Verification and Validation of System Health Management Models using Parametric Testing

Erik Reed[*] Johann Schumann[†] Ole J. Mengshoel[‡]

Abstract

System Health Management (SHM) systems have found their way into many safety-critical aerospace and industrial applications. A SHM system processes readings from sensors throughout the system and uses a Health Management (HM) model to detect and identify potential faults (diagnosis) and to predict possible failures in the near future (prognosis). It is essential that a SHM system, which monitors a safety-critical component, must be at least as reliable and safe as the component itself—false alarms or missed adverse events can potentially result in catastrophic failures. The SHM system including the HM model, a piece of software, must therefore undergo rigorous Verification and Validation (V&V).

In this paper, we will describe an advanced technique for the analysis and V&V of Health Management models. Although our technique is generally applicable, we investigate in this paper HM models in the form of Bayesian networks (BNs). BNs are a powerful modeling paradigm to express notions of cause and effect, probability, and reliability. A BN model typically contains many parameters (e.g., thresholds for discretization and conditional probability tables); they need to be set carefully for reliable and accurate HM reasoning. We are investigating the use of Parametric Testing (PT), which uses a combination of n-factor and Monte Carlo methods, to exercise our HM model with variations of perturbed parameters. Multivariate clustering on the analysis is used to automatically find structure in the data set and to support visualization. Our approach can yield valuable insights regarding the sensitivity of parameters and helps to detect safety margins and boundaries.

As a case study we use HM models from the NASA Advanced Diagnostics and Prognostics Testbed (ADAPT), which is a realistic hardware setup for a distributed power system as found in spacecraft or aircraft.

I. Introduction

System Health Management (SHM) systems are ubiquitious in many safety-critical aerospace and industrial applications, including major components of aircraft (e.g., jet engines, hydraulics, or electric power systems). During operation of the monitored system, the system processes readings from sensors throughout the system and use a Health Management (HM) model to detect and identify faults (diagnosis). Often such systems can trigger recovery activities. Consequently these systems are often referred to as FDDR (Fault Detection, Diagnosis, and Recovery). SHM systems can also be used to predict failures in the near future (prognosis). While many different approaches and tools for FDDR and SHM exist,[1–3] this paper focuses on techniques and tools using Bayesian networks.

Regardless of the chosen approach, an SHM system, which has to monitor a safety-critical component, must be at least as reliable and safe as the component itself—false alarms or missed adverse events are not acceptable. False alarms, which occur when the SHM system signals a failure but the monitored system works flawlessly, are usually a nuisance (e.g., the notorious "Check Engine" light in your car). In safety-critical systems, however, missed alarms can pose a severe safety threat. To ensure accuracy and efficacy, the SHM system—a piece of software—must undergo rigorous V&V. Since most SHMs use a model-based design

[*]University of Washington, Seattle, WA 98195, ebr3@u.washington.edu
[†]SGT, Inc., NASA Ames, Moffett Field, CA 94035, Johann.M.Schumann@nasa.gov
[‡]Carnegie Mellon University, NASA Ames, Moffett Field, CA 94035-0001, Ole.Mengshoel@sv.cmu.edu

where the HM model is compiled for efficient execution, successful V&V must consist of *model-level V&V* and *code-level V&V* (also called implementation-level V&V).[4] In this paper, we focus on *model analysis*. A detailed and deep analysis of the HM model is an important V&V activity because it can reveal modeling problems, incorrect parameters, or unusual parameter sensitivity—all of which can lead to an incorrect diagnosis or prognosis and consequently false alarms or missed events.

Here we will describe research which has been performed using Health Management models that are modeled using Bayesian networks (BNs), a very powerful modeling paradigm to express notions of cause and effect, probability, and reliability surpassing many traditional (discrete) approaches toward HM like table-driven techniques or fault trees.

The underlying idea is that each failure manifests itself in certain observable patterns even before having a conceivable impact. However, the root causes only reveal themselves if all available information is taken into account and non-trivial reasoning is performed. For example, the cause of an engine vibration can be only be diagnosed as a worn-out bearing if additional information (like low oil pressure) is considered. Figure 2 shows how this extremely simplistic example would be encoded in a Bayesian HM. Nodes attached to sensors ("vibration", "oil pressure") are shaded; "H_Bearing" is a diagnostic node. Edges indicate causal dependencies and reasoning is governed by node probabilities. Conditional probability tables (CPTs), which are attached to each node, describe in exact terms how the probability of the node depends on its local neighbors. For example, the oil pressure is low in 90% of the cases when the bearing is worn. If the bearing is OK, a low oil pressure (due to some other, unmodeled event) only shows up with a probability of $p = 0.05$. In our example, HM reasoning would calculate the probability of the diagnosis given the sensor values $p(bearing = worn|\{vibration, oilpressure\})$.

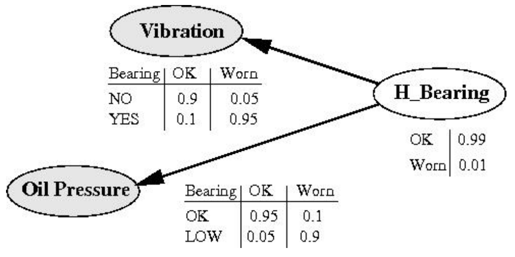


Figure 1. Small example of a Bayesian Health Model

With the given probabilities, vibration alone would not point to a worn bearing (the bearing would report 91.24% OK without any pressure reading). Only a low pressure reading in conjunction with vibration indicates a worn bearing (p=63.3%).

Such networks, which can become very large, may also be compiled into clique trees or arithmetic circuits for efficient embedded execution. Arithmetic circuits[5] push much of the work involved in performing inference to an offline phase, which can then be amortized across many online queries.

A Bayesian HM model contains many parameters (e.g., thresholds for discretization or node probability values), which must be set carefully for reliable and correct HM reasoning. We are using Parametric Testing (PT), which uses a combination of n-factor and Monte Carlo methods to exercise the HM model with variations of perturbed parameters. This technique, which has been successfully applied to various application areas[6,7] avoids the excessive numbers of cases caused by combinatorial exploration while at the same time yielding good coverage of the parameter space. The result of model analysis is a large, high-dimensional data set. In our initial experiments described here, we use multivariate clustering with a tool called AutoBayes to automatically find structure in the data set and to support visualization. PT yields valuable insights on the sensitivity of parameters (in our experiments: threshold parameters) and helps to detect safety margins and boundaries.

Although the results described in this paper are based upon ADAPT HM models, our analysis and V&V approaches can be carried over to other applications and other HM modeling paradigms (e.g., table or rule-based systems).

The rest of the paper is structured as follows: In the next section, we will present a short overview of the ADAPT testbed and its Health Management models. Section III discusses our approach to Parametric Model Analysis, its tool architecture and setup. We will discuss n-factor combinatorial exploration (Section III.B) and results of experiments with parameter perturbation (Section III.C). Section IV is concerned with the clustering analysis of the results as obtained by parametric testing. We will discuss feature selection, the clustering tool (AutoBayes), and results obtained by this unsupervised machine learning technique. Finally, Section V summarizes and discusses future work.

II. Background: The ADAPT Testbed

As a case study, we investigate a real-world electrical power system known as the Advanced Diagnostics and Prognostics Testbed (ADAPT).[8] ADAPT is located NASA Ames Research Center and is representative of electrical power systems found in aerospace vehicles, having capabilities for power storage, power distribution, and power consumption. The schematic for ADAPT is shown in Figure 2. Power is provided by three batteries, distributed over power lines and two voltage inverters (INV1, INV2) to two banks of loads, which consist of fans, lamps, and pumps. Various computer-controlled relays (EY) and circuit breakers route the power. This testbed is instrumented with a large number of sensors, which measure voltage, current, and the position of the relay switches. Table 1 lists all major components, their acronyms, and health modes.

For our SHM system, we consider scenarios taken from the ADAPT testbed. These scenarios consist of nominal runs, in which no faults were injected into the hardware system, faulty runs involving persistent faults, and faulty runs involving intermittent faults.

In this paper we discuss the ProADAPT system,[9–12] a system health management system for for ADAPT. ProADAPT uses a BN to model ADAPT; this network is used to find the root cause(s) for unexpected occurrences or failed components in the testbed, like a lack of power to the loads in ADAPT. Specifically, ProADAPT is tasked to find out which ADAPT component(s) fail and at what time it (they) failed. By looking at time series for sensors in ADAPT (voltage, current, temperature, ...), it can be relatively easy to diagnose a faulty scenario. However, the process quickly becomes non-trivial when there are multiple faults and relatively few sensors available. ADAPT provides a controlled environment in which to inject failures, either through software or hardware, in a repeatable manner. In 2009 and 2010, the international Diagnostics Competition (DXC) was arranged with ADAPT as the real-world testbed. ProADAPT is used in conjunction with the DXC Framework which simulates realistic data collection in a diagnostic and prognostic setting. Each scenario evaluated consisted of over 2200 sample points; the highest sampling frequenzy is 10 Hz. For ProADAPT, data from sensor components is input as evidence in the Bayesian network for each timestep. Both nominal and faulty ADAPT scenarios were used for experimentation. Between DXC-09 and DXC-10, the ProADAPT software had the best results in three out of four competitions in the industrial track[10, 11][a].

Figure 2 shows the ProADAPT Bayesian network used in this case study. This network, which is a typical size for this kind of application, contains 80 sensor nodes, most of which receive discretized sensor values as inputs. Commands and their feedback are also input nodes. The health of all components is modeled using 120 system health nodes. Their most likely states correspond to the best system health estimate for the electrical power network, computed using posterior probability for the BN. As an example, a health node *Health-of-Relay* with states *healthy* (posterior 0.3), *stuck-open* (posterior 0.65) and *stuck-closed* (posterior 0.05) might indicate that the corresponding relay is most likely broken and cannot close it contracts.

III. Parametric Model Analysis

Most sensor values obtained from the ADAPT testbed are continuous (e.g., voltage, temperature). In order to input these values as evidence in our BN, which is inherently discrete, such continuous sensor values must be discretized into node-specific states. This discretization is based on a number of continuous thresholds. For example, a simple voltage sensor node V could have two states: *high* and *low* voltage, delimited by one threshold. If the sensor reads a continuous value v, the state of the sensor node in the Bayesian network is determined to be *high* if $v > \Theta$ for a given threshold Θ, otherwise it is discretized as a *low* voltage.

Due to the continuous nature of many sensors in ADAPT and the use of thresholding, setting reliable and precise thresholds is important to optimize the correctness of Bayesian network diagnosis. A threshold analysis is therefore an important and integral part of SHM V&V; it serves to answer the following questions:

- To what degree does a particular threshold value Θ lead to undesirable effects (missed event or false alarm)?
- Can interactions between several threshold values and different nodes lead to incorrect results?

[a]See `https://c3.ndc.nasa.gov/dashlink/projects/36/` and `https://www.phmsociety.org/competition/dxc/10` for further information and data sets

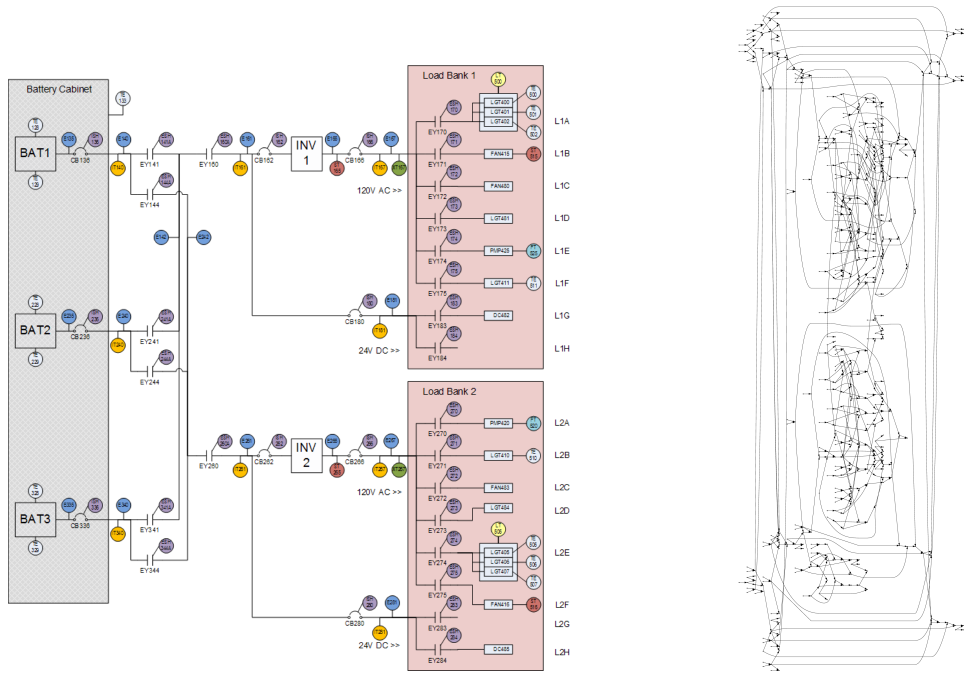


Figure 2. Schematics of ADAPT (left) and ProADAPT Bayesian network (adapt10_v7e, right)

- How critical is the exact value of a threshold Θ? For example, can a change of $\pm 5\%$, or $\pm 10\%$, substantially and detrimentally impact the outcome of diagnostic reasoning?

The answers to the last questions are of particular importance and can be difficult to determine, as sensors often produce noisy or biased data. In addition, the physical characteristics of sensors, as well as of other parts of a system, may also change dramatically as a system ages. In order to minimize the negative impact of these effects on diagnostic performance, one needs to be very careful when setting thresholds, which may also need to be revised as a system ages. In the reminder of this paper we discuss how *parametric testing* techniques can help in handling the challenges associated with these thresholds.

III.A. Testing Infrastructure

The basis for our parametric tests is a given ADAPT Bayesian network health model (see Figure 2) along with one or more scenarios. In our case, scenarios are time sequences of sensor readings. A scenario can be obtained from a healthy system (nominal scenario) or from a faulty system (fault scenario). In a fault scenario, one or more (known) faults have been injected at specific times. Figure 3 illustrates the tool architecture: given the model, the scenario, and test configuration information provided by the analyst, the Parametric Analysis (PMA) tool automatically performs a multitude of tests using variations of thresholds. The test output consists of the posterior probabilities of each of the health nodes over time. Because of the large volume of data generated (some 30 GB per experiment), manual analysis of the raw data is not possible. Consequently, the PMA tool currently produces two kinds of reports. The trace report allows the analyst to study the outputs of selected health nodes $H_1, H_2, \ldots, H_m$ for selected or all test runs. Navigation capabilities built into this autogenerated HTML report supports the analyst.

As discussed before, simultaneous variations of multiple thresholds can have substantial effects. However, multivariate analysis techniques are needed in general to reveal such, often subtle, situations. To tackle these situations, our PMA tool offers clustering-based multivariate analysis and visualization techniques.

In the following, we will discuss the major components of the PMA tool in detail, the generation of the threshold variations, report generation, and the multivariate data analysis. Please note that the PMA tool architecture is not tied in any way to the analysis of thresholds. The tool could easily be configured to perform parametric analysis of conditional probability values in Bayesian networks or of parametric scenarios, for example.

Part	Prefix	Mode (Health/Faulty)	States
Battery	BATT	Healthy	healthy
		Voltage failure or drain	stuckDisabled
Circuit breaker	ISH	Healthy	healthy
		Stuck or failed open	stuckOpen
		Stuck or failed closed	stuckClosed
Inverter	INV	Healthy	healthy
		switched off	stuckOpen
Relay	EY	Healthy	healthy
		Stuck or failed open	stuckOpen
		Stuck or failed closed	stuckClosed
Voltage sensor	EI	Healthy	healthy
		Reading stuck low	readVoltageLo
		Reading stuck high	readVoltageHi
Current sensor	IT	Healthy	healthy
		Reading stuck low	readCurrentLo
		Reading stuck high	readCurrentHi
Position sensor	ISHo	Healthy	healthy
		Reading stuck open	stuckOpen
		Reading stuck closed	stuckClosed

Table 1. ADAPT Signal and sensor naming conventions

III.B. Setup for Inputs and Outputs

Suppose a Bayesian network has m discretized input variables $\boldsymbol{X} = \{X_1, ..., X_m\}$. The cardinality of $X \in \boldsymbol{X}$ is denoted by $\Omega(X)$. A variable X with $\Omega(X) = k$ states has $k-1$ thresholds, $\{t_1, ..., t_{k-1}\}$. For example, a random variable with $\Omega(X) = 3$ states $(-\infty, 0], (0, 7], (7, \infty)$ has two thresholds $t_1 = 0$ and $t_2 = 7$.

In our parametric analysis, we are interested in varying thresholds. For simplicity, suppose that we vary each threshold independently of the other thresholds. Consider a particular threshold $t_i \in \{t_1, ..., t_{k-1}\}$. By slightly increasing or decreasing t_i with a small amount ϵ, we obtain a perturbed threshold $t_i' \in \{t_i - \epsilon, t_i, t_i + \epsilon\}$. Perturbations can be introduced for each threshold $t_i \in \{t_1, ..., t_{k-1}\}$, resulting in

$$|\{t_i - \epsilon, t_i, t_i + \epsilon\}|^{k-1} = 3^{k-1}$$

possible combinations under this simple model. In addition, the simultaneous variation of multiple variables $\boldsymbol{Y}$, where $\boldsymbol{Y} \subseteq \boldsymbol{X}$, is of interest. Suppose $|\boldsymbol{Y}| = \ell$, $\boldsymbol{Y} = \{Y_1, ..., Y_\ell\}$, and $\Omega(Y_1) = k_1$, ..., $\Omega(Y_\ell) = Y_\ell$. We

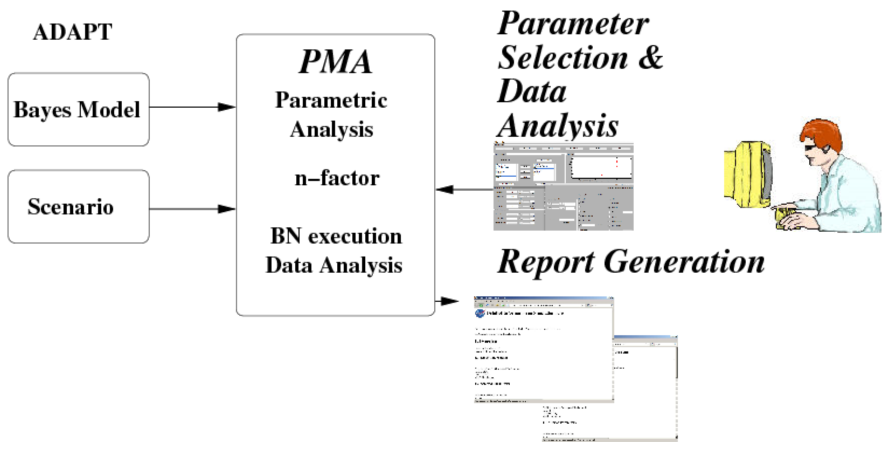


Figure 3. Architecture and workflow of the parametric analysis

then obtain

$$3^{k_1-1} \times \cdots \times 3^{k_\ell-1} \geq 3 \times \cdots \times 3 = 3^\ell,$$

where the conservative lower bound 3^ℓ is due to the fact that for each variable $Y \in \boldsymbol{Y}$, $\Omega(Y) \geq 2$.

Each ADAPT scenario contains $m = 80$ streams of sensor data from ADAPT. Data comes from voltage sensors, temperature sensors, and current sensors; each sensor reading is mapped to a discrete state of a BN node. Making the conservative assumption that all sensors produce continuous values that need to be discretized, a full exploration of the thresholds of the ADAPT BN using $m = \ell$, would require testing 3^{80} combinations, a very large and computationally infeasible number. In addition, each scenario contains a large amount of sensor readings over time, typically 2200+ timesteps, making exhaustive testing infeasible.

Using our parametric analysis approach, we avoid such exhaustive testing. Instead of varying a threshold by a fixed amount ϵ as indicated above, the variation is randomized within a fixed number of bins, each bin corresponding to a range of variation. This allows the perturbation of the thresholds to be represented discretely; the number of possible continuous threshold values can now be represented as a finite number of a combinations through the use of bin assignment. In the experiment reported here, five bins were used (see Figure 4).

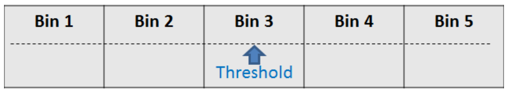


Figure 4. The binning of an individual threshold. Each bin corresponds to the level of variation from the nominal value.

Figure 4 illustrates how variation (from the nominal unmodified threshold) is binned into five bins. The nominal threshold values were perturbed by $\pm 20\%$ in our experiments ($\zeta = .2$, where ζ is the maximum percent to positively or negatively perturb a threshold). This resulted in the following as ranges of variation for the bins:

Bin 1 (large negative changes: -20% to -12%),

Bin 2 (small negative changes: -12% to -4%),

Bin 3 (minimal negative and positive changes: -4% to $+4\%$),

Bin 4 (small positive changes: $+4\%$ to $- + 2\%$), and

Bin 5 (large positive changes: $+12\%$ to $+20\%$).

As the number of bins increase for a given ζ, the range of variation each bin represents decrease. For a particular bin $b_i \in \{1, ..., n\}$ in PT with n bins, a nominal threshold θ and maximum percent to perturb ζ, a perturbed threshold θ' can be acquired via by the following formula:

$$\theta' = ((1/n) * (b_i - 1 + rand[0, 1]) - .5) * 2\zeta\theta + \theta$$

Since there are an average of 3-4 thresholds for each of the 90 ADAPT sensors requiring discretization, there are some 300 thresholds that require testing. An exhaustive combinatorial exploration is not possible due to the complexity of the parameter space, which would require some $5^{300} \approx 10^{209}$ test cases. We therefore used a combination of n-factor combinatorial exploration and Monte Carlo generation to dramatically reduce the number of test cases without losing too much coverage.

This generation method is motivated by the fact that in real systems, most failures are caused by a specific, single value of one input variable or parameter. The case that a fault is triggered by a specific combination of two variables is much less likely. Even more unlikely is a situation where 3 input variables must have specific values in order to trigger the failure; the involvement of 4 or more variables can be, for most purposes, ignored. This observation[13–15] has been used to develop the n-factor combinatorial testcase generation. Here, the generated cases completely cover all combinations of parameters up to n. Table 2 shows the number of test cases generated for various different experiments and settings, demonstrating the dramatically smaller number of test cases generated. For our experiments, we used $n = 3$, which guarantees that all errors involving the specific setting of three or fewer parameters will be exercised by at least one test.

# Thresholds	# Bins	Combinatorial	1-factor	2-factor	3-factor	4-factor
5	5	$3 \cdot 10^3$	10	68 [<1s]	360 [<1s]	1,534 [2s]
12	5	$244 \cdot 10^6$	10	99 [<1s]	690 [2s]	4,355 [85s]
5	10	$100 \cdot 10^3$	20	255 [<1s]	2,768 [2s]	24,338 [100s]
12	10	$1 \cdot 10^{12}$	20	361 [2s]	5,317 [66s]	–
20	var	$87 \cdot 10^{18}$	34	556 [2s]	9,640 [610s]	–

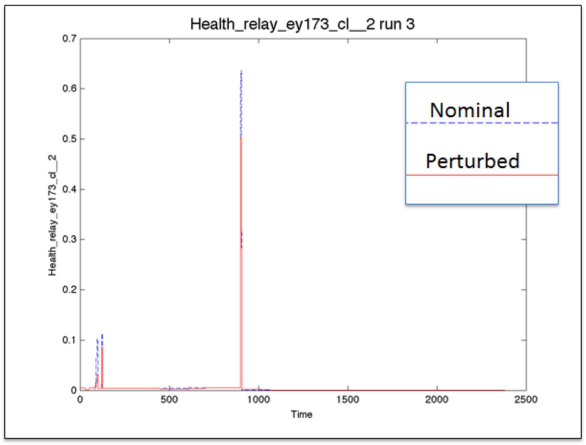
Table 2. Number of test cases generated with full exploration (combinatorial) and n-factor ($n = 1, 2, 3, 4$) exploration for various numbers of parameters and bins

In our experiment, the tool generated 1405 test-cases to be evaluated for the 300 threshold input space, each with 5 bins. The individual test-case were then evaluated using the DXC Framework, where the sensor data is passed to the BN model using a particular combination of modified thresholds.

Results for each run were obtained by collecting the posterior probabilities of health nodes for the BN at each timestep. These posterior probabilities are used to determine the health state of components in the system, indicating whether they are healthy or faulty. The ProADAPT BN outputs the posterior probabilities of 120 health nodes.

III.C. Effects of Parameter Perturbation

In this section, we present selected results of our initial experiments on parametric analysis of the ADAPT Bayesian HM with single fault scenario. The main purpose of this discussion is not to provide details on specific findings of the ADAPT BN and scenario, but to illustrate which kinds of analyses our parametric threshold testing supports, and which V&V relevant conclusions can be drawn.



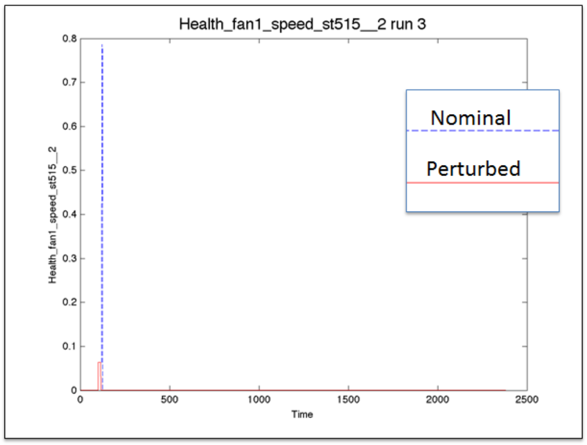


Figure 5. Selected health nodes: minimal differences (node Health_relay_ey173_cl, left) and major discrepancies (node Health_fan1_speed_st515, right)

The effects of parameter perturbation can be visualized by plotting the temporal trace of a health node which is the posterior probability of a certain state occurring over time. Figures 5, 6 are examples of such traces, showing the posterior probability of an unhealthy state occurring. Large changes are indicative of a change in state of a component in the ADAPT Testbed. Values for the perturbed trace (blue) are superimposed on the unperturbed trace (red). Figure 5(left) shows the posterior probabilities of the health for relay "ey173" (see Figure 2 for location of the relay) for a single test case with modified threshold values. Here the effect of a perturbed value (red) is only slightly different from the nominal; the Bayesian network computes the same result. In more severe cases, however, a fault can go undetected due to threshold modification. Figure 5(right) shows the nominal case detects a fault around the 100th timestep, which is undetected.

Threshold perturbation has other negative effects such as delaying a diagnosis Figure 6(left). In this case, there is a several hundred timestep difference between detection in the nominal case and the perturbed case,

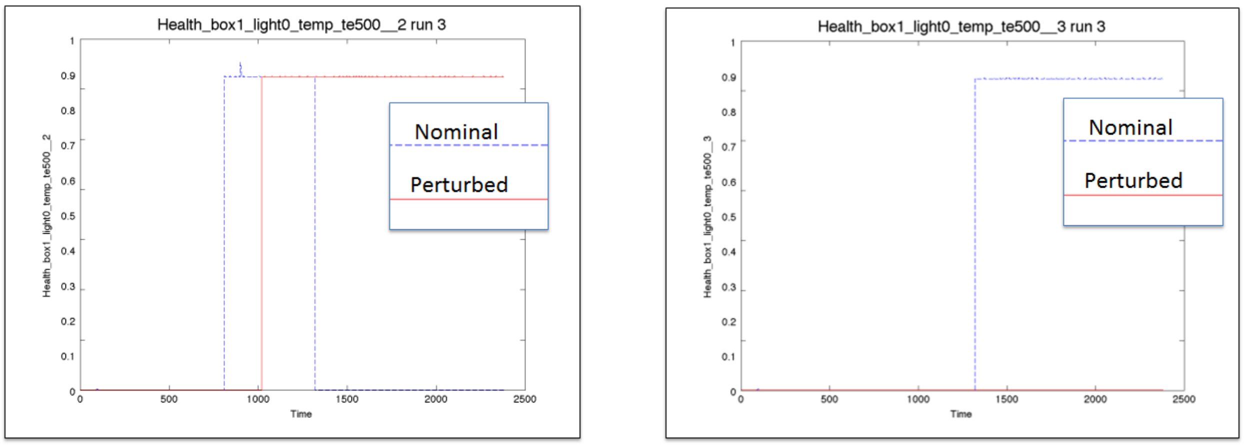


Figure 6. Temporal traces for a selected health node: event can be detected late (left) or not be detected at all (right).

corresponding to a 15 second delay in the diagnosis of a fault. Figure 6(right) shows a complete undetection of a change in state of the health node through the duration of the scenario with no change in posterior probability.

IV. Clustering Analysis

A manual analysis of the results of thousands of test cases with hundreds of variables is not possible. Therefore, specific analysis and visualization techniques are necessary to deal with this large, high-dimensional data set. In this paper we use clustering, an unsupervised machine learning technique to automatically find structure in this large data set, to analyze sensitivity and to support root-cause analysis. Since we are not interested in a full time-series analysis, we first extract features from the data, i.e., we summarize each temporal trace of a health node for each experiment by one or more scalar values. For example, we might choose the maximum posterior probability of this health node during the execution of the scenario.

Clustering then groups all test cases (each represented by a set of feature values) according to their similarity into several classes (usually up to around a dozen). Class membership for each test case can be used for visualization, report generation, and additional analysis. In the following, we discuss each of the individual processing steps and show results of experiments.

IV.A. Feature Selection

The analysis of each scenario produces a large amount of data containing time series of the posterior marginals of the health nodes of the model. In our running example, each scenario produces around 30 GB of data (2200 time steps t_k for 120 health node marginals x_j for 1405 test cases C_i). Stored as a single matrix, $M_{i,j,k}$ would be the posterior probability (a double value) of a health node x_j at timestep t_k with the threshold parameters set by test case C_i.

In order to reduce the size and dimensionality of the data set, we use feature selection (Table 3). A feature "summarizes" one temporal trace $M_{i,j,:}$[b] by a single scalar value. Selected traces of health nodes can be ignored or multiple, different features can be selected. Although feature selection loses substantial data, it allows the capture of information which is critical for the analysis and makes the learning task feasible. Table 3 lists the currently available features. The minimum value feature F_1, for example takes the minimum of any posterior value of the given health node for each test case. With this feature, the analysis can distinguish between cases where a health node always shows a healthy status and those where problems (low health posteriors) are annunciated. Similar to the maximum value, this feature can be used to detect cases of missed alarms (or false alarms).

[b]Here, we are using Matlab-style notation, where ":" denotes all indices of this dimension, in our case $1 \ldots 2200$.

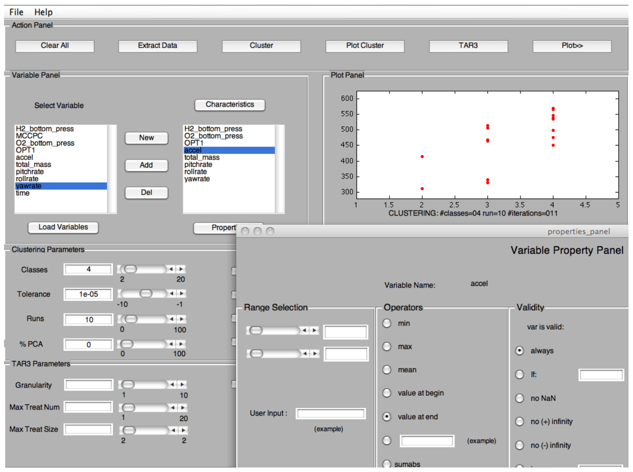
F_1	min value	$\min M_{i,j,:}$
F_2	max value	$\max M_{i,j,:}$
F_3	mean square error wrt nominal	$\frac{1}{C}\sum_k^N((M_{i,j,k} - M_{0,j,k})^2)$
F_4	KL divergence	$\frac{1}{C}\sum_k^N(M_{i,j,k}\log\frac{M_{i,j,k}}{M_{0,j,k}})$
F_5	min difference value	$\min \lvert M_{i,j,:} - M_{0,j,:}\rvert$
F_6	max difference value	$\max \lvert M_{i,j,:} - M_{0,j,:}\rvert$
F_7	tsteps larger than tol	$\sum_{\{k:\lvert M_{i,j,:}-M_{0,j,:}\rvert>tol\}} 1$

Table 3. List of features. $M_{i,j,:}$ (in Matlab-style notation) corresponds to the temporal trace of the health variable x_j for the run C_i, and $M_{0,:,:}$ to that of the run with nominal (unperturbed) thresholds. C is the number of test cases, N the length of a temporal traces. In our experiments, tol was set to 0.5.

The next set of features compare the current test case with a nominal test case C_0 with undisturbed thresholds. These features can be used to extract sets of test cases where the perturbed thresholds actually caused substantially different probabilities of the health nodes. Most commonly, we use the mean square error F_3.

Finally, F_7 is a feature which allows the analyst to use temporal properties for clustering. This feature gives an indication on how long (i.e., number of time steps) substantial differences in the posterior exists. It is helpful to distinguish cases where only short deviations from the nominal case (e.g., transients) are detected with more severe situations as shown in Figure 6. Similar features can be defined to directly extract delays in rising edges.

Within our clustering system, additional features can be easily defined. A graphical user interface (Figure 7) allows the user to select variables and features for each experiment. With this tool, bad test cases can be masked, temporal subranges selected, and customized features inserted (inset window). A major portion of the GUI is used to control the actual clustering experiments.



H2_bottom_press
MCCPC
O2_bottom_press
OPT1
accel
total_mass
pitchrate
rollrate
yawrate
time

H2_bottom_press
O2_bottom_press
OPT1
accel
total_mass
pitchrate
rollrate
yawrate

CLUSTERING: #classes=04 run=10 #iterations=011

Classes 4
Tolerance 1e-05
Runs 10
% PCA 0

Variable Name: accel

var is valid: always
value at end

Figure 7. GUI for feature selection and clustering

IV.B. Clustering

As described earlier, clustering is an unsupervised machine learning technique which takes a vector of features for each test case and sorts the test cases into a number of groups according to their similarity. More formally, clustering can be seen as solving a mixture problem. Let $f_{i,j}$ be the i-th selected feature for test case j. We furthermore assume that the feature values are Gaussian distributed, whereby mean values and standard deviations depend on the class c they belong to: $f_{i,j} \sim N(\mu_{i,c}, \sigma_{i,c}^2)$. The clustering task is now to estimate the values of μ and σ^2 for each of the classes such that the data are best explained ($\max p(f|\{\mu, \sigma^2\})$.

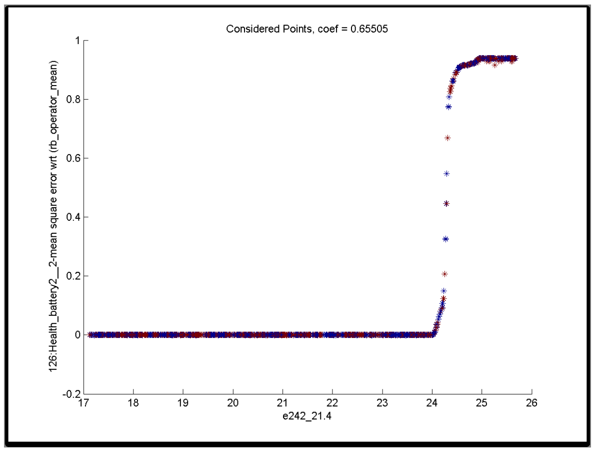
There are a number of numerical algorithms to solve this problem, most notably the Expectation Maximization (EM) algorithm[16] or k-means.

For our experiments, we use the AutoBayes tool[17c], which automatically generates customized data analysis algorithms from a compact statistical specification. We preferred this tool over a fixed algorithm because it can handle data which are not Gaussian distributed (e.g., for features, which return discrete values) and because of its user interface (Figure 7). The AutoBayes tool also generates a number of standardized HTML reports which describe the experiment with the used parametric variations and shows various plots of the data. For details on the use of the HTML reports see.[6] All figures in this paper have been extracted from these autogenerated reports.

IV.C. Experiments and Results

After features have been selected, clustering is performed, sorting each of the test cases into their respective class. Usually, we split the data into 3–7 classes. A good visual overview of the data is produced in the autogenerated report using scatter plots. Here the features are plotted over a single perturbed threshold, whereby each test case is represented as a single point. Figure 8(left) shows a scatter plot for the values of threshold `e242` on the x-axis. The nominal threshold value is 21.4 as indicated in the name. On the y-axis, the mean squared error of the health node of battery 2 (feature F_3) is shown. Each of the 1405 test cases is is shown as a cross. For values near or below the nominal threshold, the error is close to zero, which means that the BN reasoning is not sensitive to this variation. However, as this particular threshold is increased, a significant error is introduced. As the threshold surpasses 24, the error grows dramatically. From this we can conclude that modifying the 21.4 nominal threshold to an excess of 24 of component e242 will result in significant error in diagnosis of battery 2 health. In this case, the nominal value of 21.4 is considerably far away from the "cliff", so a sample safety-region around the nominal value exists.

In many situations, the error exhibits a sharp jump around some specific value of the threshold. Figure 8(right) shows that this does not have to be the case. Here, the error stays almost constant for test cases belonging to one class (shown in dark blue) and growing slowly and continuously for the other class.



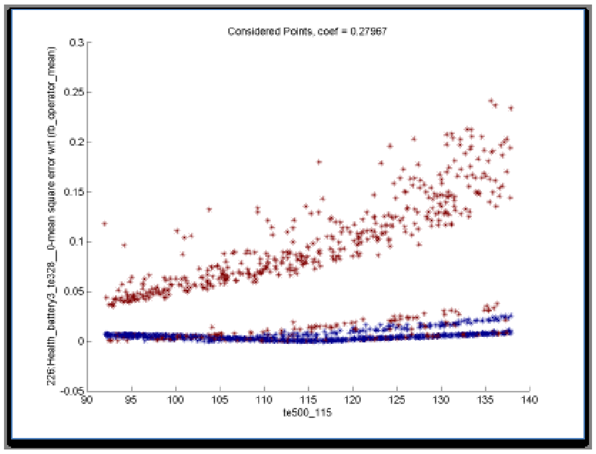


Figure 8. Scatter plot for error in health of battery 2 over threshold e242. A sharp increase in error can be detected for $\Theta_{e242} > 24.1$ (left). Continuously growing error depending on threshold variations (right).

Figure 9 shows how test cases are sorted into four different groups. Here, the minimum value of the health node te328 is shown. The lower left region (near a minimum value of zero) is entirely occupied by light and dark blue points, whereas all points to the right of the cliff belong to the orange and brown classes. It is easy to see that the light blue class only corresponds to values close to zero. Yellow points always correspond to values larger than 0.4. These two classes correspond to the two major groups of data. Of more interest to the analyst is the fact that points of the two other classes (dark blue, brown) cannot only be found in the very low and high regions, but also in an area where the posterior is between 0.1 and 0.2. This is a clear indication that some other threshold variations (not shown here) cause an independent (small) raise in the

[c]AutoBayes has been developed at NASA Ames and is available open source under the NOSA license.

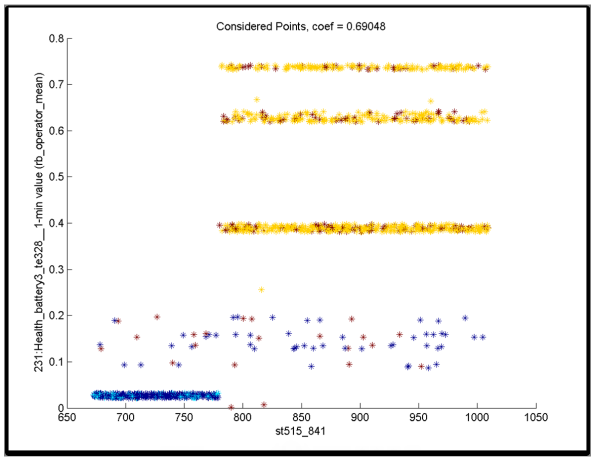


Figure 9. Scatter plots

posterior. Because this area is well separated from the low (< 0.05) and high (≥ 0.4) no false or missed alarms are to be expected.

Of more concern should be the outliers, which can be found in this figure: one test case with a threshold setting $\Theta_{st515} \approx 820$ yields a minimum value of around 0.25 (marked with a circle). Since all yellow cases are supposed to yield values larger than 0.4, this test case should be examined closely for transient effects, as even a very short transient spike affects the minimum value of that feature. A further indication of a transient effect are other outliers near that same threshold value.

In Figure 10(left) with only two classes, three major regions can be identified: (1) the blue class with errors close to zero (meaning: same behavior as the unperturbed BN), a random area (2) with errors around 0.5. Here the outcome of the reasoning can be the same as in the unperturbed case or different. However, variations in this specific threshold (st515) do not seem to cause this effect. Then, there is an area (3), where the perturbed BN will always return a wrong value; this is caused if the threshold is larger than 780 (dense brown area on the extreme right). With a nominal threshold of 661, this area is well away from nominal, so we could argue that slightly biased sensor readings do not do any harm. Figure 10(right) shows the scatter plot for the "high" value of the same threshold. If this threshold does not decrease too far, everything is fine (blue region). However, for values below 780, considerably larger errors are to be expected, although the errors are still much smaller than a result-flipping 0.5.

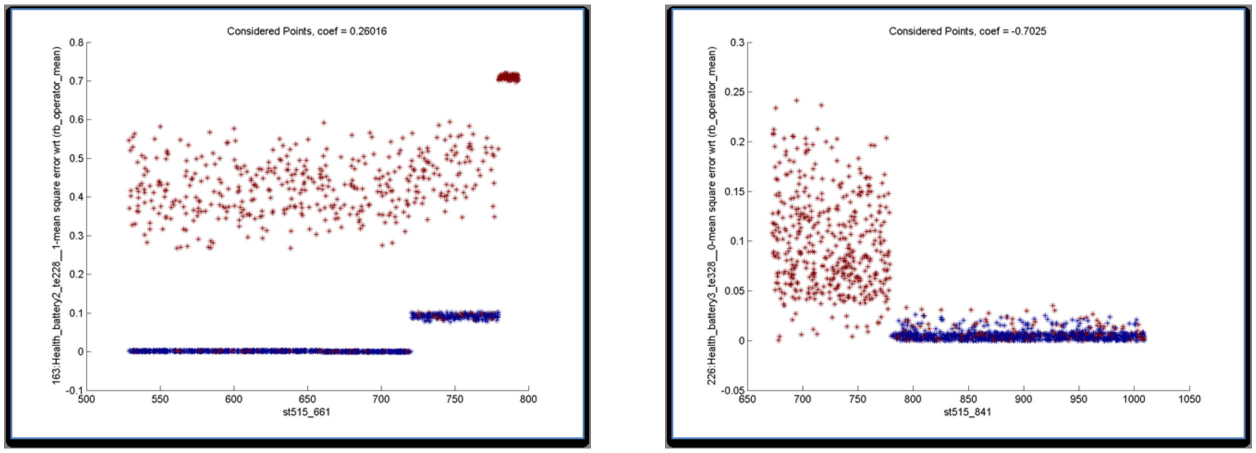


Figure 10. Scatter plots

Finally, Figure 11 shows results obtained from using the temporal feature F_7. A covariance analysis

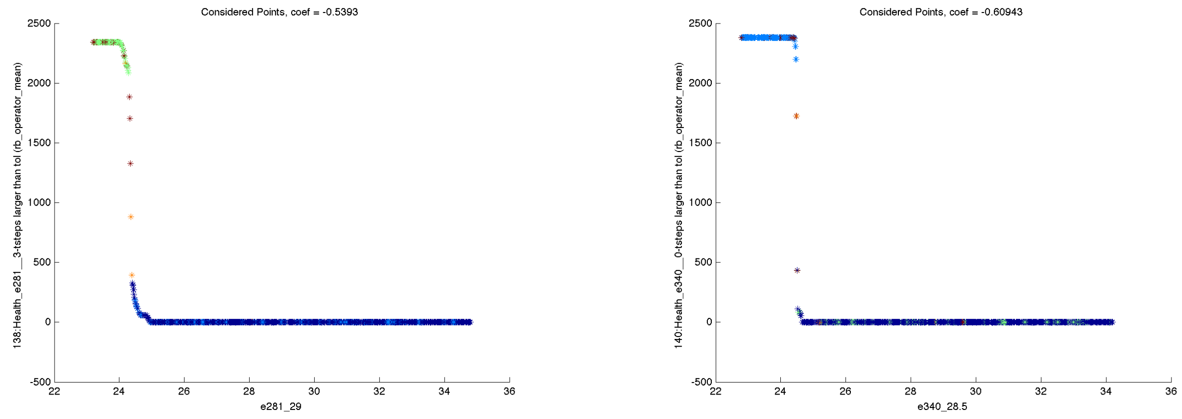


Figure 11. Scatter plots for temporal features

quickly reveals that only thresholds for e281 and e340 have any temporal influence in the sense that the overall deviation of a health value from the nominal case extends for longer periods of time. The majority of the test cases are clustered into one class (dark blue in Figure 11) representing situations where no temporal differences are observed (1078 of 1405 test cases). 179 (light blue) and 119 (light green) test cases only affected one of the signals. The remaining two classes where both signals are strongly influenced (25 of 1405) or somewhat influenced (4 of 1405, orange) are of most interest for the analysis.

Here again, clustering was able to structure the data set into different categories such that only a few test cases must be subjected to a more detailed, manual analysis. Using the autogenerated HTML reports described in the previous section can be used to specifically visualize these selected test cases.

V. Conclusions

In this paper, we have presented an advanced technique for the analysis of Health Management models. Using a combination of n-factor combinatorial exploration and Monte Carlo techniques, the health model is exercised with variations of perturbed parameters. Our technique yields a good coverage of the state space without making testing infeasible due to the combinatoral explosion. The analysis technique for health models used here is in the form of Bayesian Networks, but our approach is not limited to such a modeling formalism. We systematically perturbed discretization thresholds for the sensor nodes of the network. Incorrect threshold values can easily corrupt the diagnosis reasoning and lead to false alarms or missed events.

Our parametric testing tool generates hundreds to thousands of test cases. Each test run, when executed on the health model, produces temporal traces for the posterior values for each of the health nodes. Since such a large and high-dimensional data set cannot be analyzed manually, we use clustering, an unsupervised machine learning technique to automatically sort test results into groups and to find structure in this data set. With autogenerated HTML reports for individual runs and scatter plots to visualize clustering results, the analyst is able to quickly detect potential weaknesses and unwanted parameter sensitivity in the health model.

The experiments presented here deal only with one sort of perturbed model parameters: the thresholds. Another important source of parameters in a Bayesian health models are the entries in the CPTs. These tables contain a significant amount of design knowledge and information on the reliability of individual components (e.g., a component's mean time between failures (MTBF) converted to a probability). Only an in-depth analysis of such a health model can reveal if the model is appropriate to capture the failure modes if there are dangerous parameter sensitivities, or even if more reliable components are needed. This analysis can also be used to detect many classes of BN modeling errors, like missing or incorrect CPT values. Our approach to parametric testing and subsequent advanced analysis of the test data can help to address these questions.

Acknowledgements This work has been funded by NASA's Software Assurance Research Program (SARP) under the project "Advanced Tools and Techniques for V&V of IVHM Systems".

References

[1]de Kleer, J. and Williams, B. C., "Diagnosing multiple faults," *Artificial Intelligence*, Vol. 32, No. 1, 1987, pp. 97–130.

[2]Cordier, M.-O., Dague, P., Levy, F., Montmain, J., Staroswiecki, M., and Trave-Massuyes, L., "Conflicts versus analytical redundancy relations: a comparative analysis of the model based diagnosis approach from the artificial intelligence and automatic control perspectives," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, Vol. 34, No. 5, 2004, pp. 2163–2177.

[3]Srivastava, A. and Han, J., editors, *Data Mining in Systems Health Management: Detection, Diagnostics, and Prognostics (in Press)*, Chapman and Hall/CRC Press, 2011.

[4]Schumann, J., Srivastava, A., and Mengshoel, O., "Who guards the Guardians? — Toward V&V of Health Management Software (Short Paper)," *Runtime Verification 2010*, Springer, 2010.

[5]Darwiche, A., *Modeling and Reasoning with Bayesian Networks*, Cambridge University Press, Cambridge, UK, 2009.

[6]Schumann, J., Bajwa, A., and Berg, P., "Parametric Testing of Launch Vehicle FDDR Models," *AIAA Space*, 2010.

[7]Schumann, J., Gundy-Burlet, K., Pasareanu, C., Menzies, T., and Barrett, T., "Software V&V Support by Parametric Analysis of Large Software Simulation Systems," *Proc. IEEE Aerospace*, IEEE Press, 2009.

[8]Poll, S., Patterson-Hine, A., Camisa, J., Garcia, D., Hall, D., Lee, C., Mengshoel, O. J., Neukom, C., Nishikawa, D., Ossenfort, J., Sweet, A., Yentus, S., Roychoudhury, I., Daigle, M., Biswas, G., and Koutsoukos, X., "Advanced Diagnostics and Prognostics Testbed," *Proceedings of the 18th International Workshop on Principles of Diagnosis (DX-07)*, Nashville, TN, 2007, pp. 178–185.

[9]Mengshoel, O. J., Darwiche, A., Cascio, K., Chavira, M., Poll, S., and Uckun, S., "Diagnosing Faults in Electrical Power Systems of Spacecraft and Aircraft," *Proceedings of the Twentieth Innovative Applications of Artificial Intelligence Conference (IAAI-08)*, Chicago, IL, 2008, pp. 1699–1705.

[10]Ricks, B. W. and Mengshoel, O. J., "Methods for Probabilistic Fault Diagnosis: An Electrical Power System Case Study," *Proc. of Annual Conference of the PHM Society, 2009 (PHM-09)*, San Diego, CA, 2009.

[11]Ricks, B. W. and Mengshoel, O. J., "Diagnosing Intermittent and Persistent Faults using Static Bayesian Networks," *Proc. of the 21st International Workshop on Principles of Diagnosis (DX-10)*, Portland, OR, 2010.

[12]Mengshoel, O. J., Chavira, M., Cascio, K., Poll, S., Darwiche, A., and Uckun, S., "Probabilistic Model-Based Diagnosis: An Electrical Power System Case Study," Vol. 40, No. 5, 2010, pp. 874–885.

[13]Cohen, D., Dalal, S., Parelius, J., and Patton, G., "The combinatorial design approach to automatic test generation," *Software, IEEE*, Vol. 13, No. 5, Sep 1996, pp. 83–88.

[14]Dunietz, I. S., Ehrlich, W. K., Szablak, B. D., Mallows, C. L., and Iannino, A., "Applying design of experiments to software testing: experience report," *ICSE '97: Proceedings of the 19th international conference on Software engineering*, 1997, pp. 205–215.

[15]Wallace, D. R. and Kuhn, D. R., "Failure Modes in Medical Device Software: an Analysis of 15 Years of Recall Data," *International Journal of Reliability, Quality and Safety Engineering*, Vol. 8, No. 4, 2001.

[16]McLachlan, G. and Krishnan, T., *The EM Algorithm and Extensions*, Wiley Series in Probability and Statistics, John Wiley & Sons, New York, 1997.

[17]Schumann, J., Jafari, H., Pressburger, T., Denney, E., Buntine, W., and Fischer, B., "AutoBayes Program Synthesis System Users Manual," Tech. Rep. NASA/TM-2008-215366, NASA, 2008.