SCR as an IV&V Tool *

Steve Easterbrook and John Callahan
{steve,callahan}@cerc.wvu.edu

NASA/West Virginia University Software IV&V Facility
100 University Drive
Fairmont, WV 26554
304-367-8235, 304-367-8211(fax)

February 5, 1996

Abstract

1 Introduction

This paper describes some preliminary work on the use of SCR-style specification as a tool for Independent Verification and Validation (IV&V). Our intention is to use formal methods not as a part of the development process itself, but as a 'shadow' activity, performed by an independent team of experts. Our long-term expectation is that this approach will turn out to be a less painful way of introducing formal methods into well-established, large-scale software development processes. However, there are a number of problems that need to be solved before SCR-style methods can be used in this way. The most important of these is the need to model the relationship between informal and formal specifications.

The context for this work is the development of software for the International Space Station (ISS) project. Boeing Space and Defense Group Houston (Prime) is responsible for supervising the overall development and integration of International Space Station software. There are three Product Groups (PGs), McDonnell Douglas Aerospace, Rockwell Aerospace - Rocketdyne and Boeing Space and Defense Group Huntsville, who are developing several key Computer Software Configuration Items (CSCIs), which Prime is responsible for integrating. There are also several International Partners (IPs) including Russia, Japan, Canada, and the European Space Agency, who are developing software that will need to be incorporated into ISS. With over 45 flight computers and an estimated 1.1 million source lines of flight code, the potential problems are considerable. Software IV&V is currently being performed by Intermetrics, under an interim contract. The Intermetrics team is based at Fairmont, W.Va., with personnel stationed in Houston and Huntsville in order to interact with the development teams.

In this paper, we firstly describe the IV&V process, and discuss some of the aspects of this process that hinder effective IV&V. With this as background, the remainder of the paper focuses on the use of methods and tools within this process. We describe how the use of a method in an IV&V process differs from its use in development work, and list some of the concerns that arise when a new method is adopted in IV&V. We present a small experiment in the use of SCR-style tables. The experiment showed that the natural language used in the Software Requirements Specification (SRS) documents is inherently ambiguous, and that the

*This work is supported by NASA Cooperative Research Agreement NCCW-0040

future task of generating formal specifications from this documentation will be fraught with difficulty. We conclude that in an IV&V context, the analytical benefits offered by methods such as SCR have to be weighed against the effort needed to maintain fidelity between the formal model and the informal specification used by the development team.

2 The IV&V Process

Verification and validation (V&V) is a widely-practiced discipline that employs many types of analysis to ensure that software development efforts are successful. V&V employs many techniques throughout the software lifecycle, including analysis of requirements, design, code, performance, schedule, and cost, as well as testing. V&V is performed in-house by the development contractor, but may be the responsibility of an entirely separate team from the one developing the software. In *Independent* Verification and Validation (IV&V), the software customer hires a separate contractor to perform V&V analysis on the products and process of the software development contractor [9]. This analysis is performed in parallel with the development process, and in no way replaces the in-house V&V done by the development contractor. IV&V is applied in high-cost and safety-critical projects to reduce 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. For true IV&V, the IV&V contractor has managerial, financial and technical independence, and reports to the customer, not the developer.

An example of an IV&V activity is the analysis of specifications on the Space Station project. An SRS is written by the relevant 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. These are analysed 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, 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.

Performing IV&V on large projects is far from straightforward. Problems faced by the IV&V contractor include:

resource allocation – A complete, detailed analysis of the entire system is infeasible. Effort has to be allocated so as to maximise effectiveness. For example, a criticality and risk analyses 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 analysed, 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 detected, the cheaper it is to correct.

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 analysing 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 what kinds of analysis to perform on different products. It also has discretion over how it reports problems. 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.

These problems will continue to act as barriers to the effective application of IV&V to major, high-cost, safety-critical projects. But it is the *interaction* between the IV&V and development teams that drive improvements in both products and processes. Section 3 discusses IV&V's role as an agent of process improvement in the context of applying new tools and techniques to analyze computer software. Section 4 discusses an example of this approach through an experiment using tabular requirements in IV&V analysis efforts.

3 Methods and Tools in the IV&V process

An important aspect of IV&V work is the choice of the right methods and tools. Ideally, an IV&V contractor will have access to all the tools used by the development team, including the ability to share all project databases. However, the IV&V team also needs to supplement these with additional methods and tools, to address any gaps or weaknesses in the coverage of the developer's tools. These additional tools need to complement the developer's tools, so that interoperability does not become a problem. The use of these additional tools is an important factor in ensuring that IV&V is truly independent.

It is often the case that the use of a particular method or tool by the IV&V team leads to the adoption of that method or tool by the developers. In part this is due to the 'watchdog effect': if the developer knows that their product will be analysed in a particular way, it is in their interest to perform the analysis themselves before releasing it. If this seems to be a rather negative reason to adopt a technique, there is also a positive aspect. Because the IV&V team is out of the critical path for the software development effort, they have more scope for experimentation with new techniques than the developers [1]. Hence, in some ways the IV&V team has a role to play as a proving ground for new techniques, and can come to play a role as an agent of process improvement.

For these reasons, we believe that IV&V offers a practical route through which formal methods may be introduced on projects that would otherwise not be able to adopt them. The interaction of IV&V with the development team can drive improvements via the 'watchdog effect'. Problems identified by the IV&V team precipitate changes in the development process, as well as the product. Thus, communication between the IV&V team and the development team can drive development activities, decisions, and priorities. Currently, we are investigating properties of this bipartite model [2] as a means of coordinating design evolution and prototyping in rapid software development environments.

There are still problems to be overcome whenever the IV&V team adopts a tool that is not used by the developers. Compatibility with the developers' tools is important. For example, if IV&V use a formal specification tool, the informal specification delivered by the developers will need to be translated into the formal specification language not just once, but each time the developers produce a new draft. Any problems identified by using the tool must be traced

back to the informal specification, before they can be reported. There must be a reasonable assurance that the formal specification remains faithful to the original, otherwise any analysis performed on it is worthless. Hence, keeping track of the relationship between the formal and informal specifications is vital.

4 Experiments with Tabular (SCR-style) notations

Currently, the development contractors on the Space Station project do not make use of any formal specification methods. We are working with the IV&V team to explore how formal methods can be used to enhance the kinds of analysis they perform.

Because the IV&V process is parallel and separate from the development process, there is an unusually large amount of flexibility in how a formal method can be used. In a development process, a full commitment is required. For example, if a formal specification method is adopted, it must be used to generate and analyse complete specifications for whichever components it is used on. If this cannot be achieved with the method, it must be achieved without the method, if development is to proceed. In contrast, the IV&V activity is not in the critical path, and can experiment with partial specification and partial analysis. In fact, the aim of the IV&V agent is not to perform complete analyses, but to do just enough analysis to check specific aspects of the software.

Our experimentation with SCR has exploited this flexibility. Our first step was simply to translate some of the more opaque and ambiguous natural language requirements into a formal (tabular) notation, in order to improve understanding of them. This turned out to be a useful step in assessing the quality of the specification, and confirmed some of our suspicions about ambiguity in the original specification.

The second step was to perform a limited set of analyses on those parts of the specification that we had represented formally. In particular, the section of the specification on which we concentrated described an implicit fault recovery model, by describing the conditions under which each action needed to be taken. The formal notation made a completeness and consistency analysis possible.

The final step, that of producing a full formal model for the subsystem has not yet been taken. A number of factors indicate that we will have to take this step, not least because the IV&V team has had difficulty obtaining the original models on which the specification is based. Hence, a formal model should provide an alternative route by which to check the fault recovery requirements. However, a number of questions need to be addressed first, including the amount of effort required to produce and maintain the formal model.

We describe each of these steps in more detail below.

4.1 Partial translations

Our initial interest in SCR-style tabular notations was to reduce the ambiguity of the natural language Software Requirements Specifications. In particular, the Fault Detection, Isolation and Recovery (FDIR) requirements were difficult to understand. The IV&V team needed an alternative, clearer representation. Typically, these requirements list a combination of failure conditions that require a particular recovery action. A particularly complicated example is:

> (2.16.3.f) While acting as the bus controller, the C&C MDM CSCI shall set the e,c,w, indicator identified in Table 3.2.16-II for the corresponding RT to "failed" and set the failure status to "failed" for all RT's on the bus upon detection of transaction errors of selected messages to RTs whose 1553 FDIR is not inhibited in two consecutive processing frames within 100 millisec of detection of the second transaction error if; a backup BC is available, the BC has been switched in the last 20 sec, the SPD card

		OR			
	C&C MDM acting as the bus controller	T	T	T	T
	Detection of transaction errors in two consecutive processing frames	T	T	T	T
	errors are on selected messages	T	T	T	T
	the RT's 1553 FDIR is not inhibited	T	T	T	T
	A backup BC is available	T	T	T	T
A	The BC has been switched in the last 20 seconds	T	T	T	T
N	The SPD card reset capability is inhibited	T	T	-	-
D	The SPD card has been reset in the last 10 major (10 second) frames	-	-	T	T
	The transaction errors are from multiple RTs	T	T	T	T
	The current channel has been reset within the last major frame	T	F	T	F
	The bus channel's reset capability is inhibited	-	T	-	T

Table 1: A Leveson-style table for requirement 2.16.3.f.

reset capability is inhibited, or the SPD card has been reset in the last 10 major (10-second) frames, and either:

1. the transaction errors are from multiple RT's, the current channel has been reset within the last major frame, or
2. the transaction errors are from multiple RT's, the bus channel's reset capability is inhibited, and the current channel has not been reset within the last major frame.

The IV&V team suggested that requirements of this form be represented in a tabular notation, similar to that adopted by Heimdahl and Leveson [6] in their analysis of TCAS II. These are essentially truth tables, used to represent the combinations of conditions that lead to a single mode transitions. This notation allows us to represent arbitrary combinations of conjunctions and disjunctions, without ambiguity. For example, the conditional part of the above requirement could be represented as shown in Table 1.

For the IV&V team, this was a significant improvement in readability. More importantly, the process of producing the tables ensured that the analysts fully understood the requirement. This benefit is very important for IV&V. In many cases, just reading a specification is insufficient to really appreciate the detail. Short of repeating the development process from scratch, it can be hard for the IV&V to understand a specification in the same way that its authors understand it. Translating it into a table, however, proved to be a valuable clarification process.

There was, unfortunately, a problem. Translation of a single requirement, like the one above, was not a straightforward task. As an experiment, we gave the English language version to four different people, all of whom had some experience of representing requirements using tables, and asked them to produce the tabular form. We received four different answers, which differed in both the number of conditions identified (i.e. number of rows in the table) and the number of combinations under which the function would be activated (i.e. columns in the table). The version shown in Table 1 is a synthesis of the four answers, representing what we currently believe is the intended interpretation.

The differences in the responses show that the original requirements statement is riddled with ambiguities. For example, the mixture of 'ands' and 'ors' in the requirement is a problem because, unlike programming languages, English does not have any standard precedence rules. It is not clear how to scope the various subclauses, either. For example, the timing condition 'within 100 millisec...' could refer to the inhibition of the FDIR, or to one or both of the required setting operations. With some domain knowledge, it is possible to guess the most likely interpretation, but this is by no means a trivial task, and there is no guarantee that everyone who needs to read this requirement will get it right.

The experiment demonstrated three important results. Firstly, as expected, the tabular forms are a precise, readable alternative to natural language. In fact, they were very helpful in resolving misunderstandings. For example, it would be difficult to discover that our four subjects had different interpretations of the original requirement without asking them to re-write it. By re-writing it in tabular form, we could identify exactly where the disagreements lay, and then take each discrepancy in turn and discuss what we thought the most likely interpretation was. From this, we were able to synthesis a 'best' interpretation. Note that this final version was different from all four of the individual versions, implying that if the final version is correct, all four individual attempts were wrong!

This leads to the second result, which is that translation of informal requirements into a formal notation is error prone. All four of our subjects had some experience of using such tables, so the problem lies not in the correct use of the notation, but in the interpretation of the informal statement of requirements. The requirement we used in the experiment is perhaps an extreme example, given its rather convoluted English. However, there is enough scope for misinterpretation in the process of formalising the requirements to cause us to worry about the fidelity of our formal models.

The third result is that the whole process was remarkably good at identifying ambiguities in the original specification. By producing different interpretations and comparing them, we were able to identify a systematic pattern of ambiguities in the way the English language requirements were written. Hence, even if we fail to persuade the development team to adopt a tabular notation, we can at least help them to correct the ambiguities in the English.

In fact, the development contractors have used the tabular notation occasionally, in the most recent versions of the specifications. Initially, they resisted the IV&V team's requests to adopt a tabular notation, largely because of schedule constraints. They have now begun to use the notation for revisions of the specifications, especially in areas where reviewers had had problems with readability. We regard this as a small but important process improvement, inspired by the IV&V team.

4.2 Partial analysis

The second stage of our work was to attempt some formal analysis of the tables. One of the important validity checks for these requirements is that an action is specified for each possible combination of failure conditions. Another check is that no combination of conditions has conflicting actions specified for it. We refer to these as coverage and disjointness checks respectively [10].

At this stage we had six tables, similar to the one shown in Table 1, representing the six paragraphs, a to f, of section 2.16.3 of the requirements. There were a number of conditions common to several of the tables. Unfortunately, the wording varied, and it was not always obvious whether similar sounding phrases actually referred to the same condition, due to inconsistencies in the use of terminology. For example the condition "the bus has been switched in the major (10-second) frame" appeared in one paragraph, and "the bus has been switched in the last major frame" appeared in another. We initially assumed these to be identical. However, this led to an inconsistency in the table. In fact the former refers to the *current* frame, while

the latter refers to the *previous* frame. There were numerous places where we had to make assumptions to proceed, and we carefully recorded these as annotations to the original text, to be checked with the developers.

The resulting table is shown in Table 2. This table includes the actions that need to be taken (the *shall*s), as well as the conditions. The asterisks in the table cross-reference the conditions with the actions. Hence the table is essentially an SCR event table. It differs from the standard SCR event tables in that it shows a number of controlled variables on a single table, and omits the 'modes' column, mainly because we had not distinguished any modes at this stage.

We are exploring the use of PVS to perform coverage and disjointness analysis on this table (following [10]). As part of this process, we have simplified Table 2, by defining each of the fault recovery options as a different mode. This allows us to abstract away from the actual outputs of the subsystem, and model the gross behaviour of the FDIR controller. The result is shown in Table 3. This mode table shows all the transitions from a 'normal' mode to each of the fault recovery modes. Note that none of these modes are described explicitly in the original requirements.

Again, the process was far from straightforward. We found a number of ambiguities and inconsistencies in the expression of the requirements. This made building the table hard, but was a useful exercise, as it allowed us to clarify further our understanding of the requirements. It also helped to identify further weaknesses in the original specification, especially where the operation of the system was not described in adequate detail to explain the required fault analysis functions.

4.3 Full modelling & analysis

The previous steps merely picked out small pieces of the specification and represented them formally, using SCR-style tables. Our final step is to generate a complete formal model of the subsystem. For example, Table 3 only describes the transitions from the normal state to the initial fault diagnosis. The model does not yet include changes in the controlled variables, nor the transition back to normal mode, which includes inhibiting the FDIR for a specified time, to allow the corrective action to take effect. Clearly there are important behaviours associated with the repeated occurrence of errors, which are not represented in the table.

We aim to build a complete formal model of this section of the requirements, based on the four variable model, using a state-based approach. This should allow us to perform a wide range of consistency checks [7]. The original impetus for this work came from the difficulty the IV&V team had in obtaining the developer's design models. The IV&V team needs a model in order to validate that the specified behaviour is correct. If they cannot obtain the original models, they must build their own. Even when they do have access to the developer's models, it is often desirable to develop new models using a different paradigm from that used by the developer. Hence the choice of a formal model.

From a research point of view, our aim in developing an SCR-style model is twofold. First, we wish to evaluate the utility of this kind of analysis in the context of IV&V. In particular, we need to determine how much extra effort is required to produce a formal model, and what the payoff is in terms of the kinds of analysis it allows us to perform. As always, IV&V works within a limited budget, and it may turn out that the effort required to build the formal models is too high to permit the technique to be used widely. Second, we wish to determine what steps are involved in building formal models within an IV&V context, and how those steps might best be supported.

We had hoped to continue with the incremental approach outlined in the previous two sections, until we had a complete model. Unfortunately, the original specification and the SCR model are fundamentally different kinds of abstraction. There is no simple mapping between the original informal specification and the formal model we are developing. The original speci-

Table 2: Merged table for section 2.16.3

Text	Type	(a)	(b)	(b)	(c)	(c)	(d)	(d)	(d)	(d)	(e)	(e)	(e)	(e)	(f)	(f)	(f)	(f)
While acting as the bus controller,	cond.	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
the C&C MDM CSCI	actor																	
shall switch the channel to the backup bus	a-action1	*																
shall reset the current SPD channel	b-action1		*	*														
shall pause the Bus FDIR logic on the SPD 1553 channel for TBD (6-12 seconds) (until the channel is back on-line)	b-action2		*	*														
shall reset the SPD card	c-action1				*	*												
shall pause the Bus FDIR logic on both SPD 1553 channels for TBD (15-20 seconds) (until the channels are back on-line)	c-action2				*	*												
shall set the e,c,w, indicator identified in Table 3.2.16-II for the corresponding RT to failed,	d,f-action1						*	*	*	*					*	*	*	*
shall set failure status to failed	d-action2						*	*	*	*								
shall set the e,c,w, indicator identified in Table 3.2.16-II for itself to failed,	e-action1										*	*	*	*				
shall set failure status to failed for itself	e-action2										*	*	*	*				
shall set failure status to failed for all RT's on the bus	f-action2														*	*	*	*
the backup bus is available	cond.	T																
upon detection of transaction errors in two consecutive processing frames	trigger	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
errors are on selected messages to RTs...	cond.	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
...whose 1553 FDIR is inhibited	cond.	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
within 100 millisec of detection of the second transaction error	timing	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
the bus has been switched in the (current) major (10-second) frame.	cond.	F	F	F														
the SPD Channel's reset capability is inhibited,	cond.		F	F	-	T	-	-	T	T								
the switch to the alternate bus is inhibited	cond.		T	-			-	T	-	T								
the bus has been switched in the last major frame.	cond.		-	T			T	F	T	F								
the SPD card reset capability is inhibited,	cond.				F	F					T	T	-	-	T	T	-	-
the SPD card has been reset in the last 10 major (10-second) frames,	cond.				F	F					-	-	T	T	-	-	T	T
the transaction errors are from multiple RT's	cond.				T	T	F	F	F	F	T	T	T	T	T	T	T	T
the current channel has been reset within the last major frame	cond.				T	F	T	T	F	F	T	F	T	F	T	F	T	F
a backup BC is available,	cond.										T	T	T	T	T	T	T	T
the BC has been switched in the last 20 sec	cond.										F	F	F	F	T	T	T	T
the bus channel's reset capability is inhibited	cond.										-	T	-	T	-	T	-	T

Table 3: Mode transition table for Normal

Current Mode	Conditions										Next Mode
	two consecutive errors	bus switched last frame	channel reset last frame	bus switch inhibited	bus switched this frame	card reset inhibited	card reset last frame	multiple RTs involved	channel reset last frame	backup bus available	
Normal	@T	-	-	-	F	-	-	-	-	T	switch buses
	@T	-	F	T	F	-	-	-	-	-	reset the channel
	@T	T	F	-	F	-	-	-	-	-	reset the channel
	@T	-	-	-	-	F	F	T	T	-	reset the channel
	@T	-	T	-	-	F	F	T	F	-	reset the channel
	@T	T	-	-	-	-	-	F	T	-	RT has failed
	@T	F	-	T	-	-	-	F	T	-	RT has failed
	@T	T	T	-	-	-	-	F	F	-	RT has failed
	@T	F	T	T	-	-	-	F	F	-	RT has failed
	@T	F	-	-	F	T	-	T	T	T	FDIR has failed
	@T	F	T	-	F	T	-	T	F	T	FDIR has failed
	@T	F	-	-	F	-	T	T	T	T	FDIR has failed
	@T	F	T	-	F	-	T	T	F	T	FDIR has failed
	@T	-	-	-	T	T	-	T	T	T	All RTs failed
	@T	T	-	-	-	T	-	T	T	T	All RTs failed
	@T	-	T	-	T	T	-	T	F	T	All RTs failed
	@T	T	T	-	-	T	-	T	F	T	All RTs failed
	@T	-	-	-	T	-	T	T	T	T	All RTs failed
	@T	T	-	-	-	-	T	T	T	T	All RTs failed
	@T	-	T	-	T	-	T	T	F	T	All RTs failed
	@T	T	T	-	-	-	T	T	F	T	All RTs failed

fication is essentially a structural/procedural view of the requirements. This is consistent with the multigraph paradigm [8], used by the developers to identify possible system failures, and to prescribe results. It is also consistent with the design models, developed using the tool MATRIXx, that are used in the developer's internal V&V processes. In contrast, the formal SCR model is state-based, representing a behavioural view.

At this point we have reached a threshold. It appears that we cannot continue complete the formal model just by adding more tables. We need to take a leap of faith and abandon the existing structure of the specification. In its place, we will need to define modes and mode transitions that have no direct counterpart in the informal specification. This will allow us to generate a complete formal model, to which we can apply model checking techniques. The dilemma is that the more we develop our formal model, the more we lose track of the relationship with the original specification. As soon as this happens, our analysis loses its value, as we can never be sure that we are analysing the original specification.

5 Discussion

We have described our on-going work with SCR as a tool for an Independent V&V team to perform analysis of software requirements. Our initial results are very encouraging: the tabular representations remove the ambiguities of the English language requirements, and are easy to read by different audiences. However, our central problem is now one of maintaining fidelity between informal and formal specifications.

As we have described, the existing informal specifications reflect a very different abstraction to that used in a state-based formal model. Within a development process, this would not be a problem. If the development team can commit themselves fully to the formal specification, the informal specification becomes redundant, and can be put to one side. The formal specification can then be used both for the validation process (checking that the requirements capture the real need) and for subsequent verification processes (checking that design, implementation, etc., meet the requirements). Therefore, it does not matter if the formal and informal specifications do not agree – the formal specification can be validated in its own right, and will take precedence.

However, in an IV&V context, a formal model developed by the IV&V team cannot ever replace the informal specification. It is not the IV&V team's responsibility to write specifications for use in development. To do so would destroy their independence. The IV&V team can analyse, identify problems, and make recommendations, but they cannot do the developer's work for them. Hence, any models created by the IV&V team are purely for their own use during analysis.

The IV&V team must therefore either persuade the developers to adopt formal methods themselves, or take care to maintain fidelity between the developer's informal specifications and their own formal models. With the current state of practice, wholescale adoption of formal methods by the developers on an existing project is unlikely. The barriers are well documented, and include a lack of industrial strength tool support, the need to integrate formal methods with existing processes, and of course a training overhead [5]. However, the developers are beginning to adopt the tabular forms we presented in the previous section. An incremental introduction of formal notations seems likely. In the meantime, the IV&V team must still face the problem of fidelity between informal and formal specifications.

The fidelity problem is important to IV&V for three reasons. First, formal models developed by IV&V are produced for the purposes of checking the developer's specifications. The models are only useful for this purpose if they are accurate representations of the developer's specifications.

Second, the informal specifications will continue to evolve throughout the process. Building formal models of these specifications is not a one time activity, but a process of incremental

change. When the developers deliver a new version of a specification, the IV&V team is faced with the daunting task of determining how the changes affect the work they have already done. If they have developed formal models, the models must be updated to reflect the changed specification. Hence, the informal and formal specifications need to have full traceability between them.

Third, when analysis of the formal models reveals problems in the specifications, these problems must be traced back to the informal specification before they can be reported. Although it may be possible to present the problem to the customer and developer in terms of the formal analysis performed, correction of the problem must take place in the developer's informal specifications. Hence, reverse traceability is also important.

Although the fidelity problem seriously affects the utility of any formal analysis performed by the IV&V team, we should point out that it does not affect all the benefits of formal specification. The process of translating pieces of the informal specification into a formal notation has benefit not just for the analysis that it leads to, but also for the removal of ambiguities and improved understanding. For this benefit, it is the *process* of formalisation, rather than the end product that is important.

6 Conclusions

This paper has described our initial work in the use of formal methods in an IV&V project. We have discussed how the demands placed on methods and tools in IV&V are different from their use in a development context. We have also discussed how IV&V can act as a process improvement agent, and hence can be a fruitful way of introducing formal methods into large projects.

As with all potential uses of a new method, any extra effort needed to use the method must be more than offset by the benefits it brings. Use of a method in IV&V is no different. We can divide the benefits of using a formal method such as SCR into two areas:

1. The process of translating portions of a specification into a tabular notation helps to detect ambiguities and increase readability. The process can also be used to catch misunderstandings, thus increasing the confidence that the IV&V team are interpreting the specification correctly. The process of having several analysts produce their own tabular translations was particularly useful in this respect. Differences in the tables they produced allowed us to pinpoint exactly what the disagreement was about.
2. The resulting tables can be analysed for attributes such as coverage and disjointness. This is a substantial contribution to the IV&V team's efforts to check the technical integrity of the specifications. Such attributes are particularly hard to analyse from the informal specifications. As we develop more complete models, we plan to apply a fuller range of consistency and completeness analysis.

The problems we encountered in applying SCR were as follows:

1. The process of translating into a formal notation is error-prone. Only by duplicating the translation effort were we able to discover just how much scope there is for misinterpretation. Luckily, the resulting tables are very readable. Therefore it is much easier to compare different tables than it is to compare different versions of the informal specification.
2. For IV&V, fidelity and traceability between the informal and formal specifications is difficult to guarantee. The value of any analysis carried out by IV&V on the formal model is entirely dependent on how faithful the formal model is to the developer's informal specification. The IV&V's formal model can not be used in place of the informal specifications produced by the developers.

3. The fidelity problem is complicated by the fact that the formal models use a significantly different abstraction to that used in the informal specification. The process of incrementally building a formal model by producing tabular representations of small pieces of the original specification broke down once we began to put the tables together.

The problems of fidelity and traceability between specifications written in different notations is important enough to warrant more attention. We plan to study the problem in more detail by developing a set of tools based on the ViewPoint framework [4], which will allow us to model relationships between partial specifications written by different people. We are also exploring how this problem relates to that of linking test cases to requirements [3]. Finally, we are continuing the experiments described in this paper by examining how model checking can be used to validate the specifications.

Acknowledgments

Our thanks are due to Chuck Neppach and Dan McCaugherty for many interesting discussions of the work presented here, and to Frank Schneider, Edward Addy, John Hinkle, George Sabolish, Todd Montgomery and Butch Neal for detailed comments on earlier drafts of this paper.

References

[1] V. Basili. The experience factory and its relationship to other improvement paradigms. In *Proceedings of the 4^{th} European Software Engineering Conference, Garmish-Partenkirchen, Germany*, September 1993.

[2] J. Callahan. Bipartite process models in software development organizations. Technical Report NASA-IVV-96-005, NASA/WVU Software IV&V Facility, March 1996.

[3] J. Callahan and T. Montgomery. An approach to verification and validation of a reliable multicast protocol. In *Proceedings of the ACM International Symposium on Software Testing and Analysis (ISSTA)*, January 1996. This paper is available at http://research.ivv.nasa.gov/~callahan/Papers/issta96/issta.html.

[4] S. M. Easterbrook and B. A. Nuseibeh. Using viewpoints for inconsistency management. *BCS/IEE Software Engineering Journal*, 11(1), January 1996.

[5] D. H. Craigen S. L. Gerhart and T. J. Ralston. An international survey of industrial applications of formal methods, vol 1: Purpose, approach, analysis and conclusions. Technical Report NRL/FR/5546–93-9581, Naval Research Laboratory, 1993.

[6] M. Heimdahl and N. Leveson. Completeness and consistency analysis of state-based requirements. In *Proceedings of the 17^{th} International Conference on Software Engineering*, pages 3–14, April 1995.

[7] C. Heitemeyer B. Labaw and D. Kiskis. Consistency checking of scr-style requirements specifications. In *Second IEEE International Symposium on Requirements Engineering*, pages 56–63, March 1995.

[8] J. Sztipanovits G. Karsai C. Biegl T. Bapty A. Ledeczi and A. Misra. Multigraph: An architecture for model-integrated computing. In *ICECCS'95*, November 1995.

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

[10] S. Owre J. Rushby and N. Shankar. Analysing tabular and state-transition specifications in pvs. Technical Report CSL-95-12, Computer Science Laboratory, SRI International, 1995.