Tackling The Question, Is My Software Ready To Be Shipped To The Customer?

Khalid Lateef[1], Ganesh J. Pai[2], Joanne Bechta-Dugan[3]
[1]Titan Corporation, Greenbelt, Maryland, USA
[2,3]University of Virginia, Dept. of ECE, Charlottesville, VA, USA
E-mail: [1]klateef@ieee.org, [2,3]{gpai | jbd}@virginia.edu

***Abstract* – An important question for any project team is to assess if the project deliverables have met a pre-determined "readiness" level. This becomes even more important when a software development project gets close to a version release date. In practice, this is quantified using engineering judgment. We describe the application of Bayesian Belief networks (BBN) to analyze the readiness of software requirements specifications during software requirements review (SRR) milestone. The method proposed in this paper brings an additional level of rigor to the assessment process. Starting with a data-flow model of the Software requirements development model, we construct a BBN semi-automatically. Then, we provide a quantitative interpretation of the readiness level of software requirements in terms of artifact properties. The results of the preliminary assessment are used as evidence in the BBN to obtain a posterior distribution of readiness. We illustrate our approach by applying it to two example systems.**

***Index Terms* – Requirements analysis, Bayesian Belief Network, iterative development, software readiness assessment.**

I. Introduction

This paper relies mainly on a previous work by the author and the co-researchers [12], on software assessment with a focus on Independent verification and validation (IV&V). However this paper broadens the scope of our work by proposing models and algorithms for the software development teams. Lightweight formal methods can be applied at less critical levels to maintain rigor. It is especially valuable to evaluate requirements readiness, since requirements errors are most often responsible for software failures and they are the most expensive to correct when not discovered early in the development lifecycle [3].

The scope of this paper is limited to the requirements phase of an iterative development lifecycle. In future we plan to extend this work and implement models for design, implementation, and test phases.

One of the main benefits of using formal techniques is that it provides a clearer and more confident assessment of the readiness level. However, in practice, formal techniques are mainly applied to critical elements while the rest of the analysis is comparatively less rigorous. Consequently, the readiness evaluation of an artifact is fuzzy and it may be quantified using engineering judgement.

We use Bayesian networks (BBN) to assess the readiness level of software artifacts, specifically requirements specifications by combining evidence from diverse sources. The overall approach is to first build an annotated process model which describes the requirements analysis process, the entities involved in the process and their respective properties. Thereafter we build the BBN structure semi-automatically from the process model to encode the case for readiness i.e. the rationale with which the analyst can argue whether an artifact is ready at the desired level (Sections 2 and 3). The BBN numerical specification is generated from elicitation, empirical data and from metrics applied to the artifacts. We characterize readiness in terms of the generic properties desired from requirements specifications and describe how to quantify it as a conditional probability distribution (Section 4). To illustrate our methodology, we apply it to software requirements specifications for two real example systems (1) a fault protection system of a space probe, and (2) the non-volatile memory management module for a second aero-space system (Section 5).

II. Modeling the Requirements Analysis process

The overall process for requirements analysis can be specified as a set of work instructions or guidelines which provide a high-level overview and a flow of the activities that the requirements analysis team must perform. Our rationale is that modeling the requirements analysis process to capture such diverse factors as skill/expertise of the analysis team or compliance with process specifications will provide insight into the artifact readiness level and the reasoning used to arrive at the assessment. To model this process, we use the dataflow within a process, where a process or process activity has input entities, is enacted by agents and produces output entities. Each of these can have properties which can be quantified or qualified.

The general idea is to first build an annotated requirements development process model and identify the entities involved with their respective properties. Thereafter we

build the BBN structure semi-automatically from the process model to encode the case for readiness i.e. the rationale with which the analyst can argue whether an artifact is ready at the desired level. The process model follows practices as recommended in the IEEE-1012 and DOD-2167 standards.

Figure 1 shows a simplification of the requirements analysis process activities and input and output entities at the requirements development phase; some of the properties of interest are also shown. This model itself is a refinement of a higher level process which includes activities such as the identification of relevant inputs, and criticality based prioritization of artifacts for review.

The box labeled "Software requirements analysis" shows some of the activities prescribed in the literature [2] for requirements analysis. The model captures the notion that software requirements analysis takes as input not only the prioritized software requirements, but also system software documentation, system requirements, and relevant interface documentation. Additionally, the model also captures the idea that the team which enacts the process may use tools or approaches tailored for a particular domain. The suitability of these agents influences the quality of the analysis to some degree since these agents are used to execute the process.

We can further decompose the sub-activities shown, to include activity specific tasks: for example, in analyzing whether a requirement described using scenarios is internally complete, some of the tasks would include checking that (for a required functionality): all scenarios and their relevant pre- and post-conditions have been defined, the conditions for any temporal transitions within the scenarios have been defined, all relevant actors have been identified and that the scenarios identified can be stepped through to completion. Again, such tasks may be performed using formal methods if the requirements are critical or using relatively less rigorous techniques otherwise.

For our purpose of building a BBN, this relatively informal process model suffices. Furthermore, it is simple and high-level enough to build so that the requirement analysis team can quickly and easily build a model of their process. This model was informally validated by an analyst who had implemented these processes from an original set of work instructions. Once the process models are built, we semi-automatically build a BBN from the process models.

III. Constructing the BBN Model

A. *Bayesian networks*

A BBN is a concise representation of a joint probability distribution on a set of statistical variables, encoded as an acyclic graph of nodes and directed edges [4]. The nodes model random variables which can be discrete or continuous. Edges model the probabilistic relations between the nodes. 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, whereas 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 BBN is encoded in the structure of the digraph, while the conditional probability distributions for the nodes encode the quantitative portion. One of the strengths of a BBN is that both subjective judgment and empirical data can be used as input. Furthermore, as evidence becomes available, we can update the model and refine its assessments. The fundamental tasks of mathematical modeling using a BBN are: (1) identifying a belief structure that best describes the phenomenon being modeled, and (2) specifying the conditional probability distributions on the nodes.

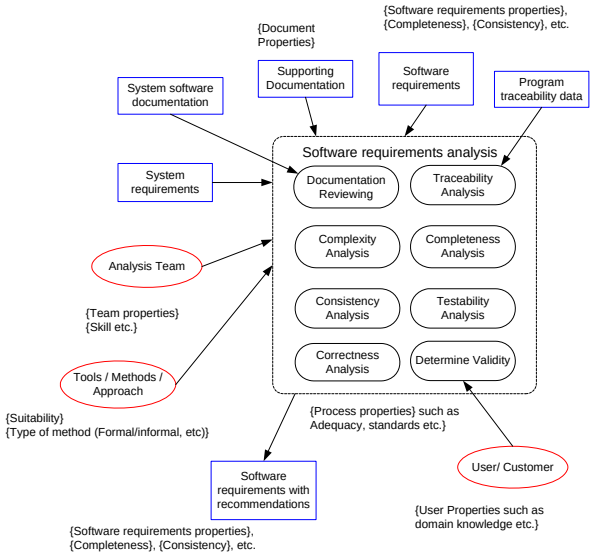

Figure 1. Requirements analysis process model

The BBN structure is built semi-automatically from the process model [5]. Briefly, this method constructs nodes for each of the entities in the process model and for their properties. It appropriately directs arcs between the nodes, such that the observable variable is a leaf node, i.e. from entities to their properties. It also directs arcs from the nodes for the input, and the process entities, to the nodes for the output entities. Since entities relevant to a process activity can exist across levels of abstraction, the algorithm generates subnets which repeat at these different abstraction levels. Although such subnets can be pruned algorithmically, checking whether it produces the correct belief structure is important, i.e. we want to inspect whether the model makes practical sense. Currently, we prune the model manually as this gives us the opportunity to modify the structure of the

BBN to account for dependencies between nodes that may not have been explicitly captured or may not be representable in the process model. Thus, the procedure mainly builds a generic structure of dependencies; the BBN used for analysis may be modified by appropriately re-directing transitions or by including or eliminating nodes.

An additional change to the network structure is binary factorization [6], which splits nodes with three or more input arcs and creates logical intermediate nodes that preserve the numerical specification of the original network. Essentially, this operation simplifies both computation and specification of conditional node probability tables. The latter is beneficial since the tables increase exponentially in size with the number of parents and their respective states.

B. *BBN model for Requirements Assessment*

A partial BBN obtained is shown in figure 2, labeled as the subnet *software requirements analysis*. *Subnet 1* models the influence of process activities on *Readiness*. In this paper, we refine the completeness analysis activity of the software requirements analysis process. The BBN for this activity are represented in *subnet 1a* and *subnet 1b*.

Similar subnets are generated for the remaining activities in the requirements analysis process. Shaded nodes represent nodes replicated across levels of abstraction as a result of algorithmic construction. We prune the BBN by simply eliminating subnet 1a, since the influences modeled by this subnet has already been captured. The resulting two BBN can be pruned further by re-directing arcs between the appropriate nodes. Alternatively, we can simply use the data obtained from the lower level BBN as evidence for the higher level BBN. The node labeled "SW requirements with recommendations" has arcs to its properties, of which two have been shown in the figure *i.e. Completeness* and *Correctness*. Together, these nodes compose the subnet *Readiness*, and the probability distribution on the root node in this subnet i.e., *SW requirements with recommendations* represents the evaluation of readiness at the software requirements review milestone.

C. *Numerical specification*

Specifying root and intermediate node probabilities in the BBN encodes its quantitative portion. These model the nature and the weight of the probabilistic relations between related nodes. The relation between nodes need not be only probabilistic. However a deterministic relationship can be easily transformed into the appropriate probabilistic function. For root nodes (nodes without parents), we specify a (prior) probability distribution which reflects either the analysis team's initial belief or the available data. Typically, prior distributions are specified such that all available data are considered. In the absence of relevant data, an alternative to a subjective prior is a non-informative prior [7].

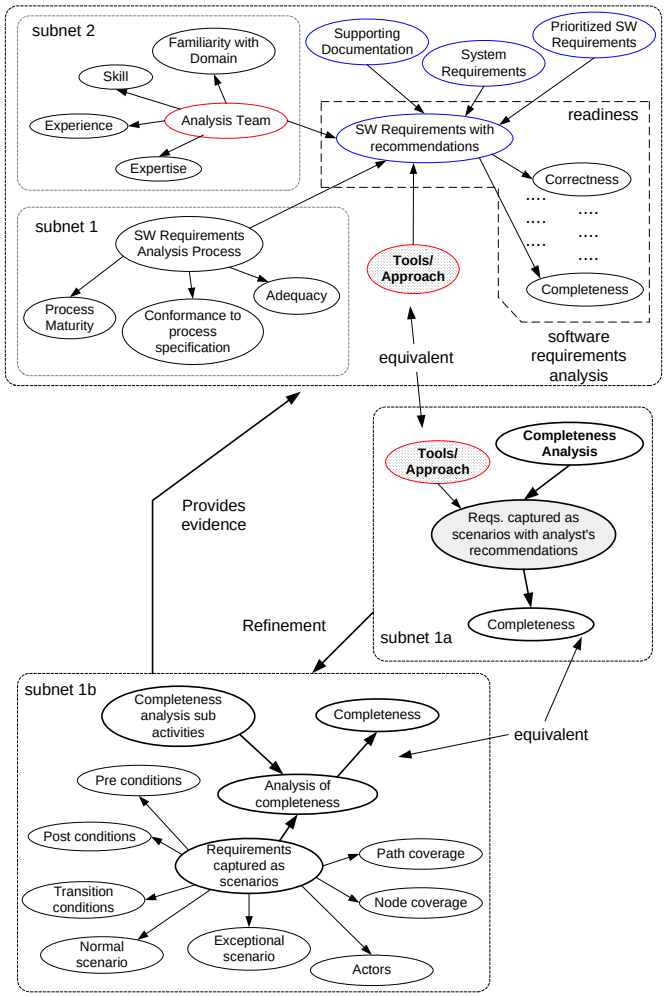

Figure 2: Partial BBN structure for the process of requirements analysis

We model each node as random variable (say X) whose states are mapped to an ordinal scale. *i.e. X: {Very Low, Low, Medium, High, Very High}*. These states can be further mapped to either a monotonically increasing, continuous numeric scale or a discrete numeric scale with integer values. The probability distribution across its states can be specified from (1) historical data or (2) a prior belief (such as a uniform distribution). Given data, i.e. observations (θ) on the set of child nodes ($\boldsymbol{\Theta}$), Bayes' rule is used to compute the posterior distribution ($p[X|\ \boldsymbol{\Theta} = \theta]$). For intermediate or child nodes (nodes with parents), we specify a continuous conditional probability density function (pdf); then, the corresponding discrete distribution is easy to build. For our analyses, we use a simple procedure developed by Neil et al. [6] to construct the conditional pdf as a tail-truncated normal distribution tN(μ, σ). This is a normal distribution truncated at both tails and normalized such that the resulting distribution is proper (integrates to 1).

IV. READINESS OF AN ARTIFACT

As mentioned earlier, analysts are typically required to assess the 'readiness' or 'maturity' of an artifact at a milestone in the process. Admittedly, readiness is an imprecise and fuzzy term; this value is, in practice, largely quantified by using engineering judgment. Provided it meets some pre-determined value, development is allowed to proceed to the subsequent development phase. Readiness assessment is essentially an inference task, *i.e.* the requirements team constructs a case for readiness based on the results of their analyses. The process model and the resulting BBN formalize the procedures and the evidence used to construct this case.

To mathematically characterize readiness, we model the readiness of an artifact as a discrete random variable M, qualified on a five point ordinal scale, i.e. *M: {very low, low, medium, high, very high}*. Properties desired from an artifact are modeled as a vector of random variables **X**, each of which can assume some state ***x***, also on the same five point scale. Readiness is then, some probabilistic function of the artifact properties and it is assessed as:

$$p[(M \geq m)|(\mathbf{X} = \{\boldsymbol{x}\})] \quad (1)$$

Given some initial prior distribution on M, results of analysis provides evidence to **X** and equation (1) is the posterior distribution of (M|**X**) computed using Bayes' rule. The result is the probability that the *readiness level* is some value m. In figure 2, this is modeled as the subnet readiness, where the root node *SW requirements with recommendations* represents requirements readiness as function of the desired properties of requirements. Thus, we can interpret readiness (1) with respect to individual properties or (2) as a function of all the properties. The former allows us to make statements of the form "The requirements are ready at a level M with respect to property X_1, but not with respect to property X_2".

V. APPLICATION TO EXAMPLE SYSTEMS

A. Non-volatile memory load component example

For this system, analysis was performed mainly for the requirements analysis stage and is scenario and inspection based. Since the example provided here is mainly to illustrate our approach, we describe analysis of one of the criteria, i.e. *completeness*, which composes overall component *readiness*. The requirements for the non-volatile memory load component were expressed as use-cases and were supplemented with natural language descriptions. In addition to use cases, design stage models for the component were also available in terms unified modeling language (UML) constructs *i.e.* class diagrams, statechart diagrams, component diagrams, etc. Figure 3 shows the use case diagram of the module and figure 4 shows the natural language specifications corresponding to the module from the requirements specifications document.

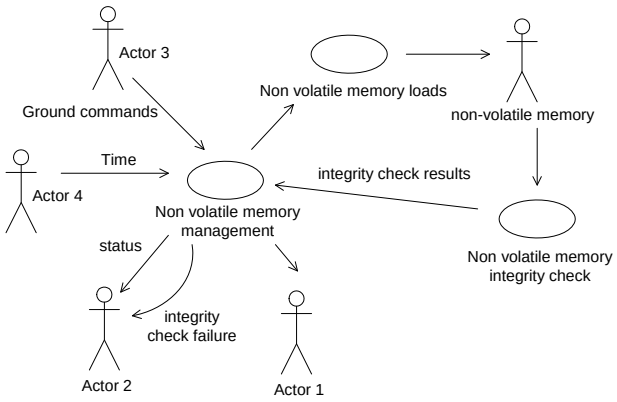

Figure 3. Use case diagram of "non-volatile memory load" operations

> The EX-3-MODULE-1 FSW, in Initialize Mode and Ground Load State, shall load data from the ground into non-volatile memory upon receiving non-volatile memory load command packets.
>
> A1. The EX-3-MODULE-1 FSW, upon receipt of an off-SBC non-volatile write request, shall append Reed-Solomon parity symbols and write the requested data into the specified non-volatile memory
>
> A2. The EX-3-MODULE-1 FSW shall load the data given at the address specified into non-volatile memory upon receipt of a non-volatile memory load command.
>
> A2.1. The EX-3-MODULE-1 FSW shall reject the non-volatile memory load command and report the rejection to Actor 1 if the load address, load size, or actual data length are invalid.

Figure 4. Natural language specifications for non-volatile memory load operations

B. Example analysis

The analysis procedure was essential to build a structured use-case description from the natural language specifications, with minimum changes to the text of the natural language, so as to try and preserve the original intent of the specifications. We believe that organizing the natural language into a structured description permits us to identify methodically, missing conditions, scenarios and statements with ambiguity. Of course, we may also convert such a structured description into formal statements and apply formal analysis to reason about the desired properties of the requirements specifications. Table 1 provides metrics which were computed once the structured use-cases were constructed. Other techniques such as use-case animation or executing the specifications may also be applied.

TABLE 1. METRICS COMPUTED ON USE-CASE SPECIFICATION

Metric	Symbol	Value	
Actors	A	1	
Missing actors	$A_{missing}$	0	
	$A_{TOTAL} = A + A_{missing}$		1
Normal scenarios	NS	1	
Missing normal scenarios	$NS_{missing}$	1	
	$NS_{TOTAL} = NS + NS_{missing}$		2
Exceptional scenarios	ES	1	
Missing exceptional scenarios	$ES_{missing}$	1	
	$ES_{TOTAL} = ES + ES_{missing}$		2
Operations		2	
Missing operations		≥ 2	
	$Op_{TOTAL} = Op + Op_{missing}$		≥ 4
Invariant conditions	I	0	
Missing invariant conditions	$I_{missing}$	2	
	$I_{TOTAL} = I + I_{missing}$		2
Pre-conditions	PreC	5	
Missing pre-conditions	$PreC_{missing}$	15	
	$PreC_{TOTAL} = PreC + PreC_{missing}$		20
Post-conditions	PostC	3	
Missing post-conditions	$PostC_{missing}$	≥ 2	
	$PostC_{TOTAL} = PostC + PostC_{missing}$		≥ 5

TABLE 2. EVIDENCE PROVIDED FOR COMPLETENESS ANALYSIS

Node	Node State
Exceptional Scenarios	0.4 – 0.5
Normal Scenarios	0.4 – 0.5
Actors	0.9 – 1.0
Operations	0.4 – 0.5
Post conditions	0.4 – 0.5
Pre-conditions	0.2 – 0.3
Invariant conditions	0.0 – 0.1

Scale: (0.0 – 0.2: Very low), (0.2 – 0.4: Low), (0.4 – 0.6: Medium), (0.6 – 0.8: High), (0.8 – 1.0: Very high)

Given the metrics computed on the use-case specifications and the BBN constructed from the process model, we may use the BBN which models the activities that evaluate completeness of use-cases. Figure 5 and Table 2 show the BBN model and the evidences provided to the BBN, respectively. Essentially, we compute the evidence from the metrics; the nodes in the BBN are assigned an initial prior distribution, assuming that child nodes are normally distributed as functions of the weighted average of parent nodes. Additionally, a variance factor expresses our degree of belief in the priors [6].

Given our assumptions of the priors and the evidence obtained from the analysis, the BBN computes the level of completeness for these specifications as approximately *Medium*, with ~37% probability. Since the specifications and the design approach uses a model based development approach, we provide an initial assumption of *High quality* for the tools and approaches used during the requirements phase.

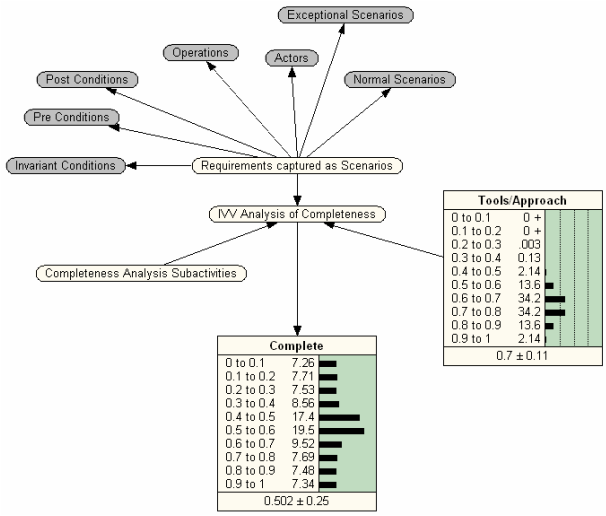

Figure 5. Example analysis (completeness) OOD-EX-3 system

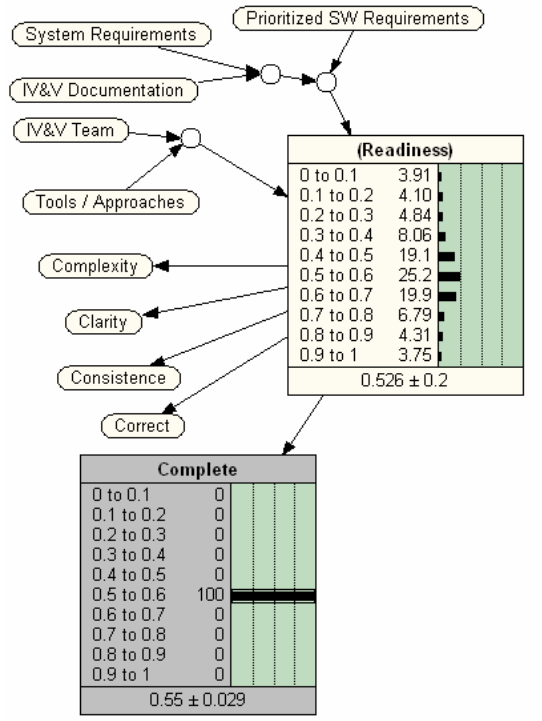

Figure 6. Example analysis of readiness given completeness

Figure 6 shows the BBN model for readiness computed given completeness at level *Medium*. Essentially, we use the BBN of figure 5, to provide evidence into the BBN at the higher level of abstraction. Given completeness alone and that other nodes are in an unknown state, *Readiness* of

the non-volatile memory load module requirements is between *Medium* and *High*.

To illustrate how the BBN can be used to evaluate readiness given all other properties, we make certain assumptions for the other nodes in this BBN. These assumptions are stated in table 3. For example, we assume that the system requirements have a *high* quality; the consistency of the requirements is *medium,* whereas the properties of correctness and clarity are in state *high*. We observe that given these assumptions, the BBN computes the readiness of the requirements to lie at a *High* level, with approx. 75% probability (Figure 7).

TABLE 3: EVIDENCE PROVIDED FOR READINESS ANALYSIS

Scale: (0.0 – 0.2: Very low), (0.2 – 0.4: Low), (0.4 – 0.6: Medium), (0.6 – 0.8: High), (0.8 – 1.0: Very high)

Node	Prior	Node State (Evidence)
System Requirements	N(0.5,0.1)	0.7 – 0.8 (High)
Documentation	N(0.4,0.1)	0.5 – 0.6 (Medium)
Team	N(0.5,0.1)	0.7 – 0.8 (High)
Tools/Approaches	N(0.5,0.1)	0.7 – 0.8 (High)
Prioritized SW Requirements	N(0.5,0.1)	No evidence
Consistence		0.5 – 0.6 (Medium)
Complexity[1]		0.7 – 0.8 (Low)
Correct		0.7 – 0.8 (High)
Complete		**0.4 – 0.6 (Medium)**
Clarity		0.7 – 0.8 (High)

C. *Fault protection system example*

In this section, we apply our methodology to evaluate readiness for a fault protection system of the a deep space probe. For this system, lightweight formal methods had been applied by Easterbrook *et al.* [8] at the requirements specification stage, resulting in a finding of 37 issues. To summarize these, there were 11 undocumented assumptions, of which some were significant, 10 cases of inadequate requirements, 9 inconsistency problems, 6 cases of ambiguous terminology and 1 logical error. From the details of their process, we provide prior distributions for the BBN nodes (table 4).

Our rationale for using these priors is as follows: the documentation, and the tools used were qualified as having the state High since the application of formal methods provides a stronger assurance of detecting potential errors. The state of the node Team was qualified as *High* since the team performing the analysis had expertise in performing both formal and informal analysis. The prioritized software requirements and system requirements were qualified as having the state *Medium* as we believe that the developers of the requirements have several years of domain experience in building space probes. Therefore the requirements would be expected to have an appreciable quality a priori.

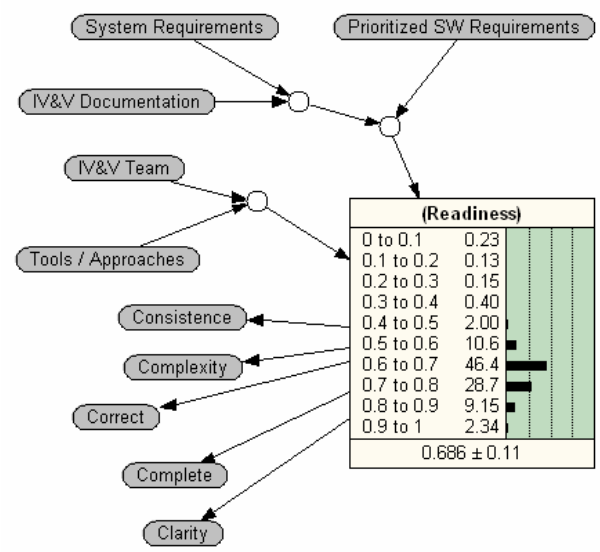

Figure 7. Example analysis of readiness given completeness and other properties

TABLE 4. PRIOR DISTRIBUTIONS FOR BBN

Node	Prior	Node State
System Requirements	N(0.6,0.2)	Medium
Documentation	N(0.75,0.1)	High
Team	N(0.75,0.1)	High
Tools/Approaches	N(0.75,0.1)	High
Prioritized SW Req	N(0.5,0.1)	Medium

Since more information about the particular process used for this system was unavailable, we directly provide evidence to the BBN shown in figure 8. The result of analysis is interpreted as pessimistic evidence and this evidence is used to assess the readiness of the fault protection system requirements. In this figure, most of the mass of the distribution of M is defined over the interval [0.3-0.5) with more than 50% of the mass in the interval [0.3-0.4) ↔ {Low}. The variance of the distribution is also small indicating a greater degree of credibility.

We feel that this assessment of the readiness level and the decision to revise the requirements (*i.e.* not proceed to the development phase) is consistent with the results of analysis in reference [8].

VI. RELATED WORK

Neil *et al.* have conducted research on building object-oriented BBN from process models [9]. Their work models the underlying process of inference and represents the BBN at a higher level of abstraction. Our work differs primarily in modeling an enacted process and building the BBN from the parameters of the input entities, and the process itself.

1 Complexity is measured on the same scale but has an inverse relationship with readiness. Hence higher values correspond with lower complexity.

To the best of our knowledge, readiness assessment in a systematic and quantitative fashion in the context of requirements analysis has not been performed before. Bayesian networks have been used in analyzing software quality [10]; however, quality has been assessed in terms of defect content of artifacts. In our work, BBN are applied to the requirements analysis process to estimate readiness in terms of artifact properties. The notion of modeling processes is not new [11], however process models have been mainly used to specify and simulate processes.

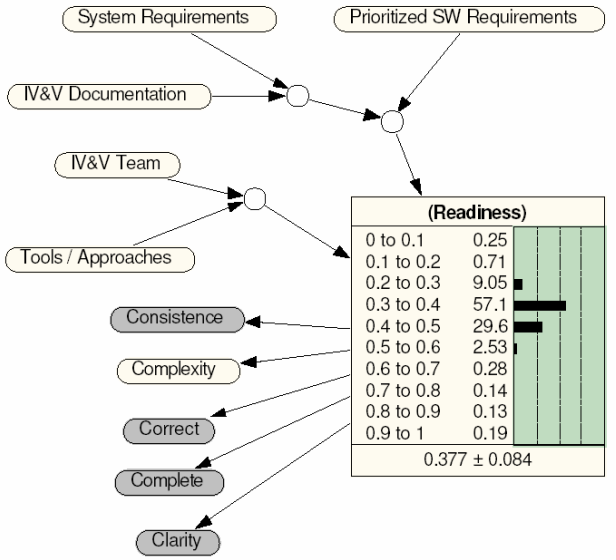

Figure 8. Readiness: fault protection example

Our notation is currently simple enough to model dataflow in a process and is sufficient for building a BBN. It is straightforward to extend and formalize the model or map it to existing process modeling formalisms so as to get the benefits of traditional process models. Additionally, we use the process model to analyze how properties of the process and its inputs influence the output of the process.

VII. Summary and Conclusions

The BBN model constructed from the requirements process model captures the diverse factors that affect the readiness of an artifact. The assessment of readiness is performed by obtaining evidence from analysis and using this data in the BBN model. In the light of evidence, the assessment of readiness is updated indicating the likelihood that it is in some particular state.

The BBN also models the intuitive notion that an informal process is less likely to detect errors or issues in an artifact as compared with a process that employs formal methods. This is evident from the variance parameter for the readiness distributions, shown in figures 7 and 8. The latter has a lower variance indicating that we are more confident in this assessment of readiness (as we should be, given that formal methods were used). Thus, the BBN allows using data from both formal and informal processes in analyzing readiness. In practice, since both of these are employed in the analysis of a complete system, the BBN provides an elegant framework to aggregate arguments from both sources. Additionally, parameters which influence the analysis process, and in turn the assessment of readiness of the artifact, (such as documentation, appropriateness of the tools or methods used, expertise of the analysts) are also modeled. We can measure properties of interest in the artifact and use these easily within the BBN formalism.

The BBN numerical specification requires an identification of prior probability distributions (both conditional and unconditional). These are specified either by elicitation from expert opinion or from empirical/ historical data. We assess readiness using the results of analysis as evidence in the BBN. Evidence can be supplied in the form of metrics applicable to artifacts, problems found from formal or informal analysis, etc. The BBN is capable of modeling the intuitive notion that an informal process is less likely to detect errors or issues in an artifact as compared with a process that employs formal methods. We validate the requirements analysis process model and in-turn the BBN model via consultation with practitioner(s).

The BBN encodes a comprehensive argument for *artifact readiness level*, quantifies this level, and indicates potential problem areas to the development team/ customer. Thus, the BBN structure is a mechanism to formalize (1) the requirements analysis process and (2) the underlying reasoning used to assess readiness. Thus, this approach provides a mathematical basis for the so-called "go/no-go" decision. The BBN is versatile enough to model both probabilistic and deterministic relations. Consequently, it has greater expressive power compared to a functional form or a checklist based approach for assessing readiness levels.

Acknowledgements

We would like to thank the NASA which supported this work under Grant NAG5-11953 and TITAN Corporation for supplying the example system used in this paper.

References

[1] M.V. Zelkowitz and I. Rus, "Understanding IV&V in a safety critical and complex evolutionary environment: The NASA Space Shuttle program," in *Proceedings of the International Conference on Software Engineering*, 2001.

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

[3] R. Lutz, “Analyzing software requirements errors in safety-critical embedded systems,” in *Proceedings of the IEEE International Symposium on Requirements Engineering*, Jan. 1993.

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

[5] G.J. Pai, K. Lateef, and J.B. Dugan, “Bayesian networks applied to software IV&V,” in *Proceedings of the 29th Annual IEEE/NASA Software Engineering Workshop*, April 2005.

[6] M. Neil *et al*., “Modeling subjective causes and objective consequences in Bayesian networks,” *Technical Report 101 v2.1*, Dept. of Computer Science, Queen Mary, University of London, Nov. 2003.

[7] G.E.P. Box and G.C. Tiao, Bayesian Inference in Statistical Analysis, John Wiley and Sons, 1992.

[8] S. Easterbrook *et al*., “Experiences using lightweight formal methods for requirements modeling,” IEEE Transactions on Software Engineering, vol. 24, no. 1, pp. 4–14, Jan. 1998.

[9] M. Neil, “Using process and enterprise modeling with OOBNs to design and automatically generate BBN decision support applications,” *Technical Report 102 v00.012*, Dept. of Computer Science, Queen Mary, University of London, Apr. 2002.

[10] N.E. Fenton *et al*., “Software quality prediction using Bayesian networks,” in *Software Engineering with Computational Intelligence,* T.M. Khoshgoftaar, Ed. Kluwer Academic Publishers, 2003.

[11] L.J. Osterweil *et al*., “Little-JIL/Juliette: A process definition language and interpreter,” in *Proceedings of the 22nd International Conference on Software Engineering*, June 2000, pp. 754–757.

[12] G.J. Pai, J. Bechta-Dugan, K. Lateef, “Analysis of Milestone Readiness Levels During the Software Requirements Development Phase”, Workshop on Software Assessment (WoSA) 2005, Chicago, USA