an IV&V group, is highly effective, but requires additional coordination mechanisms to reap the benefits of such analysis.

7. Acknowledgments

We acknowledge the assistance of Chuck Neppach and Dan McCaugherty (Intermetrics) for discussions regarding IV&V pragmatics and Darryl Lakins and George Sabolish (NASA) for insight into IV&V effectiveness issues. We thank our students, Amer Al-Rawas, Swarn Dhaliwal, Zhong Zhang and Gevony Laughlin for their inputs to this work.

8. References

[1] N. G. Leveson *"Safeware: System Safety and Computers"*, Reading MA: Addison Wesley, 1995

[2] NASA *NASA Software Assurance Guidebook,* NASA-GB-A201, prepared by the Software Assurance Technology Center, Goddard Space Flight Center, 1989

[3] C. Heitemeyer, B. Labaw and D. Kiskis, "Consistency Checking of SCR-Style Requirements Specifications", *Second IEEE Int. Symp. on Requirements Engineering,,* York, UK, March 1995, pp56-63.

[4] S. M. Easterbrook and B. A. Nuseibeh, "Using ViewPoints for Inconsistency Management", *BCS/IEE Software Engineering J.,* 11(1), Jan 1996.

[5] V. Basili, "The Experience Factory and its Relationship to Other Improvement Paradigms", *Proc, 4th European Software Engineering Conference,* Garmish-Partenkirchen, Germany, September 1993.

[6] M. Klein, "Core Service for Coordination in Concurrent Engineering", *Proc. 4th IEEE Workshop on Enabling Technologies: Infrastructure for Collaboration Enterprises,* Coolfront, West Virginia, April 1995.

[7] A. Finkelstein, J. Kramer, B. Nuseibeh, L. Finkelstein and M. Goedicke, "ViewPoints: A Framework for Multiple Perspectives in System Development", *Int. J. of Software Eng. and Knowledge Eng.* 2(1) Jan 1996, pp31-5.