Bayesian Networks applied to Software IV&V

Ganesh Pai, Joanne Bechta-Dugan and Khalid Lateef†
University of Virginia, Department of ECE, Charlottesville, Virginia, USA
†TITAN Corporation, Greenbelt, Maryland, USA
{gpai,jbd}@virginia.edu, †Khalid.Lateef@titan.com

Abstract

In this paper, we describe how Bayesian networks can be used to merge quantitative and qualitative information to support IV&V of use cases. Essentially, simple metrics from the use cases are computed, which are then input to a Bayesian network. This network models the relationships between the observable parameters of an IV&V process for use cases, and the desired features of the requirements specifications. The output of the network is an assessment of the maturity of the requirements, in terms of the probability that they exhibit the desired properties. We apply our proposed approach to a real system: a software simulator built to test attitude control for an aerospace system, to illustrate how IV&V can be quantitatively supported.

Keywords: *Use cases, software requirements specifications, probabilistic networks, independent verification & validation, software reliability engineering.*

1 Introduction

Requirements errors are most expensive to correct if they are not discovered early during development; such errors are most often responsible for software failures [1, 2]. Therefore, receiving early feedback regarding the validity and/or maturity of evolving requirements during software requirements specification (SRS) development, is valuable. Software independent verification and validation (IV&V) may be used in critical projects to assess and mitigate risk, to increase product quality, and to reduce cost [3]. Typically, IV&V, which takes place in parallel with development, often needs to be relatively rapid to positively influence the development process. The longer the analysis takes, the less useful its results may become, especially for evolving requirements during the SRS development phase.

Owing to the popularity of the unified modelling language (UML) and model-driven development of software, one commonly finds requirements being modelled as UML use cases. The documentation of these use cases and their associated scenarios frequently appears as informal text. Additionally, the UML semantics are semi-formal. Consequently in practice, determining that the requirements are acceptably mature mainly depends on engineering judgement. The primary motivation of the work presented in this paper, is to develop a methodology to (1) harness the existing inspection/subjective analysis techniques and (2) systematically and quantitatively analyse the readiness of requirements, captured as UML use cases.

One approach for the analysis of requirements during IV&V has been the application of lightweight formal methods, where partial analysis is performed on partial specifications, and there is no commitment to converting the entire informal requirements document into a formal one [4, 5]. There is limited guidance in the literature regarding performing V&V on use case based requirements in general, [6] and their applicability has not been addressed in the context of IV&V constraints. To the best of our knowledge, these techniques have not yet been applied in IV&V of UML use cases. In this paper we focus mainly on the use of Bayesian networks for quantitative IV&V analyses.

The rest of this paper is organised as follows: In section 2, we first describe a real example system and provide a high-level overview of the functionality desired from the system. Then, in section 3 we present a process flow for IV&V of use cases. The Bayesian network used to support this process flow is detailed in section 4. Section 5 presents a discussion of the analysis while section 6 presents related work in the literature. We conclude the paper in section 7 and identify avenues for future work.

2 Real example system

To illustrate our approach, we consider a real example system (figure 1): a hybrid hardware/software simulator (PC) built to test the attitude control subsystem (AC) of an aerospace system. The simulator is used to

test the AC to verify that it functions correctly. The PC not only models system dynamics but also simulates the sensors and actuators of the AC. Additionally, the simulator interfaces directly with the AC hardware, providing a real-time, hardware-in-the-loop test environment. The user interacts with the simulator to provide test scripts, to command or configure the simulator, and to display telemetry information. The simulator can be used to model the ground support equipment software as well. Its main purpose is low-level testing of the flight hardware and sensors.

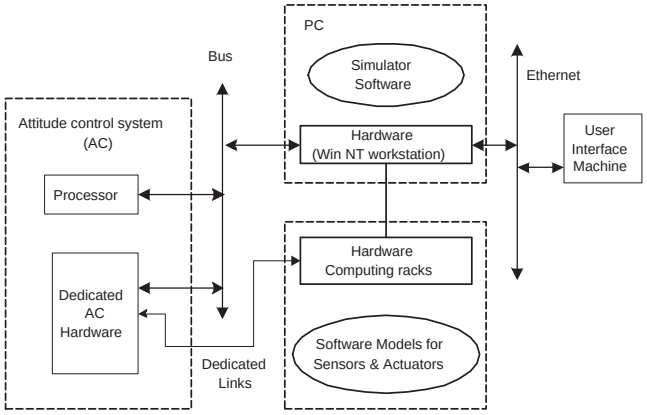

Figure 1: Block diagram of the simulator system

The initial requirements document for this system had been constructed by its developers in natural language, and it included the interface specifications between the simulator and the ground support equipment functions. For this paper, we consider the interface requirements between the PC and the software model for a digital sun sensor. Specifically, the latter provides fine measurement of the sun-direction and this measurement data is communicated to the simulator during a simulation run. Use cases were not available directly; hence, these were constructed from the natural language requirements document using the procedure shown in table 1.

2.1 Requirements representation as use cases

Figure 2 shows three software requirements from the original natural language requirements document. In the figure, the underlined text represents the main use cases, the *underlined italicized text* represent conditions, the **boldface text** represent actors, the ***boldface italicized text*** highlights an association between an actor and a use case, while the **underlined boldface text** highlights a secondary use case related to the main use case by means of an *includes* relationship. Such use cases are captured in our use case descriptions as the operations of a scenario of the main use case.

The original requirements document distinguished software requirements from hardware/system requirements and each software requirement, if there was more than one per module, was numbered. We see in figure 2 that each requirement further had a description of the necessary functionality in a structure which permitted the use of an algorithmic approach to constructing the use cases(table 1).

This procedure constructs the main use cases, the secondary use cases which are related to the main use cases by relationships of type *includes*, the actors, as well as the associations between actors and use cases. It also assigns unique names to both use cases and actors, corresponding to the the respective functionalities described in the original requirements document. In the use case descriptions, secondary use cases are included as operations of a scenario with the corresponding actions and conditions, whereas they are shown explicitly in the use case diagram. Each actor, use case and functionality name are included in the domain dictionary to permit traceability analyses between the use cases and the original requirements. The use cases are, thus, a direct translation of the original requirements and consistency between the two was not otherwise established.

To further maintain consistency, scenario descriptions were only re-structured and minimally altered, without actually translating them into more precise descriptions. For example, the sentence {*the sun presence bit shall be 1...*} was re-written as {*Sun presence bit = 1*} in the post-conditions for the associated operation. However, it is straightforward to precisely represent these operations using the events-conditions-actions table model, as applied by Gervasi *et al.* [7]. Indeed, with the use of a domain based parser, as in their work, the use case construction procedure can be modified to operate on the parse trees of natural language requirements, to automatically build the use cases. One of the avenues for our future work is constructing a domain model of the system to formalise the scenario and operation descriptions using the object constraint language (OCL) of the UML.

Figure 3 shows the use case descriptions that were obtained, while figure 4 shows the corresponding use case diagram. In figure 3, the highlighted text represents some of the problems identified with the original requirements after they were structured into use cases. Specifically four pre- and post-conditions, and one alternate scenario was missing in the use case *PC-SW in SunSensor Interface* while two conditions were unclear in the use case *AC-SW in SunSensor Interface*.

Subsystem: Digital Sun Sensor Interface.

Software Requirement:

The hybrid dynamic simulator software ***shall output to*** the **hybrid dynamic simulator hardware** values for the two sun sensor data words, with the specified format. The command shall have values for *Na* and *Nb*, computed for the time at the cycle tick, offset for the true sun sensor sampling times; this is a time-critical signal. *Na* and *Nb* shall be computed by inverting the equations for the sun angles *a* and *b* as a function of *Na* and *Nb*. The coefficients in the equations for *Na* and *Nb* shall be configurable. *The sun presence bit shall be 1 if the sun is within +/- 32 degrees of the sun sensor optical axis, 0 otherwise*; the sun presence bit shall also be commandable. The **sun sensor orientation** in the spacecraft frame shall be configurable

Ground Support Equipment Software Requirement (1):

In *flight algorithm test mode*, attitude control system ground support equipment software shall model the digital sun sensor values, and ***output them*** to the **attitude control system ground support equipment hardware** **digital sun sensor stimulus interface**, either when the signals are set directly, or when the corresponding angle is set.

Ground Support Equipment Software Requirement (2):

In *flight hardware test mode*, attitude control system ground support equipment software ***shall output to*** the **attitude control system ground support equipment hardware digital sun sensor sun emulator interface** a commandable value.

Figure 2: Sample natural language requirement

Table 1: Procedure to construct use cases from the natural language requirements

Procedure: Construct use case
1. $\forall$ software requirements R_i in the original requirements document
2. **identify** required functionalities $f_i \cap \forall\ f_i\ \exists$ a unique name
3. **identify** system boundary
4. **identify** actors A_i outside the system boundary $\cap\ \forall\ A_i\ \exists$ a unique name
5. $\forall\ f_i\ \exists$ use case u_i with operations $o_i \cap$ $name(u_i) = name(f_i) \cap$ $actions(o_i) = actions(f_i) \cap$ $conditions(o_i) = conditions(f_i)$
6. $\forall\ u_i$ **identify** associations a_{ij} between u_i and actor A_j
7. **identify** relationship r_{ij} between functionalities f_i and f_j
8. **determine** whether $type(r_{ij}) = \{extends\}$ or $\{includes\}$
9. $\forall\ f_j$ **for which** $type(r_{ij}) = \{includes\}\ \exists\ r_{ij}$ from u_i to u_j in the use case diagram $\cap$ $\exists$ operations o_j in u_i in the use case description **such that** $name(o_j) = name(f_j) \cap$ $actions(o_j) = actions(f_j) \cap$ $conditions(o_j) = conditions(f_j)$
10. $\forall\ o_i$ of u_i **identify** temporal ordering (o_i)
11. **include** $name(f_i)$, $name(u_i)$in the domain dictionary
12. **include** $name(A_i)$ in the domain dictionary

Table 2: Use case metrics

Metrics	Explanation
NS	No.(normal scenarios)
ES	No.(exceptional scenarios)
Op	No.(operations)
$PreC$	No.(pre-conditions)
$PostC$	No.(post-conditions)
A	No.(actors)
$DN_{incorrect}$	No.(instances of improper use case notation)
N_{rep}	No.(instances of repeated use case or actor names)
S_{amb}	No.(statements for which meaning is unclear or ambiguous)

Use Case: PC -SW in SunSensor_Interface
Actors: PC-HW
Main scenario:
- Operation: SunSensor_Signal_Output()
  Pre-condition: {Sun detected}
  Post-conditions: {SunSensor data word values available} AND {data word matches specified format} AND {Values for Na and Nb available} AND {time of computation = cycle tick time} AND {Sun within ± 32 degrees of SunSensor Optical axis} AND {Sun presence bit = 1} AND {{Sun outside ± 32 degrees of SunSensor Optical axis} AND {Sun presence bit =0}

- Operation: Command_Sun_Presence_Bit()
  Pre-condition: {Sun detected}
  Post-conditions: {Sun presence bit value set by user} AND {Values are 1 or 0}

- Operation: Configure_Orientation()
  Pre-condition: Not specified
  Post-conditions: {SunSensor orientation value set by user} AND {Value matches specified format}

- Operation: Configure_equation_coefficients()
  Pre-condition: Not specified
  Post-conditions: {Na and Nb computed as a function of sun angles} AND {sun angles a and b available} AND {user specifies values for coefficients of equations of Na and Nb} AND {Coefficient values match specified format}

Alternate Scenario:
Pre-condition: ¬ {Sun detected}
Post-conditions: Not specified

Use case: AC-SW in SunSensor_Interface
Actors: AC-HW

Scenario (1):
- Operation: SunSensor_Model()
  Pre-condition: {Flight algorithm test mode = true} AND { *{Signal set directly}* OR *{Corresponding angle set}* }
  Post-condition: {Sun Sensor values available from SunSensor_model} AND {output to AC-HW SunSensor stimulus interface}

Scenario (2):
- Operation: SunSensor_Emulator()
  Pre-condition: {Flight hardware test mode = true}
  Post-condition: {output commandable value to AC-HW SunSensor sun emulator interface}

Figure 3: Use case descriptions

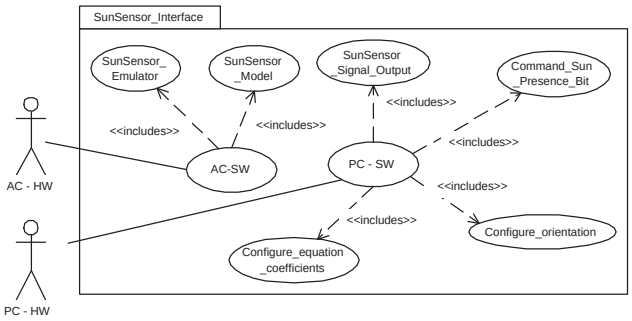

Figure 4: Use case diagram

2.2 Computation of metrics from use cases

Use cases and the associated analysis can be used to compute some simple metrics (table 2) which can be directly obtained from the use cases. Besides these, additional metrics have been developed by Marchesi [8], for use case complexity analysis. Each of these metrics are evaluated for every main use case in a use case diagram. The metrics subscripted with the term *missing* are metrics to measure the related missing parameter as identified by the IV&V analysis. These derived metrics are defined as follows.

$$NS_{total} = NS + NS_{missing} \quad (1)$$

$$ES_{total} = ES + ES_{missing} \quad (2)$$

$$Op_{total} = Op + Op_{missing} \quad (3)$$

$$PreC_{total} = PreC + PreC_{missing} \quad (4)$$

$$PostC_{total} = PostC + PostC_{missing} \quad (5)$$

$$A_{total} = A + A_{missing} \quad (6)$$

$$DN_{incorrect} + DN_{correct} = TotalNo.(Statements) \quad (7)$$

These metrics used as input to the bayesian network developed in section 4.

3 IV&V process flow

The primary function of IV&V at the SRS stage is the identification and mitigation of risks. A proposed IV&V process flow, developed in earlier work [9], for addressing the risks associated with use case based requirements is shown as an activity diagram in figure 5. The tasks identified include aspects of the traditional IV&V analysis criteria at the SRS phase *i.e.* analysing correctness, completeness, consistency, accuracy, readability and testability [3].

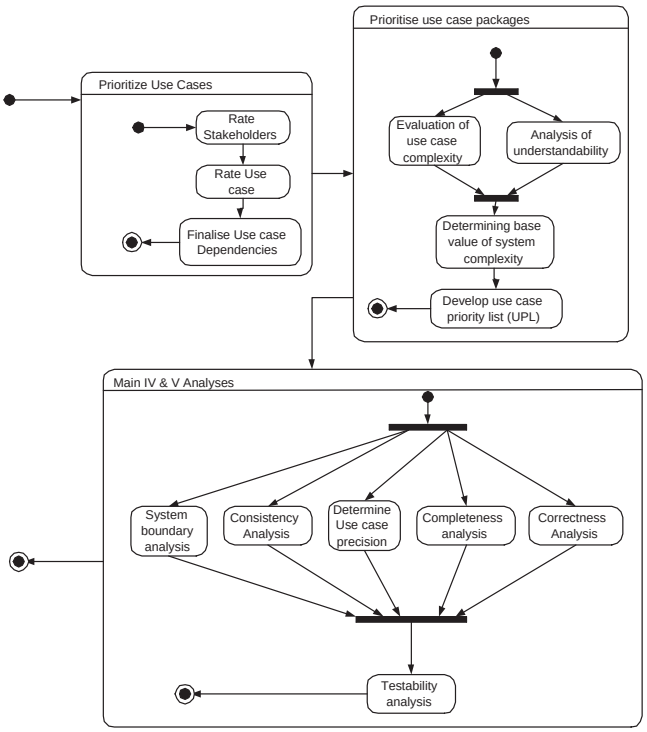

Figure 5: IV&V procedure: use case analysis

This process flow is used to construct the bayesian network with which quantitative analysis can be performed. We note that the observable parameters of some of the activities in the process flow, are the metrics which were described earlier in this paper. Specifically, the main rationale in using a probabilistic network (also known as a Bayesian network (BN)) is formalising the reasoning used during IV&V, by modelling the relationships between the observable (IV&V) process parameters and the desired features of requirements in general *i.e.* clarity, lack of complexity, completeness, correctness, consistence, identification of constraints, traceability and testability. In the next section, we describe the Bayesian network formalism and its application to IV&V.

4 Bayesian networks applied to IV&V

Briefly, a BN is a concise representation of a joint probability distribution on a set of statistical variables, encoding an acyclic graph of nodes and directed edges [10, 11]. The nodes model the variables and the edges model the probabilistic relations between them. Each

node has an associated conditional probability distribution $P(A|\pi(A))$ which characterizes the relationship of the node with its immediate parents $\pi(A)$. The joint probability distribution for a node is computed by marginalization, while the conditional posterior distribution for the nodes given evidence *i.e.* observations about the state of a node, is computed using Bayes' rule. The qualitative part of a BN is encoded in the structure of the digraph, while the specification of conditional probability distributions for the nodes encodes the quantitative portion.

One of the strengths of a BN is that both subjective judgement and empirically obtained data can be used as input. It is flexible enough to allow inclusion of evidence into nodes when it is available. Additionally, the network topology may be changed by adding nodes, or links between nodes. The underlying mathematics permits the propagation of evidence in either direction in the network to update belief. The task of modelling using a BN is essentially one of identifying the correct belief structure, and specifying the conditional probability distributions on the nodes.

4.1 BN structure

Figure 6 shows one possible BN reflecting our reasoning or belief as to the relations between the proposed IV&V process activities and the general properties which need to be checked for maturity assessment of use cases. The nodes with the capitalized names - *Clarity, Complexity, Complete, Correct, Consistent, Constraints, Traceable* - represent the features that are typically desired from a requirements specification document. Each of the parents for these nodes are factors that we believe are relevant in determining the maturity of the requirements by applying the IV&V process flow. The BN was essentially constructed manually by relating the key process activities to the properties of the requirements, that the activities are checking.

For example, the clarity of a use case is being checked by the activities that examine whether the use case boundary and its scope has been clearly defined, as well as by the presence of domain or context information. Complexity, on the other hand is related to the number of interactions among the use cases, between actors and use cases, the *UCP* metric as suggested by Marchesi [8], and on how understandable the use cases are. Consistence and correctness are dependent on whether the result of scenario executions match the expected output, as well as the consistent use of use case modelling rules, unique use case and actor names and use case design rules.

One approach to validate such a network is to actually determine from IV&V analysts whether these are the

Figure 6: Example BN for IV&V

criteria used. The presence of other alternative criteria is easily handled since the network can be modified to include them. We note that the BN shown here is not a comprehensive set of conditions that may be used to assess maturity.

4.2 Specifying root node probabilities

Each node in this BN has three states $\langle True, Unknown, False \rangle$. *True* represents the state where the conditions for that node are true, while *False* represents the state that the conditions for the node are false. The state *Unknown* captures the uncertainty that the analyst is not completely sure whether the node is in either of the former two states. The prior distribution on the root nodes is a specification of the likelihood that the node is in a particular state. Some of these are constructed from the metrics defined earlier in this paper. Other nodes can presently only be quantified from observations of an inspection or from subjective judgement *e.g.* the node *Understandability* which models cognitive complexity. As another instance, consider the node *Completeness*. In an ideal case, a given use case may be determined as being

completely specified at a particular level of abstraction if all of its scenarios, both normal and exceptional have been defined, all conditions existing on the flow of events in the scenarios have been defined, all the actors have been identified, all the pre- and post-conditions have been specified and if scenarios have been walked through to determine exceptional situations.

To specify the probability distributions for the root nodes, the fraction of missing pre- and post-conditions, actors, and scenarios can be computed from equations (1), (2) and (4)-(6). *e.g.* for the node *Post Condition*, we have $P(PostCondition = F) = PostC_{missing}/PostC_{total}$.

In reality, this value is the lower bound since we may not know in the early stages whether more post-conditions have been left unspecified. This uncertainty can be captured in the state *P(Post Condition = Uncertain)*, while *P(Post Condition = True)* = 1 - (*P(Post Condition = False)* + *P(Post Condition = Uncertain)*). Values for *P(Post Condition = Uncertain)* can typically be quantified from historical information of the IV&V process *i.e.* whether the IV&V process has been successful in the past in identifying missing conditions.

We note that if use cases are formally specified, completeness may be automatically checked or a complete set of scenarios may be generated for each use case, as suggested in reference [6]. The advantage of the BN is that such evidence may be directly entered into the nodes. *i.e.* we may specify that (*P(Scenario Completeness = True) = 1*). When lightweight formal methods are used for requirements analysis, we may populate some nodes with information obtained from these techniques while other nodes can be populated with the proposed approach. The most difficult task in the BN based approach is quantifying the conditional probability tables for intermediate nodes. The size of the table grows exponentially with the number of parents and their states. To construct the conditional node probability tables (NPT) associated with child nodes, we use the quantifying judgement (QJ) method [12], summarised in section 4.3.

4.3 Specifying conditional node probability tables

The QJ methodology developed by Donohue *et al.* [12], proposes that the knowledge that a child node is in a particular state is contributed by those parent nodes which are in the same state as the child. Two pieces of information need to be elicited for each of the child nodes:

- *The order of importance of the parent nodes and the relative weights for each node in the order.* This is a partial order, termed as relative contribution value (RCV). The absolute contribution value (ACV) is then constructed by multiplying the weight assigned to the higher ranked parent with the weight assigned to the lower ranked parent which immediately precedes it in the order.

- *The amount of new information added when a particular parent node is added, given that another parent node already exists.* This knowledge, specified either in terms of the fraction of overlap between the ACVs of two nodes or the fraction added, is called the relative added value (RAV). The RAV is elicited relative to the node with lower ACV. The absolute added value (AAV) is computed from the RAV as $AAV_i = ACV_i - (RAV_j)ACV_j$, if the RAV is specified as a fraction of overlap. Alternately, if the RAV is specified as a fraction added, then $AAV_i = ACV_j - ACV_i(1 - RAV_i)$. Here i is the node being considered, while j is the node overlapping with node i. All the 2-way, 3-way and n-way overlaps need to be elicited if the number of parents is greater than 2. For simplicity in computing the AAV we assume that for $n > 4$, the fraction of overlap is zero. Consequently, the number of parents is restricted to at most 4 by construction to facilitate ease of NPT construction and model usage.

Using these values, the NPT is automatically generated and as a function of ACV and AAV values for each child node. For this paper, the RCV and the RAV were elicited from an IV&V analyst who ranked the respective nodes in the BN.

5 Discussion

The original requirements document from which use cases were constructed was specified in natural language. This document was 22 pages long. 14 modules were analysed and there were 52 use cases in all, including primary and secondary use cases. These were described as operations in scenarios of the main use cases. The analysis was conducted by one graduate student, with no relevant domain knowledge of the system, over a period of three months.

5.1 Analysis results

We provide use case analysis results and the computed metrics for the *Digital sun sensor Interface* module in table 3.

Figure 7 shows a snapshot of the BN subnet used to estimate the likelihood that the module *Digital Sun Sensor Interface* was completely specified. As shown in

Table 3: Digital Sun Sensor Interface metrics

Metrics	Value	Node probability
NS_{total}	1 + 0(missing) = 1	P(Regular scenario = F) = 0
ES_{total}	0 + 1(missing) = 1	P(Exception Scenario = F) = 1
Op_{total}	6 + 1(missing) = 7	P(Scenario Walkthrough = F) $\geq$ 0.143
$PreC_{total}$	5 + 4(missing) = 9	P(Pre Condition = F) $\geq$ 0.444
$PostC_{total}$	16 + 4(missing) = 20	P(Post Condition = F) $\geq$ 0.2
A_{total}	2 + 0(missing) = 2	P(Actor = F) = 0
S_{amb}	2	P(Understandability = F) = 0.0833
Size of Use case description	24 statements	

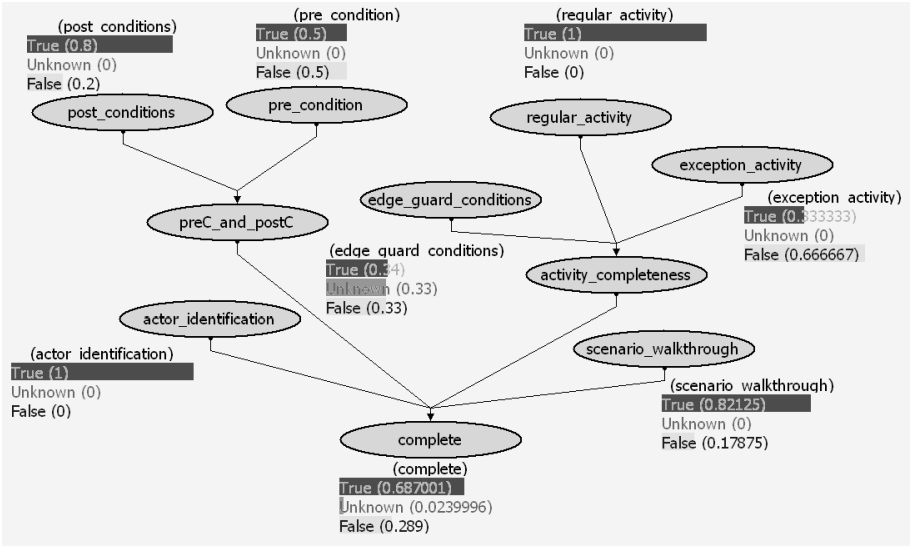

Figure 7: Analysis for node Complete

the figure, the network estimated that about 68% of the module was completely specified.

Figure 8 shows the values that were elicited for the quantifying judgement analysis from the IV&V analyst.

Complete	RCV		RAV	SW	PPC	AC	AI
Scenario Walkthrough	1		SW	1			
Pre and Post Condition	2		PPC	0.3			
Activity Completeness	5		AC	0.6	0.7		
Actor Identification	5		AI	0.65	0.55	0.1	

Figure 8: Elicited values for QJ

Table 4 shows the probability distribution over all the properties that were assessed using the BN for the module *Digital Sun Sensor Interface*.

In analysing all the BNs, we made the following assumptions: the lower bound of the metric values was used to populate the nodes. For the nodes where metric values were available, the likelihood of a node being in state *Unknown* was assumed to be zero. Since edge guard conditions are applicable to conditions specified as a statechart diagram, and since this conversion was not performed, we assumed that it was equally likely that this node was in one of its three states. Also, since constraint information was not specified in the requirements for the module, this subnet in the BN was pop-

Table 4: Complete BN analysis: DSS Interface

Node	Probability distribution {T,U,F}
Clarity	{0.65, 0.34, 0.0}
Complexity	{0.607, 0.345, 0.048}
Complete	{0.687, 0.024, 0.289}
Consistence	{0.5672, 0.2038, 0.2289}
Correct	{0.5127, 0.1310, 0.3561}
Traceable	{0.693, 0.34, 0.0}
Constraints	{0, 1, 0}

ulated with (*P(state = unknown)* = 1) for all its parents. Additionally, the probability distribution for the node *Understandability* was constructed as the fraction of the statements that were ambiguous to the size of the use case descriptions, although this is not a measure of cognitive complexity of the use cases.

In general, the original requirements document was carefully constructed, went through at least three revisions and was relatively consistent in terminology. The lack of sufficient domain information, however, resulted in increased cognitive complexity of the document. Completeness and correctness could not be checked apart from the analysis performed. This requires more rigorous examination of the requirements and the application of formal methods, which was out of the scope of the intended work. Although use cases were not directly available, the procedure for structuring the natural language requirements into use cases provided insights into identifying missing scenarios, ambiguous conditions and in general, understanding the requirements better.

5.2 Interpretation of the BN analysis

To apply the BN for maturity analysis of all the requirements, we observe that each element of table 4 is a probability distribution for the event that a certain property of a set of requirements captured as a use case has been met to a certain degree. Recalling that the exit criterion for the IV&V process at the SRS stage is that at least 95% of the requirements should be mature before development can proceed. This can be considered as determining whether each element in table 4 has a value of least 0.95, as the probability of being in the state *true*.

A comprehensive argument for requirements maturity, or the lack thereof, can be systematically built by evaluating table 4 for every use case in the set of software requirements. Essentially, we can build a quantitative case to establish whether the exit criterion has been met by every use case and in turn, the requirements specification. Table 4 by itself provides a concise representation of potential problem areas to the development team or the customer. For example, the node *Correct* has a low probability of being in state *true*. Thus, effort can be diverted to algorithm and control flow analysis. Consequently, in the early stages, the network is valuable in the decision making process and in identifying potential problem areas of the requirements, where resources should be deployed. This is augmented with the BN structure which formalises the IV&V process and the underlying reasoning used to arrive at the assessment of maturity.

The use case based analysis and the resulting metrics provide an empirical means to quantifying the probability distributions for some nodes in the BN. Quantifying the NPT in the BN required an elicitation of the relative importance and information contributed by the parent nodes. Since the analysis is being performed at a very early stage, the true nature of the relations between the nodes is yet unknown. Hence, a subjectively qualified NPT is an attractive option. Thus, the variance in the model results and the actual values is to be expected. These are primarily due to the uncertain nature of the NPT itself.

The semantics of the BN NPT are versatile enough to model not only deterministic relationships but also continuous and discrete probabilistic relationships. Thus a BN has a greater expressive power over a functional form or a checklist based approach for maturity assessment. It is also worthwhile to reiterate that probabilities can be propagated in the reverse direction in the BN. This permits reasoning about flaws in the process itself. That is, supposing the model estimates low maturity for a certain parameter but an alternative approach provides evidence that this assessment is incorrect, the evidence can be incorporated into the network to reason about the potential contributors to the incorrect assessment. Thus, the network can be tuned and its estimations can be refined. Furthermore, there also exist probabilistic networks where intervals can be propagated instead of point values, as has been done in this paper. Thus it is possible to represent the states of the nodes as distributions rather than point values, to account for early variance in model estimations.

It is possible, with historical information about IV&V process activities and a relatively small amount of relevant data, to learn the conditional probability distributions for the nodes. Furthermore, as development and IV&V teams iterate, the nature of the relations between the nodes becomes more apparent. The NPT can then be updated and model results can be refined.

6 Related work

Use case based techniques have been applied for verification and validation by Winter *et al.* [13]. In this work, use cases are coupled to class diagrams by specifying scenarios using extensions to the activity diagram construct of the UML, and by mapping both activity diagrams and class diagrams to activity graphs. The authors precisely define the extensions for activity graphs and activity diagrams that facilitate this coupling. The main applicability of such coupling seems to be in verifying that designs represented by class diagrams are consistent with the use case based requirements specifications. With respect to validation, the authors propose inspection and scenario walkthroughs as the mechanism

of choice and develop a tool to assist in V&V. Sutcliffe *et al.* [6, 14] present two alternative approaches for V&V of scenario based techniques. In the first of these, use cases and the related modeling notations are formally defined using temporal semantics and a tool is built to automatically generate all possible normal scenarios. Exceptional scenarios may either be automatically constructed or the tool can be used to assist users when they specify scenarios. In the second approach, the authors build a BBN of human error assessment and use this in the validation of system performance and reliability requirements. Easterbrook *et al.* [15, 4] have addressed the use of lightweight formal methods for IV&V, however, they do not address the applicability of their techniques to use cases.

In general most techniques address V&V at the SRS stage and advocate a formal approach to use case analysis. However the context of existing work is V&V that is to be performed during the development process. They do not address the applicability of the techniques to use cases in the IV&V context where rapid analysis is required. This is the main point of difference between our work and the existing work in the literature. Our work also differs in identifying activities for IV&V of UML use cases and in building a BN model to systematically and quantitatively assess maturity. The proposed IV&V process can be easily used in conjunction with lightweight formal techniques. These may be selectively applied to use cases that have been identified as having a high priority for IV&V. For example, it is straightforward to map the operations in the use case descriptions (figure 3) to the ECATAB model developed by Gervasi *et al.* [7].

It is clear that a significant degree of assurance for IV&V cannot be achieved by using informal and ad-hoc techniques. The process of formalising the requirements helps identify problems early on. We support this notion and observe that this may be used regardless of whether requirements are translated into use cases or if use cases are the baseline model for requirements. The BN based approach can easily aggregate information from both formal and informal IV&V techniques; results from the formal analysis can be confidently used by representing these results as "hard" evidence.

7 Conclusions

Our contribution in this paper, is the development of a Bayesian network using the IV&V process flow, for use case analysis. The input to the network are metrics and subjective information obtained from the IV&V process activities. This information can be combined in the BN to systematically and quantitatively assess the maturity of evolving requirements. The BN provides a means to formalise the IV&V process, the underlying reasoning and its results are a concise representation of potential problems areas within the requirements.

Acknowledgements

We would like to thank the NASA IV&V center, which supported this work under NASA Grant NAG5-11953, TITAN Corporation, and Mike Chapman at WVU/NASA IV&V Center for supplying the example system used in this paper.

References

[1] R.R. Lutz, "Analyzing software requirements errors in safety-critical embedded systems," in *Proc. of the Intl. Symp. on Req. Eng.*, Jan. 1993.

[2] B.W. Boehm, *Soft. Eng. Economics*, Prentice Hall, 1981.

[3] NASA Software IV&V Facility, "Software independent verification and validation handbook for program management," NASA GSFC technical report, Aug. 2000.

[4] S. Easterbrook et al., "Experiences using lightweight formal methods for requirements modeling," *IEEE Trans. on Soft. Eng.*, vol. 24, no. 1, pp. 4–14, Jan. 1998.

[5] V. Gervasi and B. Nuseibeh, "Lightweight validation of natural language requirements: A case study," in *Proc. of the 4th Intl. Conf. on Req. Eng.*, June 2000.

[6] A.G. Sutcliffe, et al., "Supporting scenario-based requirements engineering," *IEEE Trans. on Soft. Eng.*, vol. 24, no. 12, pp. 1072–1088, Dec. 1998.

[7] V. Gervasi and B. Nuseibeh, "Lightweight validation of natural language requirements," *Software - Practice and Experience*, vol. 32, pp. 113–133, 2002.

[8] M. Marchesi, "OOA metrics for the Unified Modeling Language," in *Proc. of the Euromicro Conf. on Soft. Maintenance and Reengineering*, Mar. 1998.

[9] K. Lateef, "OO V&V requirements techniques," Technical report, NASA Software IV&V Facility, Aug. 2003.

[10] J. Pearl, *Probabilistic reasoning in intelligent systems*, Morgan Kaufmann, 1988.

[11] F.V. Jensen, *An Introduction to Bayesian Networks*, Springer-Verlag, 1996.

[12] S.K. Donohue and J.B. Dugan, "Transforming expert opinion into subjective conditional probabilities: the quantifying judgement methodology for bayesian belief networks," *In review, IEEE Trans. on Knowledge and Data Eng.*, July 2004.

[13] M. Winter et al., "Coupling use cases and class models as a means for validation and verification of requirements specifications," *Req. Eng. Jrnl.*, vol. 6, no. 1, pp. 3–17, 2001.

[14] A. Sutcliffe and A. Gregoriades, "Validating functional system requirements with scenarios," in *Proc. of the IEEE Joint Intl. Conf. on Req. Eng.*, 2002.

[15] S. Easterbrook and J. Callahan, "Formal methods for V&V of partial specifications: an experience report," in *Proc. of the Intl. Symp. on Req. Eng.*, 1997.