Automated uncertainty quantification analysis using a system model and data

Saideep Nannapaneni, Sankaran Mahadevan
Department of Civil & Environmental Engineering
Vanderbilt University
Nashville, TN 37235, USA
saideep.nannapaneni@vanderbilt.edu
sankaran.mahadevan@vanderbilt.edu

David Lechevalier
Le2i,
Université de Bourgogne,
BP 47870, 21078 Dijon, France
david.lechevalier@etu.u-bourgogne.fr

Anantha Narayanan
Department of Mechanical Engineering
University of Maryland
College Park, MD 20742, USA
anantha@umd.edu

Sudarsan Rachuri
Systems Integration Division, Engineering Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899, USA
sudarsan.rachuri@nist.gov

***Abstract*— Understanding the sources of, and quantifying the magnitude of, uncertainty can improve decision-making and, thereby, make manufacturing systems more efficient. Achieving this goal requires knowledge in two separate domains: data science and manufacturing. In this paper, we focus on quantifying uncertainty, usually called uncertainty quantification (UQ). More specifically, we propose a methodology to perform UQ automatically using Bayesian networks (BN) constructed from three types of sources: a descriptive system model, physics-based mathematical models, and data. The system model is a high-level model describing the system and its parameters; we develop this model using the Generic Modeling Environment (GME) platform. Physics-based models, which are usually in the form of equations, are assumed to be in a text format. The data is also assumed to be available in a text format.**

The proposed methodology involves creating a meta-model for the Bayesian network using GME and a syntax representation for the conditional probability tables/ distributions. The actual Bayesian network is an instance model of the Bayesian network meta-model. We describe algorithms for automated BN construction and UQ analysis, which are implemented programmatically using the GME platform. We finally demonstrate the proposed techniques for quantifying the uncertainty in two example systems.

Keywords-Bayesian network; meta-model; generic modeling environment; uncertainty quantification; automation;

I. INTRODUCTION

Uncertainty quantification (UQ) involves the estimation of the uncertainty in the output quantity of interest of a system or a model. UQ also requires aggregation of errors and uncertainty from several sources of aleatory and epistemic uncertainty [1]. Bayesian networks (BNs) [2] have become a popular approach to perform uncertainty quantification. They are being used in several fields such as information retrieval, data fusion and engineering decision-making [3], safety assessment of software-based systems [4], civil infrastructure networks [7] and manufacturing systems [8]. The popularity of BNs is increasing for two reasons. First, they allow the integration of various types of uncertainty that combine in different ways [9, 10]. Second, they offer a systematic approach for uncertainty aggregation and management by fusing heterogeneous information available in multiple formats (numerical as well as text) from multiple sources.

BNs rely on an accurate, custom-built model of the domain. The Generic Modeling Environment (GME) [11] is a tool for creating high-level descriptive models of objects in various application domains. A domain is specified in GME by constructing a unique meta-model, which describes the various objects, properties, and relationships in that domain. The tool can be then used to build models of real-world objects in that domain.

Lechevalier et al., [12] propose the idea of using domain-specific modeling languages and tools to bridge the gap between the modeling and analytics procedures in the manufacturing domain. The key idea is to obtain analytical models from the domain-specific manufacturing system models, also called instance models.

This paper proposes a method to automatically generate a BN from instance models, physics-based models and available data on the system. Our method to automate the construction of a BN and UQ analysis is based on a Bayesian network meta-model, which is explained in Section 3.

The remainder of this paper is organized as follows. Section II provides an introduction to BNs, techniques for BN construction, BN learning algorithms and meta-modeling. Section III describes the proposed methodology and algorithms for BN construction and UQ analysis. Two examples – a mathematical example and an injection molding example are used to demonstrate the proposed methodologies in Section IV. Conclusions are provided in Section V.

II. BACKGROUND

A. Bayesian networks

A Bayesian network is a probabilistic, acyclic, graphical model, consisting of nodes and directed arcs. The nodes represent the variables in the system. The arcs represent

dependency relationships between variables, which are quantified by conditional probability distributions. Nodes that have a directed edge pointing towards a node *n* are called the 'parent nodes' of node *n*. Mathematically, a Bayesian network is the joint probability distribution of a set of variables $\boldsymbol{X} = (X_1, X_2 \dots X_n)$ represented as

$$Pr_B(\boldsymbol{X}) = \prod_{i=1}^{n} Pr_B(X_i|\Pi_{X_i}) \quad (1)$$

where Π_{X_i} represents the set of parent nodes of X_i and $Pr_B(X_i|\Pi_{X_i})$ represents the conditional probability distribution of X_i, given its parent nodes. If X_i has no parent nodes, then $Pr_B(X_i|\Pi_{X_i})$ represents the marginal probability distribution of X_i.

B. Techniques for Bayesian network construction

The techniques for constructing a Bayesian Network can be broadly divided into three types: physics-based, data-driven, and hybrid approaches. The physics-based approach relies on the availability of a set of mathematical equations that represent all the relevant relationships between the system variables. The data-driven approach assumes that no such equations exist and that ample data about the system is available. This data is provided as an input to BN learning algorithms that can "learn" the structure of the network.

In some cases, mathematical equations are available only for some segments of the system and data is available for the other segments. In such a scenario, a hybrid approach is taken, where physics-based equations might be used to model some dependencies in the BN whereas the remaining dependencies are learned from the available data. The overall Bayesian network is constructed in two stages – (1) a partial Bayesian network is obtained using the available physics-based models, and (2) the Bayesian network constructed in step 1 is used as a prior, denoted as $\Pr(G)$ in Equation (4), for learning the remaining dependencies using the Bayesian network learning algorithms.

C. Bayesian network learning algorithms

The goal in using learning algorithms is to identify the Bayesian network structure that best describes the available data. The task of learning consists of two steps: structure learning and parameter learning. Structure learning involves finding a graphical structure that best represents the dependency between nodes based on available data. Parameter learning involves quantifying the dependencies among several nodes by estimating the parameters of the associated conditional probability distributions [13].

The structure learning algorithms can be broadly divided into three categories: constraint-based, score-based, and hybrid. Constraint-based algorithms use conditional independence tests to learn the structure of the BN. Mutual information test is a commonly used conditional independence test. The expressions for mutual information ($I_{X,Y}$) in the case of discrete variables and continuous variables are given in Equations (2) and (3), respectively, as

$$I_{X,Y} = \sum_{Y}\sum_{X} p(x,y)\ log\left(\frac{p(x,y)}{p(x)p(y)}\right) \quad (2)$$

$$I_{X,Y} = \int_{Y}\int_{X} p(x,y)\ log\left(\frac{p(x,y)}{p(x)p(y)}\right) \quad (3)$$

where $p(x,y)$ represent the joint probability distribution of X and Y, $p(x)$ and $p(y)$ represent the marginal distributions of X and Y, respectively.

In score-based learning, every possible BN structure is assigned a network score based on 1) the goodness-of-fit for available data and 2) a set of heuristic optimization techniques that obtain the structure that maximizes the score. A commonly used scoring metric is Bayesian Dirichlet equivalence (BDe). The BDe based scoring criterion maximizes the posterior probability of a network-structure given data and is given as

$$\begin{aligned} \Pr(G|D) &\propto \Pr(D|G)\Pr(G) \\ &= \Pr(G)\int \Pr(D|G,\Theta)\Pr(\Theta|G)\,d\Theta \end{aligned} \quad (4)$$

where G, D represent the structure of the Bayesian network and available data, respectively.

Hybrid algorithms employ both conditional independence tests and network scores for learning the BN structure. The conditional independence tests are used to reduce the space of possible BN structures whereas score-based methods are used to obtain the optimal structure among them.

Parameter-learning algorithms estimate the parameters of the various conditional probability distributions from available data using the maximum-likelihood approach.

D. Meta-modeling

Modeling tools have become essential to the design and analysis of complex systems. Using such tools involves a process and a paradigm. The modeling process conforms to a set of rules that minimizes errors and facilitates the presentation and communication of models. The modeling paradigm or the modeling language, such as GME, contains all syntactic, semantic and presentation information regarding a domain, and represents the rules that govern the construction of models.

In recent years, the notion of meta-modeling has been added to process and paradigm. The outcome of the meta-modeling task is a meta-model that encodes all the concepts and rules of the modeling paradigm. GME, the modeling language we used, offers a meta-modeling language called MetaGME, which is based on Unified Modeling Language (UML) Class Diagrams [14], to create domain-specific meta-models. The meta-models described in this paper were built using MetaGME.

In related work, Nannapaneni et al., [15] present a technique for using domain-specific models for analytics. This technique uses the system model to extract the reliability block diagram using the concepts of functional decomposition and function-component association for reliability analysis. Aguila and Sagrado [16] developed a Bayesian network meta-model that they referred to as BayNet and which has different modules for representation of Bayesian network structure (BayNet structure) and inference (BayNet Reasoning). BayNet allows for modeling of discrete variables only, whereas this paper seeks to develop a generalized methodology to handle discrete, continuous and functional nodes.

III. METHODOLOGY

Our method for using Bayesian networks for uncertainty quantification can be divided into two steps 1) Automated BN construction using available models and data, and 2) UQ analysis using the constructed BN.

A. Automated BN construction using available models and data

As stated in Section II, a BN can be constructed using mathematical models or data or a combination of both. We consider two construction cases in this paper: one using physics-based models and one using data. It is straightforward to construct a BN manually when models are available. This paper, however, focuses on the automated generation of a BN.

The variables required for construction of a BN can be obtained from the manufacturing system description. We incorporate this description into a domain-specific model in GME, which, as noted, is as an instance of the corresponding meta-model developed using MetaGME. The details of constructing a generic meta-model for manufacturing systems are not discussed here, but an illustrative example is provided in Section IV. The system variables in the descriptive system model are used as a basis for identifying the nodes and their preliminary ordering for the BN that will be generated. Data associated with the system variables is then used to obtain the BN representing the system.

1)Automated BN construction using physics-based models

Physics-based models are assumed to be available as equations in a text (.txt) file. The models could be present in any random order. The algorithm presented below will order the equations and build a BN from them. An illustrative example is provided in Section IV.

1) Create two lists, x_L and x_R to store the variables to the left and right of the equality sign.
2) Create a dictionary object D with the left hand side (LHS) variables of an equation as the key and the list containing all the right hand side (RHS) variables of that equation as the value.
3) Since a Bayesian network is a layered structure, the variables in the top layer, also called root variables, are given by $x_R - (x_R \cap x_L)$.
4) The second layer comprises all variables that can be defined by a subset of the top-level variables. This can be achieved by selecting the keys whose values are a subset of the first layer variables.
5) Similarly, every other layer consists of variables that can be defined by the variables in the above layers. The procedure specified in step 4, i.e., looking into the dictionary D, is used to select all the variables in the current layer.
6) Step 5 is repeated several times until all the variables in the system are defined.

2) Automated BN construction using available data

When physics-based models are not available, we propose to use the system model and the data associated with the process variables in the system model to construct the BN. Apart from data, the system model may provide qualitative information about the dependencies between several variables, which can be used to improve BN construction from data. The Bayesian network learning algorithm we used is the scored-based method described in Wilczyński and Dojer [17]. The BDe criterion (Section II.C) is used as a scoring metric. A key features of this algorithm is that it can handle both discrete and continuous variables, which are often present in manufacturing scenarios. The continuous variables are handled by discretizing them into two-component, Gaussian-mixture models. A linear, Gaussian conditional probability distribution (CPD) with a constant mean and unknown variance is fit for each continuous node. Note that the mean of the CPD is a linear combination of parent nodes.

The constructed Bayesian network can be validated using model validation techniques [18] such as model-reliability metric, area-metric etc. The available dataset can be divided into a training and a test data; the BN can be constructed with the training test and validated with the test set.

B. Uncertainty quantification using the Bayesian network

The BN-based UQ methodology described in [8] is used in this paper. The important steps are mentioned below for the sake of completeness. UQ analysis using a BN can be divided into three tasks: 1) Construction of a BN (Section II.B), 2) Model calibration, where the unknown parameters are estimated using any observation data, and 3) Forward uncertainty propagation, where the posterior distribution of the output quantity of interest is constructed using posterior distributions of estimated model parameters.

The procedure for performing automated UQ analysis using a BN can be divided into three steps. The first is to transform the BN (constructed using III.A) into an instance model in GME. The second is to implement the above UQ methodology for the BN meta-model on the GME platform.

The third is to apply the UQ methodology to the BN instance model.

Using the instance model of the BN, the UQ analysis methodology, and any new data, uncertainty quantification can be carried out for the BN constructed in Section III.A.1, in an automated manner. Figure 1 shows the proposed methodology for automated UQ analysis.

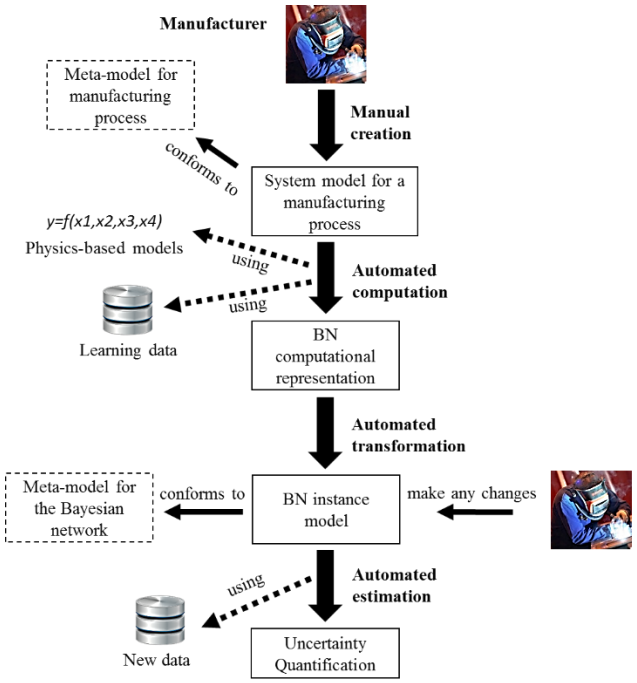


Figure 1. Methodology for automated UQ analysis

1) Bayesian network meta-model

Figure 2 shows the BN meta-model created using GME.

Description of the meta-model:

In the BN meta-model (Figure 2), 'BayesianNetwork' represents the root component, and 'Node' represents any node in the Bayesian network. There can be three types of nodes in the Bayesian network: 'DiscreteNode', 'ContinuousNode', and 'FunctionalNode'. A discrete node represents a variable that has a finite number of states. Similarly, a continuous node represents a variable that is continuous. A functional node represents a variable, which can be known deterministically when the values of its parent nodes are known. Functional nodes are used to represent any functional relationships that may be available between nodes in a Bayesian network.

A 'Node' is specified by an inheritance relationship denoted by the triangle icon between the Node and its subtypes. In a BN, one node is connected to another forming an edge; this is represented in the meta-model using the 'src' (source) and 'dst' (destination) tags at the 'Node' component and by the 'Edge' component, which is a connection type of component.

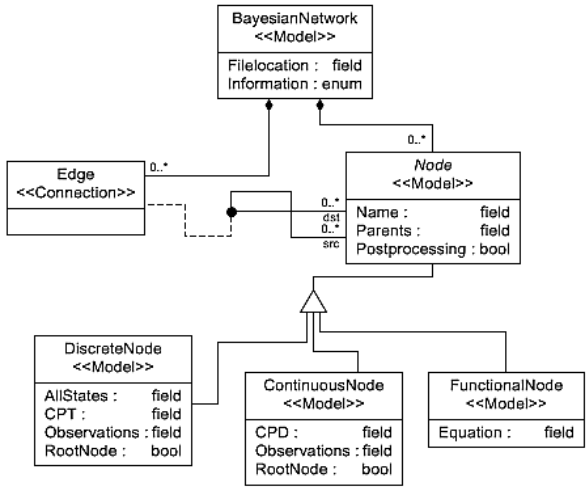


Figure 2. Meta-model for the Bayesian network

The next step in defining the meta-model is to provide a set of required attributes for each of the classes. The most important attributes in a BN are the node names and the CPDs/CPTs. Additional attributes that are required for our UQ analysis are described below. The 'BayesianNetwork' component is associated with two attributes – 'Filelocation' and 'Information'. The 'Information' attribute is an 'enumeration' type and can take only two values – 'Models' and 'Data'. The 'Filelocation' attribute refers to the location of the file that contains either models or data. The attributes that are common to all three types of nodes such as 'Name', 'Parents', and 'Postprocessing' are associated with the 'Node' component. All the parent nodes associated with a node are provided in 'Parents' attribute. 'Postprocessing' is a Boolean variable, that specifies whether or not the variable requires post processing analysis (posterior distribution analysis). Apart from the common attributes, each type of node has a different set of attributes.

Additional attributes for a discrete node include 'RootNode', 'CPT', 'AllStates' and 'Observations'. A node with no incoming edges (i.e., with no parent nodes) is called a 'root node'. The Boolean attribute 'RootNode' is provided to identify root nodes. Only Continuous and Discrete nodes in a BN can be root nodes. All the possible finite states of the discrete variables are provided in the 'AllStates' attribute. Any new observational data is provided with the 'Observations' attribute. The conditional probability table for the discrete variable or marginal probability table (for root nodes) is defined in the 'CPT' attribute. For illustration, Table I defines a discrete parent node 'A' with three possible states 'A_1','A_2','A_3' and marginal probabilities of 0.1,0.6,0.3 respectively. Other attributes such as 'Observations' and 'Postprocessing' are not mentioned below because the goal here is to demonstrate the definition of a CPT.

TABLE I. REPRESENTATION OF A ROOT DISCRETE NODE

Attribute	Value

Name	A
RootNode	True
Parents	
AllStates	A_1, A_2, A_3
CPT	0.1,0.6,0.3

Note that, when defining the marginal probabilities, the order of probabilities should be the same as the order of states, which is defined in 'AllStates' attribute. Since A is a root node, it has no associated parent nodes; therefore, the value corresponding to Parents in Table I is empty. Next, consider a discrete node with discrete parents. Let A and B be the two parent nodes each with two states, A = {A_1, A_2} and B = {B_1, B_2}. Let C represent the child node with two states, C = {C_1, C_2}. The conditional probability table is given in Table II.

TABLE II. CPT OF A DISCRETE NODE WITH DISCRETE PARENTS

C | A, B	**A = A_1, B = B_1**	**A = A_1, B = B_2**	**A = A_2, B = B_1**	**A = A_2, B = B_2**
C = C_1	0.6	0.7	0.2	0.4
C = C_2	0.4	0.3	0.8	0.6

The case when the discrete child node has continuous parent nodes or a combination of continuous and discrete parent nodes is discussed below. The key ideas in dealing with continuous parent nodes involve discretizing their ranges and defining a conditional probability for the child node in each of the ranges. Let A, B represent a discrete and a continuous parent node of a discrete child node C. Assume A has two states, A = {A_1, A_2} and B follows a uniform distribution between 10 and 20. Let the range of B be divided into two uniform intervals; therefore, B can be considered as a discrete variable. The corresponding conditional probability table is given as shown in Table III

TABLE III. CPT OF A DISCRETE NODE WITH DISCRETE AND CONTINUOUS PARENTS

C | A, B	**A = A_1, B = [10,15]**	**A = A_1, B = (15,20]**	**A = A_2, B = [10,15]**	**A = A_2, B = (15,20]**
C = C_1	0.6	0.7	0.2	0.4
C = C_2	0.4	0.3	0.8	0.6

In Table III, the squared brackets also include the equality whereas the parentheses do not. If B = [10, 15], then 10 <= B <= 15 whereas B = (15, 20] represents 15 < B <= 20. The representation of C is shown in Table IV.

TABLE IV. REPRESENTATION OF A CHILD DISCRETE NODE

Attribute	**Both parents are discrete**	**One discrete and one continuous**
Name	C	C
Root Node	False	False
Parents	A,B	A,B
AllStates	C_1,C_2	C_1, C_2
CPT	A_1,B_1 : 0.6,0.4; A_1,B_2 : 0.7,0.3;A_2,B_1 : 0.2,0.8; A_2,B_2 : 0.4,0.6	A_1, [10,15] : 0.6,0.4; A_1, (15,20] : 0.7,0.3; A_2, [10,15] : 0.2,0.8; A_2, (15,20] : 0.4,0.6

Consider the case when B is represented using a Normal distribution and divided into two disjoint intervals B <= 15 and B > 15 – represented as (..15] and (15..) respectively. The same representation can be extended to the case when all the parent nodes are continuous. Each continuous node is discretized and treated as a discrete variable; a similar procedure can be followed as for the case of a discrete node with all continuous parents.

The attributes for the continuous node include 'RootNode', 'CPD' and 'Observations'. The definitions for 'RootNode' and 'Observations' are identical to their discrete counterparts. 'CPD' represents the conditional probability distribution or marginal probability distribution (for root nodes). For illustration, consider a normally distributed variable 'A' with parameters (mean, standard deviation) 10 and 1. Attributes such as 'Postprocessing' and 'Observations' are not mentioned below. Representation of 'A' is given in Table V.

TABLE V. REPRESENTATION OF A ROOT CONTINUOUS NODE

Attribute	**Value**
Name	A
RootNode	True
Parents	
CPD	Normal(10,1)

Next, different combinations of parent nodes, A and B, for a continuous child node C are considered and the corresponding representations are given in Table VI.

TABLE VI. REPRESENTATION OF A CHILD CONTINUOUS NODE

Attribute	**Both parents are discrete**	**One discrete and one continuous**	**Both parents are continuous**
Name	C	C	C
RootNode	False	False	False
Parents	A(A_1,A_2), B(B_1,B_2)	A (A_1,A_2), B(continuous)	A,B (Both continuous)
CPD	A_1,B_1 : Normal(5,1); A_1,B_2 : Uniform(10,14); A_2,B_1 : Normal(10,2); A_2,B_2 : Uniform(12,17)	A_1:Normal(2*B,1);A_2: Uniform(B-2, B+2)	Normal(A+2*B, 1)

After discrete and continuous nodes, functional nodes are considered. As stated earlier, functional nodes are deterministically known when conditioned on all the parent nodes, either discrete or continuous. Functional nodes have only one additional attribute called 'Equation'. The expression connecting the parent nodes to the child node is given here. If A and B represent the continuous parent nodes, for a functional node C, and if C = A + 2*B, then the variable is represented as shown in Table VII.

TABLE VII. REPRESENTATION OF A FUNCTIONAL NODE WITH CONTINUOUS PARENTS

Attribute	**Value**
Name	C

Parents	A,B
Equation	A + 2*B

When a child node has a functional node as a parent node, the functional node can be treated in the same manner as a continuous node for CPD/CPT representation. If a functional node has finite states, it can be modeled as a discrete node where one particular state has a conditional probability of 1 and all the other states have zero as their conditional probabilities. A functional node may also be modeled as a continuous node with zero variance. However, some Bayesian network packages do not allow modeling a variable with zero variance. In such cases, a functional node can be defined as continuous node with a very small variance. To avoid confusion, we have chosen to model functional nodes explicitly.

2) Algorithms for automated UQ analysis

Two algorithms are implemented for the BN meta-model: 1) to construct the instance model of the BN, and 2) to perform UQ analysis.

The first algorithm constructs the BN using available physics-based models or data. In case physics-based models are available, the algorithm presented in Section III.A.1 is used to construct the BN. If data is available, the BN is learnt using BNFinder package in Python, as stated in Section III.A.2. The BN is constructed in Python and stored as a JSON representation. The JSON file is then automatically converted into a GME instance model. The user can make any changes to the BN instance model (such as adding observations or selecting variables for post processing, deleting some nodes), directly using the GME interface. When all the required changes are made to the BN instance model, the second algorithm is executed for automated BN analysis.

The UQ analysis is carried out using PyMC [19] package in Python. We have implemented a program that reads the BN instance model and creates a JSON file to represent the data corresponding to all the nodes in the BN. A Python script then takes the JSON file as input and creates a BN model to carry out UQ analysis. Markov Chain Monte Carlo (MCMC) sampling, using Metropolis-Hastings algorithm [20], is carried out to construct the posterior probability distributions of several variables in the BN.

IV. ILLUSTRATIVE EXAMPLES

1) UQ analysis using data

A mathematical example is used to demonstrate the proposed methodologies for automated UQ analysis using data. In this example, all possible combinations of discrete and continuous nodes are available. Figure 3 shows a simple meta-model for manufacturing processes, created using GME. An instance of this meta-model (shown in Figure 4) is used in this example. The model consists of three individual processes namely 'Process1', 'Process2' and 'Process3'.

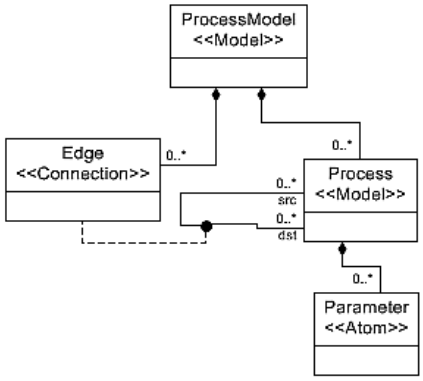


Figure 3. Mathematical example: Meta-model

Figure 4. Mathematical example: Instance-model

Using the conditional probability distributions in Table VIII, we created a synthetic dataset of 500 samples to use as inputs to the BN learning algorithm. We can see from the model that 'Process1', 'Process2', and 'Process3' occur in that sequential order. Assume that all the variables measured for a process are independent of each other. Therefore, from the model, we can deduce that 'D1','D2' must be the root nodes in the Bayesian network. The variables 'C1', 'C2', and 'D3' will be in the second layer of the BN and each of them could have 'D1', 'D2', or both as parent nodes. Similarly, 'C3', 'D4', and 'C4' will be in the third layer and each of them can be expressed using a subset of the variables associated with Process2 as parent nodes. Adding such qualitative information makes the learning process faster and more accurate. The statistical results are given in Table IX.

The topology of the learnt BN matches that of the true BN; this can be observed by comparing the parent nodes of each child node from the true and learnt CPDs (Tables VIII and IX), although there are minor differences in the conditional probability distributions. The differences can be attributed to the assumptions made in the learning process (linear Gaussian CPD) and to the amount of data available for BN learning. As the amount of data increases, the accuracy of the conditional probability distributions can be improved. As stated in Section III.B.2, two model interpreters are created to perform automated UQ analysis. Using the first interpreter, a GME instance model of the constructed BN is created (Figure 5). In the instance model, observations are added for nodes 'D4','C1', and 'C4'. Using the observation data, the posterior distributions of root nodes'D1' and'D2' are obtained. The statistics of the prior and posterior distributions are provided in Table X.

TABLE X. MATHEMATICAL EXAMPLE: PRIOR AND POSTERIOR PROBABILITIES OF CALIBRATION PARAMETERS

Variable	True value	Prior probability	Posterior probability
D1	0	0.3,0.7	1,0
D2	1	0.6,0.3,0.1	0.38,0.62,0

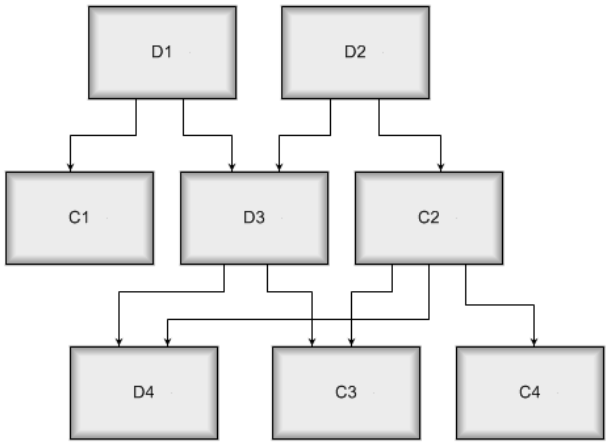


Figure 5. Mathematical example: BN instance model

2) UQ analysis using physics-based models

An injection molding process is used to demonstrate automated BN construction and UQ analysis using physics-based models. The injection molding process can be considered as a combination of three sub-processes: melting of the polymer, injection into the mold, and cooling to form the part. Each sub-process is associated with a set of parameters (input variables as well as model parameters). The goal is to estimate the energy consumption per part in the injection molding process. Physics-based models are assumed to be available for energy consumption (Equations 5 to 11).

$$V_{shot} = V_{part}(1 + \epsilon + \Delta) \tag{5}$$

$$P_{melt} = \rho \times Q_{avg} \times C_p \times (T_{inj} - T_{pol}) + \rho \times Q_{avg} \times H_f \tag{6}$$

$$E_{melt} = \frac{P_{melt} \times V_{shot}}{2 \times Q_{avg}} \tag{7}$$

$$E_{inj} = P_{inj} \times V_{part} \tag{8}$$

$$E_{cool} = \frac{\rho \times V_{part} \times [C_p \times (T_{inj} - T_{ej})]}{COP} \tag{9}$$

$$E_{reset} = 0.25(E_{inj} + E_{cool} + E_{melt}) \tag{10}$$

$$E_{part} = \left[\left(\frac{0.75 \times E_{melt} + E_{inj}}{\eta_{inj}} + \frac{E_{reset}}{\eta_{reset}} + \frac{E_{cool}}{\eta_{cool}} + \frac{0.25 \times E_{melt}}{\eta_{heater}}\right) \times \frac{(1 + \epsilon + \Delta)}{\eta_{machine}} + P_b \times t_{cycle}\right] \tag{11}$$

In the above set of equations, $V_{shot}, V_{part}, \epsilon, \Delta$ represent the volume of a shot, volume of a part, shrinkage and buffer respectively. $P_{melt}, \rho, Q_{avg}, C_p, T_{inj}, T_{pol}, T_{ej}, H_f$ represent the power consumption for melting, density of polymer, average flow rate, heat capacity, injection temperature, polymer temperature, ejection temperature, heat of fusion respectively. $E_{melt}, E_{inj}, E_{cool}, E_{part}$ represent energy consumption for melting, injection, cooling and total energy for a part respectively. More details are available in [21].

A BN instance model (Figure 6) is created by executing the interpreter for BN construction, as stated in Section III. For illustration, all the efficiency coefficients (η, COP) are assumed to be 0.7 and P_b, which is the power consumed by the basic units, is assumed to be zero. The statistics of variables are provided in Table XI.

TABLE XI. INJECTION MOLDING: VARIABLES AND THEIR VALUES

Parameter	Values	Parameter	Values
$V_{part}(m^3)$	0.002048	$H_f(kJ/kg)$	240
ϵ	0.0185	$\rho(kg/m^3)$	U(960,990)
Δ	0.01	$T_{inj}(^oC)$	215
$T_{ej}\ (^oC)$	U(45,60)	$C_p(J/(kgK))$	2260
$T_{pol}\ (^oC)$	49	$P_{inj}(MPa)$	93

All the parameters with constant values can be removed from the BN instance model. Observation data is assumed to be available on the total energy consumption (E_{part}), and ejection temperature (T_{ej}). The parameters to be calibrated are the density of the polymer (true value = 985), and ejection temperature (true value = 55). The prior distributions associated with the calibration parameters are given in Table XI. The measurement errors associated with temperature measurements and energy are assumed to be normal distributions with zero mean and standard deviations of 2^oC and 100 kJ respectively. A normal distribution is typically assumed for measurement errors because it is symmetric about zero; therefore positive and negative errors are equally probable and small errors are more likely to occur than large errors in a controlled experiment [22]. The modified BN instance model (BN instance model after removing all constants) is then used to carry out UQ analysis. The UQ analysis can be carried out by executing the second model interpreter. The prior and posterior distributions of the calibration parameters are provided in Figure 7.

V. CONCLUSION

This paper proposed a methodology for automated Bayesian network construction and uncertainty quantification analysis. The BN is constructed by fusing the information from available system models, physics-based models, and collected data. The system model is assumed to be available as a domain-specific model in the GME platform. Physics-based models are assumed to be available as equations in a text format.

A meta-model for the BN is developed along with a syntactic representation of the conditional probability distribution/tables in GME. The meta-model is created such that the BN can have discrete, continuous, and functional nodes. On the meta-model, two model interpreters (also called model translators) are written to construct the BN and to perform uncertainty quantification analysis. An algorithm is presented to generate the BN from a set of physics-based models. In the absence of physics-based models, the BN is constructed using BN learning algorithms.

The constructed BN is represented as an instance model of the BN meta-model. The instance model can be modified to include any further information about the BN such as observation data and post-processing information. Using the instance model, the second program constructs a Bayesian network for UQ analysis using PyMC package in Python. The proposed method is illustrated using two examples – an injection molding process and a mathematical example.

Automated UQ analysis using physics-based models is demonstrated for injection molding and a mathematical example is used to demonstrate UQ analysis using data. The injection molding example consists of only functional nodes and continuous nodes, whereas the mathematical example is created such that it has all combinations of discrete and continuous variables.

The goal of this work is to assist manufacturers in performing uncertainty analysis using automated tools, without requiring expertise in UQ methods and UQ-specific tools. This will help manufacturers make better use of their data analytics capabilities by allowing them to give proper consideration to uncertainty.

Future work is needed to develop algorithms for automation of analysis such as sensitivity analysis, verification and validation. Also, algorithms for the construction of dynamic Bayesian networks (for tracking system evolution over time), and diagnostic and prognostic analysis need to be investigated.

ACKNOWLEDGMENT

The research reported in this paper was funded in part by the National Institute of Standards and Technology under Cooperative Agreements No. 70NANB14H036 and No. 70NANB13H159, and NIST's Foreign Guest Researcher Program.

REFERENCES

[1] H.R.Bae, R.V. Grandhi, and R.A.Canfield, "Epistemic uncertainty quantification techniques including evidence theory for large-scale structures." Computers & Structures, 82(13), 1101-1112. 2004.

[2] F.V.Jensen, "An Introduction to Bayesian Networks>" Springer-Verlag, 1996.

[3] G.Dahll, "Combining disparate sources of information in the safety assessment of software-based systems," Nuclear Engineering and Design. 2000; 195: 307-319.

[4] L.M. de Campos, J.M. Fernández-Luna and J. F. Huete, "Bayesian networks and information retrieval: an introduction to the special issue." Information Processing & Management (Elsevier) 40 (5): 727–733, 2004.

[5] N.Friedman, M. Linial, I. Nachman, D. Pe'er, "Using Bayesian Networks to Analyze Expression Data." Journal of Computational Biology, (3/4): 601–620, 2004.

[6] X.Jiang, R.E.Neapolitan, M.M.Barmada, and S.Visweswaran, "Learning Genetic Epitasis using Bayesian Network Scoring Criteria," BMC Bioinformatics 12: 89, 2011.

[7] M.Bensi, and A. Der Kiureghian, "Seismic hazard modeling by Bayesian network and application to a high-speed rail system," Proceedings of International Symposium on Reliability Engineering and Risk Management, (Ed: J. Li), Tongji University Press, Shanghai, China, September 2010.

[8] S.Nannapaneni and S.Mahadevan, "Uncertainty Quantification in Performance Evaluation of Manufacturing Processes," IEEE International Conference on BigData (Big Data), IEEE, 2014.

[9] S.Sankararaman, Y.Ling and S.Mahadevan, "Uncertainty quantification and model validation of fatigue crack growth prediction," Engineering Fracture Mechanics, 78(7), 1487-1504, 2011.

[10] B.Liang and S.Mahadevan, "Error and uncertainty quantification and sensitivity analysis in mechanics computational models," International Journal for Uncertainty Quantification, 1(2), 2011.

[11] A. Lédeczi, A. Bakay, M. Maroti, P. Völgyesi, G. Nordstrom, J. Sprinkle, and G. Karsai. "Composing domain-specific design environments." Computer 34, no. 11: 44-51, 2001.

[12] D.Lechevalier, A.Narayanan, and S.Rachuri. "Towards a domain-specific framework for predictive analytics in manufacturing." IEEE International Conference on Big Data (Big Data), IEEE, 2014.

[13] M.Scutari, "Bayesian Network Constraint-Based Structure Learning Algorithms: Parallel and Optimised Implementations in the bnlearn R Package," Journal of Statistical Software, June 2014.

[14] The Unified Modeling Language. http://www.omg.org/spec/UML/2.5/ (Accessed 9/2015)

[15] S.Nannapaneni, A.Dubey, S.Abdelwahed, S.Mahadevan and S.Neema, "A Model-Based Approach for Reliability Assessment in Component-Based Systems," Proceedings of the Annual Conference of the Prognostics and Health Management Society, 2014

[16] I.M. del Aguila, and J. del Sagrado. "Metamodeling of Bayesian networks for decision-support systems development." Knowledge Engineering and Software Engineering (KESE8): 8, August 2012.

[17] B.Wilczyński and N.Dojer, "BNFinder: exact and efficient method for learning Bayesian networks." Bioinformatics, 25(2), 286-287, 2009.

[18] Y.Ling and S.Mahadevan, "Quantitative model validation techniques: New insights", Reliability Engineering & Systems Safety, 111, 217-231, 2013

[19] A.Patil, D.Huard, and C.J.Fonnesbeck. "PyMC: Bayesian stochastic modelling in Python." Journal of statistical software 35.4 2010: 1.

[20] S.Chib, and E.Greenberg, "Understanding the metropolis-hastings algorithm." The american statistician, 1995, *49*(4), 327-335.

[21] J.Madan, M.Mani, and K.W.Lyons,"Characterizing energy consumption of the injection molding process," Proceedings in ASME 2013 International Manufacturing Science and Engineering Conference collocated with the 41st North American Manufacturing Research Conference (pp. V002T04A015-V002T04A015), 2013.

[22] N.C.Barford, "Experimental measurements: precision, error and truth." Chichester: Wiley, 1985, 2nd ed.

TABLE VIII. MATHEMATICAL EXAMPLE: CPD/CPT OF VARIABLES USED FOR SYNTHETIC DATASET GENERATION

Variable	CPT/CPD					
D1	0.3,0.7					
D2	0.6,0.3,0.1					
C1 | D1	Normal(10 + 4*D1,2)					
C2 | D2	Normal(6+2*$D2^2$,1)					
C4| C2	Normal(0.1*$C2^2$ + 0.6*C2 + 1, 2)					
C3 | C2, D3	D3 = 0, Normal(0.15*$C2^2$,2)			D3 = 1, Normal(2*C2,1)		
D3|D1,D2	(0,0)	(0,1)	(0,2)	(1,0)	(1,1)	(1,2)
	0,1,0.9	0.3,0.7	0.4,0.6	0.6,0.4	0.8,0.2	0.9,0.1
D4|D3,C2	(0, C2<9)	(0,9<C2<11)	(0, C2>11)	(1, C2<9)	(1,9<C2<11)	(1, C2>11)
	0.4,0.6	0.3,0.7	0.6,0.4	0.7,0.3	0.8,0.2	0.3,0.7

TABLE IX. MATHEMATICAL EXAMPLE: CPD/CPT OF VARIABLES IN THE LEARNT BN

Variable	CPT/CPD					
D1	0.32,0.68					
D2	0.6,0.31,0.09					
C1 | D1	D1=0, Normal(9.68,1.87)			D1=1, Normal(14.04,1.93)		
C2 | D2	D2=0, Normal(6.03,1)		D2=1, Normal(7.97,1)		D2=2, Normal(14,0.84)	
C4| C2	Normal(2.46*C2 -6.63,2.28)					
C3 | C2, D3	D3 = 0, Normal(5.83*C2-25.2,9.58)			D3 = 1, Normal(2*C2+0.12,1.37)		
D3|D1,D2	(0,0)	(0,1)	(0,2)	(1,0)	(1,1)	(1,2)
	0.09,0.91	0.33,0.67	0.25,0.75	0.56,0.44	0.79,0.21	0.88,0.12
D4|D3,C2	(0, C2<10.3)		(0, C2>10.3)	(1, C2<10.3)		(1, C2>10.3)
	0.35,0.65		0.67,0.33	0.72,0.28		0.31,0.69

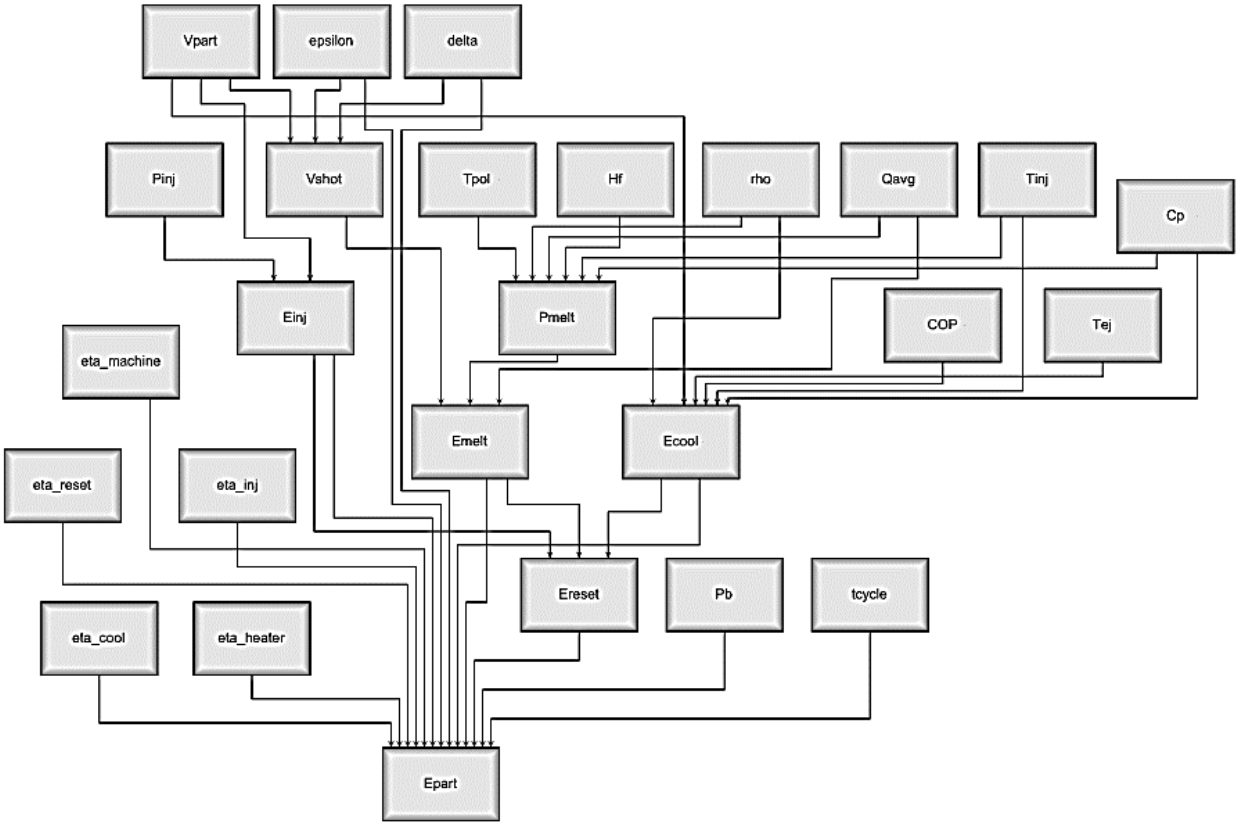
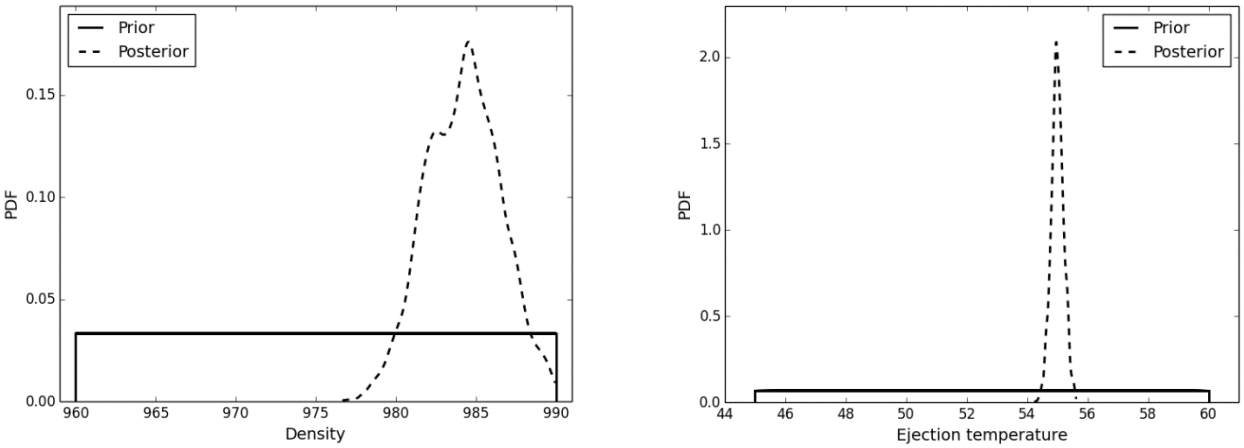


Figure 6. Injection molding example: BN instance model using physics-based models



(a) (b)

Figure 7. Injection molding example: Prior and posterior distributions of (a) Density and (b) Ejection temperature