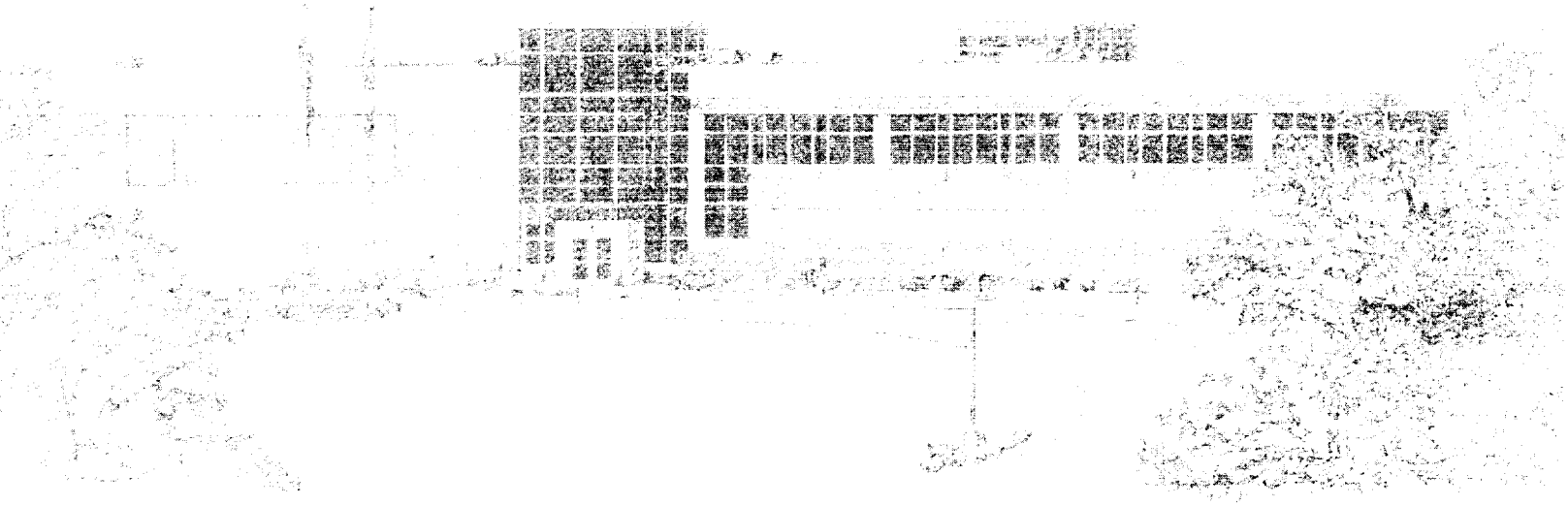
NASA-IVV-97-010
WVU-IVV-97-010
WVU-CS-TR-97-013

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

# Formal Methods for Verification and Validation of partial specifications: A Case Study

By Steve Easterbrook and John Callahan



National Aeronautics and Space Administration

West Virginia University

# Formal Methods for Verification and Validation of partial specifications: A Case Study

Steve Easterbrook and John Callahan
NASA/WVU Software Research Lab,
NASA Independent Verification and Validation Facility
100 University Drive
Fairmont, WV 26554
USA
easterbrook@ivv.nasa.gov
callahan@cs.wvu.edu

## Abstract

This paper describes our work exploring the suitability of formal specification methods for independent verification and validation (IV&V) of software specifications for large, safety critical systems. An IV&V contractor often has to perform rapid analysis on incomplete specifications, with no control over how those specifications are represented. Lightweight formal methods show significant promise in this context, as they offer a way of uncovering major errors, without the burden of full proofs of correctness. We describe a case study of the use of partial formal models for V&V of the requirements for Fault Detection Isolation and Recovery on the space station. We conclude that the insights gained from formalizing a specification are valuable, and it is the process of formalization, rather than the end product that is important. It was only necessary to build enough of the formal model to test the properties in which we were interested. Maintenance of fidelity between multiple representations of the same requirements (as they evolve) is still a problem, and deserves further study.

**Key Words:** validation, formal methods, lightweight, requirements, specification

## 1 Introduction

Requirements engineering methods typically provide a set of notations for expressing software specifications, together with tools for checking properties of specifications, such as completeness and consistency. In general, such methods demand a full commitment. It is assumed that the method will be used to construct a complete specification, which will then act as a baseline for subsequent development phases. However, to validate and verify large specifications for safety-critical real-time systems, it is sensible to apply a number of different methods, to overcome weaknesses and biases of each individual method. For example, a formal method might be used to model a critical portion of an informal specification, to check safety and liveness properties of that portion. In order to manage the application of multiple methods, it is necessary to develop and maintain alternative representations of partial specifications, and to express the relationships between them.

This paper describes a case study of the use of formal 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.

There are a number of questions that need to be addressed before formal methods can be used in this way. Most published case studies of formal methods have focussed on the use of a formal specification as a baseline from which design and code can be verified [1]. In contrast, we have been applying formal methods for intermittent "spot checks" to test for errors as the requirements evolve. The term "lightweight formal methods" has been used to describe this approach [2]. In this context, the formal specification may be dispensable -- what is important is the insight gained from *the process of* formalizing partial views of the requirements and from validating properties of the resulting models. However, it is still necessary to demonstrate fidelity between the original (informal) specification, and the formal model. Furthermore, iterative application of this approach can be greatly facilitated if the relationships between the partial views are captured [3].

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

NASA-IVV-97-010

NASA IV&V Facility, Fairmont, West Virginia

# Formal Methods for Verification and Validation of partial specifications: A Case Study

**Steve Easterbrook and John Callahan**

**July 28, 1997**

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

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

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

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). 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 line of flight code, the potential problems are considerable. Software IV&V is being performed by Intermetrics. 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 section 2, we outline the IV&V process, and discuss the aspects of this process that hinder effective IV&V. The remainder of the paper focuses on the use of methods and tools within this process. The study made use of a combination of AND/OR tables [4], the Software Cost Reduction (SCR) approach [5], and the SPIN model checker [6]. Application of formal methods in this context was not always easy. The informal specification from which we derived our models did not permit an easy translation into a state-based model. We encountered problems in demonstrating fidelity, and providing traceability between the two. Section 4 discusses the issues involved in maintaining multiple representations of requirements, and the use of consistency checking to increase fidelity.

We conclude that in an IV&V context, the analytical benefits offered by formal methods have to be weighed against the effort needed to maintain fidelity between a formal model and the informal specification used by the development team. An IV&V team needs to be able to perform partial analyses on partial specifications, without being tied to any one formalism. The analysis carried out must be sufficient to reveal important problems, as opposed to surface defects. Further analysis is a waste of effort until these problems have been fixed. This conclusion implies a change of perspective for the use of formal methods: while the specification is still evolving it is important to identify quickly any major defects; it is not necessary to perform a complete analysis. Tools that are geared towards finding and characterizing such problems (E.g. SCR* [5], Nitpick [7], etc.) are more useful than tools geared towards proving correctness (E.g. theorem provers).

# 2 Context: The IV&V Process

For *Independent* Verification and Validation (IV&V), the software 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 in no way replaces in-house verification and validation. 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. Most importantly, the IV&V contractor should be engaged as early as possible in the project: studies have shown that IV&V has the biggest impact in the early phases, especially in the requirements phase [8].

As an example IV&V activity, consider the analysis of specifications on the Space Station project. The relevant development contractor writes a Software Requirements Specification (SRS) for each Software Configuration Item (CSCI). These specifications 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. The IV&V contractor analyzes these for technical integrity, in order to identify any requirements problems and 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 that either the requirements be rewritten, or 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 maximize effectiveness. A criticality and risk analysis may 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.

**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. It also has discretion over 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.

**Lack of voice** - The IV&V contractor may have difficulty in getting its message across, especially when 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 more willing to listen to assurances from the developer. The effectiveness of IV&V then depends on having a high-placed advocate within the customer organization.

Despite these problems, IV&V has been shown to be a cost-effective means of improving the quality of the software product, and providing extra assurance for high-cost, safety-critical projects [9, 10]. In addition to providing analysis of project artifacts (e.g. requirements, code, test plans), the presence of IV&V in the lifecycle also has a positive effect on the quality of the software. Our work suggests that the *interaction* between the IV&V and development teams drives improvements in both products and processes [11, 12]. This effect, however, is difficult to capture and quantify.

In this paper we report our work in the evaluation and adaptation of tools and methods for IV&V, and in particular the potential of formal methods to address the problems we have described. The choice of the right methods and tools is important to the success of IV&V. 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 developers know that their product will be analyzed 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 [13]. Hence, in some ways the IV&V team can play a role as a proving ground for new techniques, and can come to be an agent of process improvement. For these reasons, we believe that IV&V offers a practical route through which formal methods may be introduced into projects that would otherwise not be able to adopt them.

There are still problems to be overcome whenever the IV&V team adopts a tool that is not used by the developers. If the IV&V team uses 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.

Lightweight formal methods seems to offer an ideal tools for IV&V. We use the term 'lightweight' to indicate that the methods can be used to perform partial analysis on partial specifications, without a commitment to developing and baselining complete, consistent formal specifications. The formal methods are used to model critical chunks of an informal specification, to check that key properties hold. Application of the methods is driven by the needs of the project, and is used as a modeling tool to answer questions that arise during verification and validation. In the remainder of the paper we describe a case study that investigated the applicability of lightweight formal methods to an IV&V effort.

(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 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.

**Figure 1: An example of a level 3 requirement for Bus FDIR. This requirement specifies the circumstances under which all remote terminals (RTs) on the bus should be switched to their backups.**

# 3 The Case Study

The purpose of this case study was to analyze the detailed Fault Detection, Isolation and Recovery (FDIR) requirements associated with the bus controller for the main 1553 communications bus on the space station. The study was conducted at the request of the Independent Verification and Validation (IV&V) team. The IV&V team was having particular difficulty validating the bus FDIR requirements, as they were hard to read, and some of the properties they wished to test could not be established using existing informal methods. The study was conducted by the authors, as part of a larger study of the use of formal methods in the V&V process [14].

The requirements for Bus FDIR had been expressed in natural language, with a supporting flowchart showing the processing steps involved. The flowchart did not have the status of a requirement, but was merely provided for guidance; the intention was that the prose completely expressed the requirements. The prose contained a number of long complicated sentences, expressing complex conjunctions and disjunctions of conditions (E.g. see figure 1). The IV&V team had recommended that to improve clarity, the requirements should be re-written in a tabular form (specifically, as truth tables similar to those used by Heimdahl & Leveson [4]). This recommendation had been rejected because of the cost involved in re-writing them all. The IV&V team was the faced with the problem of how to validate the informal statement of the requirements, given that this informal statement would act as the baseline. Hence, the IV&V team generated their own tabular versions, in order to facilitate the kinds of analysis they wished to perform.

## *3.1 Approach*

Our approach to the application of formal methods in an IV&V context follows a simple four step process as follows:

1) restate the requirements in a clear, precise and unambiguous format;
2) identify & correct internal inconsistencies;
3) test the requirements by proving statements about expected behavior;
4) feed the results back to the requirements authors.

In some cases step 1 involves the use of an intermediate representation, as abstractions introduced in the formalization do not necessarily have any natural correspondences in the original requirements statements. In such cases, intermediate representations help to ease the translation process, and help to provide a traceability path between the informal and formal statements.

For this case study, the four step approach was applied as follows. Each individual requirement was restated as a truth table, to clarify the logic. These were then combined into a single state-machine model, using SCR [5]. The SCR model was checked for consistency using the SCR toolset. Properties of the model were then tested in two ways. Firstly, static properties of the state model, such as disjointness and coverage, were tested using the built-in checker in the SCR tool. Secondly, dynamic properties of the model were tested by translating the SCR state machine model into PROMELA [6], and applying the SPIN model checker to explore its behavior. This allowed us to explore timing properties that

| | | | | 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: The truth table version of the requirement shown in figure 1, showing the four conditions (the four columns) under which the action should be carried out. A dot indicates "don't care".**

could not be validated in SCR. The results were fed back to the development team through the normal IV&V reporting process.

The choices of method and notation to use in the study were entirely pragmatic. In each case, the choice was made only after the previous step was underway, and the decision was driven by a need to test certain properties. The initial truth table representation was chosen as it was precise, easy to read, and represented approximately the same abstraction as the original informal requirements (See table 1). These tables are modeled on the AND/OR tables adopted by Leveson during the development of the RSML specifications for TCAS II [4].

SCR was chosen as it offered a tabular notation that corresponded well to the truth tables that the IV&V team had already adopted, and it provided tool support for checking consistency of state models. The consistency checking in SCR is sufficiently sophisticated that it allowed us to express some of the validation properties as consistency checks. Specifically, we wanted to establish that each failure mode in the model was disjoint (i.e. that there was no condition in which two different failure modes would be diagnosed simultaneously), and that together the failure modes covered all possible combinations of failure conditions. By expressing the failure modes as a mode transition table in SCR, we could test these properties automatically. Most importantly, the tool did not require us to define the complete state model in order to test these properties. A further advantage of SCR was that the consistency checker in the SCR tool provides a counter-example whenever an inconsistency is found. The provision of counter-examples is important in tracing problems back to the informal specification, and in convincing the development team that there really is a problem.

The SPIN model checker was adopted for the final part of the study[1], as it allowed us to explore dynamic aspects of the state model, including the ability to simulate different types of error condition. Properties of interest were expressed as "never clauses", so that the model checker would report any reachable state where the property was satisfied. Interesting error conditions, such as intermittent and repeating faults, were simulated by including a test harness in the PROMELA model, to set and reset the inputs to the fault detection model. Strictly speaking this test harness was unnecessary, as it merely produced a specialization of the original model. However, we found it useful as a way of reducing the search space, and observing certain key behaviors directly, rather than searching for them in the model.

## *3.2 Procedure*

The generation of a tabular interpretation of each individual requirement proved to be hard, as there are a number of ambiguities in the prose requirements. These ambiguities concern the associativity of 'and' and 'or' in English, and the correct binding of subclauses of long sentences. For example, in figure 1, it is not clear what the phrase "in two consecutive processing frames" refers to. To confirm the existence of such ambiguities, the requirement shown in

[1] Note: since this case study was conducted, the spin model checker has been incorporated into the SCR toolset [15].

| Current Mode | Conditions | | | | | | | | | | | Next Mode |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | errors in two cons. frames | bus swch'd last frame | bus switch inhibit | bus swch'd this frame | backup BC avail. | BC swch'd in last 20 sec | card reset inhibit | card reset last 10 frames | errors from mult. RTs | channel reset last frame | channel reset inhibit | |
| Normal | @T | - | - | F | - | - | - | - | - | - | - | switch buses |
| | @T | - | T | F | - | - | - | - | - | - | F | reset the channel |
| | @T | T | - | F | - | - | - | - | - | - | F | |
| | @T | - | - | - | - | - | F | F | T | T | - | reset the card |
| | @T | - | - | - | - | - | F | F | T | F | T | |
| | @T | T | - | - | - | - | - | - | F | T | - | switch RT to backup |
| | @T | F | T | - | - | - | - | - | F | T | - | |
| | @T | T | - | - | - | - | - | - | F | F | T | |
| | @T | F | T | - | - | - | - | - | F | F | T | |
| | @T | - | - | - | T | F | T | - | T | T | - | switch BC to backup |
| | @T | - | - | - | T | F | T | - | T | F | T | |
| | @T | - | - | - | T | F | - | T | T | T | - | |
| | @T | - | - | - | T | F | - | T | T | F | T | |
| | @T | - | - | - | T | T | T | - | T | T | - | switch all RTs |
| | @T | - | - | - | T | T | T | - | T | F | T | |
| | @T | - | - | - | T | T | - | T | T | T | - | |
| | @T | - | - | - | T | T | - | T | T | F | T | |

**Table 2: An SCR Mode transition table. Each of the central columns represents a condition, showing whether it should be true or false; '-' means "don't care"; '@T' indicates a trigger condition for the mode transition. The four columns of table 1 correspond to the last four rows of this table. The semantics of SCR require this table to represent a function, so that the disjunction of all the rows covers all possible conditions (coverage), and the conjunction of any two rows is false (disjointness).**

Figure 1 was given to four different people, for translation into tabular form. Four semantically different tables resulted. By comparing these different interpretations, an extensive list of ambiguities was compiled. The ambiguities were resolved through detailed reading of the documentation, and questioning the original authors. This process also revealed some inconsistencies in the way in which terminology was used.

Merging the individual AND/OR tables to produce the SCR model (Table 2) was not straightforward. Although there were a number of conditions common to several of the tables, 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 equivalent. 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.

Having obtained a clearer statement of the requirements, the next step was to explore some of the properties that ought to be true of these requirements. Example properties are "for each combination of failure conditions, there is an FDIR response specified" and "for each combination of failure conditions there is at most one FDIR response specified". These properties correspond to checks for coverage and completeness of a mode table in SCR. By constructing a state-based model in which each of the requirements represented a transition from the "normal" mode to a unique failure mode, the coverage and disjointness tests in the SCR tool would test these properties Note that the failure modes are not identified explicitly in the original specification. In Table 2, we have named them after the corrective action rather than the type of failure, to preserve traceability to the informal specification. A number of disjointness problems were identified at this stage, which are described below.

The final part of this study was to explore the dynamic properties of the model. For example, some of the requirements express conditions that test whether various recovery actions have already been tried. In order to validate these conditions, it was necessary to explore the dynamic behavior of the specified system in the face of multiple failures, and recurring failures. To do this we needed a way to exhaustively search the state space for safety and liveness properties. We needed to add non-determinism to the model, to simulate possible occurrences of faults, as inputs to the system being modeled. The state-based model expressed in SCR was translated into PROMELA, and the model checker SPIN was used to explore the behaviors.

```
proctype FDIR_mode_pc ()
{
mtype mode = normal;

end:
  do :: new_frame?true ->
    if
      :: (errors == false || FDIR_inhibit_RT == true) -> mode = normal
      :: else -> if
         ::  /* [branches here for each failure mode] */
         ::
         :: (((backup_BC_avail == true) && (BC_switched_20_sec == false)
    && ((cd_reset_inhibit == true) ||  (cd_reset == true))
    && (((errors_mult_RTs == true) &&  (ch_reset_last == true))
    || ((errors_mult_RTs == true) &&  (ch_reset_inhibit == true)
    && (ch_reset_last == false)))) && mode == error1) ->
    atomic{ mode = fm_BC_failure;
            BC_switch_count = BC_SWITCH_TIMEOUT };
                 BC_switcher!true

         :: else -> assert(false)     /* This is a coverage error */
       fi
     fi; new_frame!false
  od
}
```

**Figure 2: Excerpt from the promela model. The inner if loop contains branches for each of the failure modes shown in table 2. Only the code for the failure mode "Switch BC to backup" is shown here. This branch corresponds to four rows in Table 2.**

The PROMELA model consisted of a number of concurrent processes. The main process modeled the failure diagnosis model, and was translated more or less directly from the SCR mode table. Figure 2 shows a portion of this process, and can be compared with the mode table shown in table 2. Additional processes were added for: the state space of inputs to the system, a timer to model timeout behavior, and a simulation of the state of each of the devices that the FDIR system interacts with.

Experimentation with this model in SPIN indicated some inconsistencies in the timing constraints that had not been revealed in the SCR model. Once these were fixed, the model checker was used to verify properties such as "if an error persists after all recovery actions have been tried, the bus FDIR will eventually report failure of itself to a higher level FDIR domain". Figure 3 shows how a claim such as this is translated into a Linear Temporal Logic (LTL) formula, to be used by the model checker.

## 3.3 Findings

In addition to a number of minor problems with inconsistent use of terminology, the following major problems were reported:

1) There were significant ambiguities in the prose requirements, as a result of the complex sentence structure. Some of these ambiguities could be resolved by studying the higher level FDIR requirements, and the specifications for the bus architecture. The ambiguities that arose from the combination of 'ands' and 'ors' in the same sentence could not be resolved in this way, and could lead to mistakes in the design. These ambiguities were detected in the initial reformulation of the requirements as truth tables.
2) There was one missing requirement to test the value of the Bus Switch Inhibit Flag before attempting to switch to the backup bus. This was detected during the test for disjointness in the SCR specification.

```
#define p  (errors==true)
#define q  (mode==fm_BC_failure)
[](p ->  <>(!p \/  q))
```

**Figure 3: A sample proof claim for the SPIN model. The symbols [] and <> are the usual temporal operators 'always' and 'eventually'. The claim is "whenever an error occurs, eventually a state will be reached in which the error is cleared or the bus controller will report failure of itself to a higher level FDIR domain". This claim failed, leading to finding 4 described in section 3.3**

3) The prose requirements were missing a number of preconditions that enforce the ordering in which the inference rules should be applied. The accompanying flowchart for these requirements implied a sequence for these rules. An attempt had been made in the prose requirements to express this sequence as a set of preconditions for each rule, to ensure that all the earlier rules have been tested and have failed. The preconditions did not completely capture the precedences implied by flowchart. This corresponded with an informal observation made by the IV&V team that the ordering of the requirements should be made explicit. This problem was found during the test for disjointness in the SCR specification.

4) The timing constraints expressed in the requirements were incorrect. Several of the failure isolation tests referred to testing whether certain FDIR actions had already been tried "in the previous processing frame". However, as each FDIR recovery action is followed by a time-out in order for the action to take effect, and as further FDIR intervention is only initiated on occurrence of errors in two consecutive processing frames, these tests can never be true. This was discovered during model checking of the PROMELA model.

### *3.4 Observations*

The study analyzed 15 pages of level 3 requirements, and was conducted over a period of four months, by one person working part time. The total effort was approximately 1.5 person months. The main effort was in formalizing the requirements. Translation from the SCR model to PROMELA was relatively straightforward, and took two days effort. Once a formal model was obtained, testing of the properties was straightforward, as both the SCR tool and the SPIN model checker provided facilities for automated checking of these properties, and provided counter-examples when the tests failed. Although problems were found both during formalization and the property checking, the latter problems were more serious. It is unlikely that they would have been discovered in this phase without the use of formal methods.

A major difficulty during this study was the volatility of the requirements. New drafts of the requirements document were being released approximately every two months. This meant that in at least one case (finding 3 above), the problem had already been fixed by the time it was discovered in this study. This particular finding had already been observed informally and reported by the IV&V team, and had been addressed by reducing the complexity of this section of the requirements. We mitigated the problem of fluctuating requirements by only doing the minimum amount of modeling necessary to test the properties that were of interest. For example, the SCR model is not a complete state model, as it models only a subset of the state transitions expressed in the requirements. The transitions for returning to the normal state have not been modeled. This partial model was sufficient to perform the coverage and disjointness analysis.

It should also be noted that in order to perform the analysis in this study, the SCR notation was slightly misused. The modes shown in figure 4 do not represent true modes in the SCR sense – a more correct representation would express these as output events from the FDIR system. However, defining them as modes permitted the use of coverage and disjointness tests on the transitions. This reflects our pragmatic approach in which the formal method is applied in whatever way gives the most benefit, without necessarily following the original intent of the method.

## 4 Discussion

This paper presented a case study on the use of formal methods for IV&V. The results are very encouraging: the translation process was extremely valuable in identifying ambiguities and improving our understanding of the specification. In this process, a number of errors were found. Analysis of a partial formal specification demonstrated several important errors in the specification, and appears to be a powerful means of gaining maximal results from minimal effort. We constructed just enough of a model to test the properties we were interested in, without any further commitment to the method.

The study described in this paper differs from other studies in the literature in several ways. The majority of published case studies of the use of formal methods are post hoc application of formal method to on-going or finished projects. Such studies are demonstrate what formal methods can do, and help to refine the methods, but they do not help to answer questions of how such methods can be integrated with existing practices on large projects. A few notable exceptions have used formal methods 'live' during the development of real systems [1, 4, 16, 17]. However, in all these cases, the emphasis was on the use of formal notations as a part of the baseline specifications, from which varying degrees of formal verification of the resulting design and implementation are possible. In contrast, we applied formal methods as a means of performing V&V on an informal baseline specification. Rather than treating formal

specification as an end product of the requirements phase, we used it to answer questions and improve the quality of the existing specifications.

The introduction of formal methods as an IV&V technique for large projects offers a low risk approach to technology transfer. This approach allows us to be flexible and pragmatic. The amount of formal modeling performed can be varied according to the project needs. In particular, the case study indicates that significant results can be obtained from the use of formal methods using *partial* validation of *partial* specifications. For an IV&V team, the formal methods provide an additional tool to be used on highly critical sections of a specification, or to test certain key properties, where existing techniques cannot help.

There are two potential pitfalls with this approach: it is hard to guarantee fidelity between informal and formal specifications, and it is hard to manage consistency between partial specifications expressed in different notations. These problems arise from the need to maintain both informal and formal specifications of the same requirements.

The fidelity issue is more of a problem in IV&V than in development. A formal model developed by the IV&V team cannot replace the informal specification. The IV&V team must therefore either persuade the developers to adopt formal notations themselves, or take care to maintain fidelity between the developers' informal specifications and their own formal models. The formal models are only useful for checking the developer's specifications if they are accurate representations of the developer's specifications. Also, 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 the fidelity problem can affect 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 for improved understanding. For this benefit, it is the *process* of formalization, rather than the end product that is important. In particular, we observed that the IV&V analysts had a much better understanding of the requirements after conducting the translation exercise than they would normally obtain.

The fidelity problem is really a special case of a more general problem: management of consistency between partial specifications expressed in different notations [3]. For instance, the AND/OR tables have a clear relationship with the SCR mode tables, but if we make a correction to one of the AND/OR tables, it is fairly tedious to identify the corresponding correction in the SCR tables. Similarly, each time the developers issue a new informal specification, we need to update our tabular representations. Although it may seem that the use of both AND/OR tables and SCR models together would compound this problem, the opposite is true. The AND/OR tables mapped clearly onto the textual requirements, while the relationship between the AND/OR tables and the SCR model was relatively straightforward. Therefore, the use of AND/OR tables as an intermediate representation reduced the traceability gap, and made it easier to keep the formal model up to date. There remains, however, a significant bookkeeping problem.

There is a growing body of work on managing consistency in specifications. Our previous work demonstrated how to delay the resolution of inconsistency, and provided a generic framework for expressing consistency relationships [3]. Other work has taken consistency checking further, making use of semantic models underlying a method to determine what consistency rules are needed and how to operationalize them. For example, Heitmeyer's work with consistency checking in SCR [5] uses the semantics of SCR to define a series of consistency rules ranging from simple syntactic checks (e.g. that all names are unique) to sophisticated properties of tables (e.g. coverage and disjointness). Similarly, Leveson's work on consistency checking in RSML [4] uses the semantics of the statechart formalism to determine a set of consistency rules that can be tested, tractably, using a high level abstract model. In both these approaches, the completeness of the formal specifications is important, and consistency checking is seen as part of the process of obtaining a complete, consistent specification.

Unfortunately, these approaches do not help with consistency checking between partial specifications expressed in different notations. Because the IV&V process is concurrent with and complementary to the development process, there is an unusually large amount of flexibility in how a formal method can be used. There is no need to make a commitment to any one formal notation, just as there is no need to develop complete specifications. 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. Development of complete formal models is therefore unnecessary and may be counter-productive. For example, in our second experiment, the limited analysis we performed on a partial model was sufficient to reveal a major problem; the existence of this problem meant that any further effort to complete the model would have been wasted.

While the use of partial specifications offers greater flexibility in the use of methods and tools, it also means that we do not have a well-defined method from which to generate a set of consistency relationships. There are implicit consistency relationships between the assorted partial specifications drawn from different methods, but there is no overall 'method' to tell us what these relationships are. Actually, there is a method: the problem is that the method is implicit, and to some extent is generated on the fly. For example, there is a method for generating SCR mode tables from the AND/OR tables, but the method was not defined before we did it. With some effort, we could formalize this method, and define semantic relationships between the two types of table. However, this effort will only be worthwhile if we intend to re-use the method extensively. In the meantime, we would like to have tools to help us keep track of consistency relationships in our opportunistic use of partial specifications.

In our previous work defining consistency relationships between viewpoints, we assumed that the majority of such rules are defined by the method [18]. The viewpoints framework explicitly supports the process of method definition, in which, among other things, the inter-viewpoint relationships are defined. Hence the general problem of defining arbitrary relationships between any two notations is avoided. However, we also recognized that some consistency relationships could not be defined in this way, and gave the example of a user-defined synonym relationship between two different labels. We outlined an approach to discovering such relationships through low level process monitoring. We now regard this type of consistency relationship as vital to any approach involving partial specifications.

# 5 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 formal methods in IV&V into three areas:

- The process of translating portions of a specification into formal notations helps to detect ambiguities and increase readability, even if the translation is only partial. The process can also be used to catch misunderstandings, thus increasing the confidence that the IV&V team is 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 where the ambiguities were.
- The partial formal models can be analyzed for internal consistency, to help establish the technical integrity of the original, informal specification. This step also helps to check that the models are faithful to the original.
- The partial formal models can be tested against key domain properties, to validate the original requirements. In our study, this included both static and dynamic properties of the state-based model. Such properties are particularly hard to analyze from the informal specifications. Most importantly, this analysis can be conducted without the need to build complete models.

From this study, we conclude that lightweight formal methods are an ideal tool for an IV&V agent. They address many of the problems we identified in section 2:

**Limited resources:** Lightweight formal methods can be applied to selected portions of specifications. The amount of modeling and analysis can be adjusted to fit resource constraints.

**Short timescales:** Partial analysis can generate preliminary results quickly, as the analysis can proceed even without a full model.

**Lack of access:** The case study demonstrated that a formal modeling effort can be based almost exclusively on informal documents. Interaction with the development team was only necessary to check assumptions, and to discuss the analysis results.

**Evolving Products:** Small, partial models can be generated quickly, and updated as the specification evolves. The investment in each model is small enough that they can be discarded if the specification changes significantly.

**Avoiding trivial/obvious problems:** Formal analysis can reveal subtle problems that escape the notice of informal, inspection-based methods. In particular, it is a powerful way of detecting timing and safety-related

problems. The formal analysis also allowed us to explore the significance of potential errors before reporting them.

**Lack of voice:** Formal methods can help the IV&V agent to strengthen their case when they report issues back to the customer and developer. Animation of formal models provides a powerful was of demonstrating errors, and helps to provide a more precise characterization of each problem.

The problems we encountered in applying formal methods were as follows:

- 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, our chosen formal notations were very readable. Therefore it is much easier to compare different tables than it is to compare different versions of the informal specification.
- 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.
- Opportunistic use of partial specifications means that there is not a well-defined method from which to derive consistency rules. Maintenance of consistency in our partial specifications became a real problem.

The problem of consistency checking in partial 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 ViewPoints framework, 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 case scenarios to requirements [19].

### *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. This work was supported by NASA Cooperative Research Agreement NCCW-0040.*

## 6 References

[1] D. Craigen, S. L. Gerhart, and T. Ralston, "Formal Methods Reality Check: Industrial Usage," *IEEE Transactions on Software Engineering*, vol. 21, pp. 90-98, 1995.

[2] H. Saiedain, J. P. Bowen, R. W. Butler, D. L. Dill, R. L. Glass, A. Hall, M. G. Hinchey, C. M. Holloway, D. Jackson, C. B. Jones, M. J. Lutz, D. L. Parnas, J. Rushby, J. Wing, and P. Zave, "An Invitation to Formal Methods," *IEEE Computer*, vol. 29, pp. 16-30, 1996.

[3] S. M. Easterbrook and B. Nuseibeh, "Using ViewPoints for Inconsistency Management," *Software Engineering Journal*, vol. 11, pp. 31-43, 1996.

[4] M. Heimdahl and N. Leveson, "Completeness and Consistency Analysis of State-Based Requirements," *IEEE Transactions on Software Engineering*, vol. 22, pp. 363-377, 1996.

[5] C. L. Heitmeyer, B. Labaw, and D. Kiskis, "Consistency Checking of SCR-Style Requirements Specifications," *Second IEEE Symposium on Requirements Engineering*, York, UK, March 27-29, 1995.

[6] G. J. Holtzmann, *Design and Validation of Computer Protocols*: Prentice Hall, 1991.

[7] D. Jackson and C. A. Damon, "Elements of Style: Analysing a software design with a counter-example detector," *International Symposium on Software Testing and Analysis (ISSTA'96)*, San Diego, CA, 8-10 January 1996.

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

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

[10] J. D. Arthur, M. K. Groener, S. Gupta, M. Cannon, and Z. Khan, "Reducing the Mean Time to Remove Faults Through Early Error Detection: An Experiment in Independent Verification and Validation," *18th Minnowbrook Workshop on Software Engineering*, Blue Mountain Lake, NY.

[11] J. Callahan and G. Sabolish, "A Process Improvement Model for Software Verification and Validation," *Journal of the Quality Assurance Institute*, vol. 10, pp. 24-32, 1996.

[12] S. M. Easterbrook, "The Role of Independent V&V in Upstream Software Development Processes," *Second World Conference on Integrated Design and Process Technology (IDPT-96)*, Austin, TX, Dec 1996.

[13] V. Basili, "The Experience Factory and its relationship to other improvement paradigms," *Proceedings of the 4th European Software Engineering Conference (ESEC'93)*, Garmish-Partenkirchen, Germany, September 1993.

[14] R. W. Butler, J. L. Caldwell, V. A. Carreno, C. M. Holloway, P. S. Miner, and B. L. Di Vito, "NASA Langley's Research and Technology Transfer Program in Formal Methods," *Tenth Annual Conference on Computer Assurance (COMPASS 95)*, Gaithersburg, MD, June 1995.

[15] R. Bharadwaj and C. L. Heitmeyer, "Verifying SCR Requirements Specifications using State Exploration," *Proceedings of First ACM SIGPLAN Workshop on Automatic Analysis of Software*, Paris, France, January 14, 1997.

[16] A. Hall, "Using formal methods to develop an ATC Information System," *IEEE Software*, vol. 13, pp. 66-76, 1996.

[17] R. A. Kemmerer, "Integrating Formal Methods into the Development Process," *IEEE Software*, vol. 7, pp. 37-50, 1990.

[18] S. M. Easterbrook and B. A. Nuseibeh, "Managing Inconsistencies in an Evolving Specification," *Second IEEE Symposium on Requirements Engineering*, York, UK.

[19] J. R. Callahan and T. L. Montgomery, "An Approach to Verification and Validation of a Reliable Multicasting Protocol," *International Symposium on Software Testing and Analysis (ISSTA'96)*, San Diego, CA, 8-10 January 1996.

# 7 Biographies

Steve Easterbrook is currently the Senior Research Associate at the NASA/WVU Software Research Lab, in Fairmont, West Virginia. He received a BSc from the University of York in 1986, and a PhD in Software Engineering from Imperial College, London, in 1991. He was a lecturer in Computer Science and Artificial Intelligence at the University of Sussex, UK, from 1990 to 1997. His research interests include requirements engineering and software specification, and he has published extensively on the problems of resolving conflicting requirements and managing evolution in specifications. He is a member of the ACM and IEEE Computer Society, and edits the newsletter of the BCS Requirements Engineering Specialist Group. He has served on a number of program committees, including ASE, RE and IWSSD, and is on the editorial board of Automated Software Engineering.

John R. Callahan is an Assistant Professor of Computer Science in the Department of Statistics and Computer Science at West Virginia University in Morgantown, West Virginia. He received his Ph.D. in Computer Science from the University of Maryland, College Park in 1993 and is currently working on research in verification and validation of computer software. He has worked for Xerox Corporation of Palo Alto, CA, NASA's Goddard Space Flight Center in Greenbelt, MD, and the Department of Defense's Air Force Data Services Center at the Pentagon. Dr. Callahan currently serves as co-Principal Investigator of the NASA/WVU Cooperative Research Project at the NASA Software IV&V Facility in Fairmont, West Virginia.