

A Framework for Systematic Benchmarking of Monitoring and Diagnostic Systems

Tolga Kurtoglu, Ole Mengshoel, Scott Poll

Abstract— In this paper, we present an architecture and a formal framework to be used for systematic benchmarking of monitoring and diagnostic systems and for producing comparable performance assessment for different diagnostic technologies. The framework defines a number of standardized specifications, which include a fault catalog, a library of modular test scenarios, and a common protocol for gathering and processing diagnostic data. At the center of the framework are 13 benchmarking metric definitions. The calculation of these metrics is illustrated on a probabilistic model-based diagnosis algorithm utilizing Bayesian reasoning techniques. The diagnosed system is a real-world electrical power system, namely the Advanced Diagnostic and Prognostic Testbed (ADAPT) developed and located at the NASA Ames Research Center. The proposed benchmarking approach shows how to generate realistic diagnostic data sets on large-scale, complex engineering systems, and how the generated experimental data can be used to enable “apples to apples” assessments of the effectiveness of different diagnostic algorithms.

Index Terms— Fault Detection, Fault Diagnosis, Systems Health Management, Bayesian Reasoning, Model-Based Diagnosis.

I. INTRODUCTION

SYSTEM health management (SHM) is of interest to most, if not all, of NASA’s missions. NASA’s spacecraft and aircraft contain multiple sub-systems including navigation systems, power systems, and propulsion systems, and it is crucial to keep these subsystems healthy during a mission [1-3]. Novel automatic or semi-automatic techniques – implemented in hardware, software, or both – have the potential of bringing increased autonomy and improved quality to SHM at NASA. In this paper, we put special emphasis on automated diagnosis and monitoring [3-7], including its application to sensor validation [8-11]. A wide range of algorithms for diagnosis and monitoring, both model-based and data-driven, have been developed. Unfortunately, there are several closely related challenges associated with developing

and deploying such algorithms and systems for aerospace vehicles; we high-light the following two in this paper. (In related work, the challenges of model construction and real-time reasoning are considered [4,11,12].)

First, there is a lack of SHM data sets that are realistic and standardized, in particular at the system and subsystem levels. This makes it difficult for researchers to benchmark diagnostics algorithms and systems (the data challenge). On the hardware side, different techniques and data sets, including destructive testing, non-destructive testing, accelerated life testing, reliability databases, etc. are available to support the component level. Unfortunately, there is a lack of similar data sets, at the system and sub-system levels, available for benchmarking diagnostic technologies (and in particular model-based approaches). This makes it more difficult to empirically evaluate integrated systems where software (implementing algorithms for diagnosis, monitoring, prognosis, etc.) monitor and control hardware sub-systems (such as electrical power systems) in a realistic fashion.

Second, there is a lack of support for comparative analysis of different diagnostic algorithms and systems; lack of common vocabularies and ontologies; and well-defined metrics (the evaluation challenge). This makes it difficult to understand the pros and cons of different alternatives, which might lead to sub-optimal design choices being made, with obvious unfortunate consequences for performance and safety.

In this paper, we present an architecture and a framework with the goal of improved benchmarking system health management systems. We consider model-based techniques for diagnosis and monitoring, and define relevant detection and isolation metrics. As a case study, we consider probabilistic model-based diagnosis utilizing Bayesian networks and arithmetic circuits. The diagnosed system is the Advanced Diagnostic and Prognostic Testbed (ADAPT), a real-world electrical power system. Our benchmarking approach shows how to generate realistic data and compute metrics in the context of ADAPT, thereby facilitating research on system health management systems. We emphasize the use of a common fault catalogue and common metrics, which together enable “apples to apples” assessments of the effectiveness of different technologies, including the probabilistic techniques investigated here.

The significance of this work is as follows. First, we start addressing the data challenge by presenting the ADAPT EPS, which provides a setting for generating standard, realistic

Manuscript received May 19, 2008.

Tolga Kurtoglu is with the Mission Critical Technologies/NASA Ames Research Center, Moffett Field, CA 94035 USA (phone: 650-604-1738; e-mail: tolga.kurtoglu@nasa.gov).

Ole Mengshoel is with the USRA-RIACS/NASA Ames Research Center, Moffett Field, CA 94035 USA (phone: 650-604-4199; e-mail: ole.j.mengshoel@nasa.gov).

Scott Poll is with the NASA Ames Research Center, Moffett Field, CA 94035 USA (phone: 650-604-2143; e-mail: scott.d.poll@nasa.gov).

problem sets for diagnosis and monitoring, including sensor validation. Second, by introducing a benchmarking architecture and uniform metrics, we enable systematic empirical evaluation of different model-based diagnostic algorithms, which again should lead to improved understanding and comparative analysis of different diagnostic algorithms. In other words, we start addressing the evaluation challenge. Such benchmarking results can eventually be used to select between or integrate different diagnostic approaches.

The organization of the rest of this paper is as follows: Section 2 presents a review of metrics defined for diagnostic health management techniques and summarizes associated benchmarking efforts. Section 3 introduces our proposed benchmarking framework and provides an overview of the approach. The fundamentals of the Advanced Diagnostic and Prognostic Testbed are described in Section 4. Section 5 presents major components of the benchmarking framework including the fault catalog, scenario descriptions, and the experimental procedures involved. Benchmarking metrics are defined in Section 6. Section 7 discusses an approach to probabilistic model-based diagnosis along with its application to the ADAPT EPS. Section 8 summarizes initial benchmarking results for our probabilistic approach, followed by concluding remarks.

II. RELATED WORK

The development of monitoring and diagnostic technologies is of great interest to military and industrial aerospace applications. As these algorithms become more readily available, the necessity for assessing the performance of alternative diagnostic tools becomes important. As a result, there is an increasing need to define metrics that will allow the evaluation and benchmarking of competing diagnostic technologies.

Several institutions and organizations have proposed metrics to address this need [13-19]. Among those, the SAE [13]'s "Health and Usage Monitoring Metrics" defines probability of detection and probability of false alarms as key metrics for evaluating diagnostic algorithms. In addition, these definitions are supplemented by a variety of measures and statistics to present the results.

DePold et al. [14-15] introduced metrics to evaluate the accuracy and cost effectiveness of diagnostic systems. Their approach is based on the receiving operating characteristics (ROC) analysis [16], which illustrates the trade-off space between the probability of false alarm and the probability of detection for different signal to noise ratio (SNR) levels. The method is used to test the relative accuracy of diagnostic systems based on different threshold settings. Later, they also proposed a combined metric [15] that accounts for consequential event costs including missed detection, false alarms, and misdiagnosis.

Another widely used metric for diagnostic accuracy is the Kappa Coefficient [13]. It is based on the construction of a confusion matrix that summarizes diagnostic results produced

by a reasoner over a number of test/use cases. The Kappa Coefficient is used to measure the ability of an algorithm to discriminate between multiple fault candidates.

Apart from these approaches, several researchers have attempted to demonstrate benchmarking capability in addition to defining evaluation metrics [17-19]. Among these, Orsagh et al. [17] provided a set of 14 metrics to measure the performance and effectiveness of prognostics and health management algorithms for US Navy applications [18]. The metrics are defined separately for detection, isolation, and prognosis. For detection, the metrics include thresholds, accuracy, reliability, sensitivity to load, speed, or noise, and stability. The isolation metrics are same as the detection metrics but also include measures for discrimination and repeatability. Finally, the prognosis metrics are concerned with predicted condition, and remaining life. They also combined individual metrics into a composite score by implementing a weighted average sum. Moreover, they defined effectiveness metrics as a separate category that can be used to incorporate non-technical aspects such as operation, maintenance and implementation costs, computer resource requirements, and algorithm complexity into the analysis. Using these metrics, one can assess the overall effectiveness and benefit of diagnostic health management systems. Other researchers have also proposed similar cost-benefit formulations for health management systems. [20-22]. These approaches, however, are primarily concerned with higher-level trade-off's in health management systems and are not focused specifically on the performance of monitoring and diagnostic algorithms.

Finally, Bartys et al. [19] presented a benchmarking study for actuator fault detection and identification (FDI). This method developed by the DAMADICS Research Training Network introduced a set of 18 performance indices used for benchmarking FDI algorithms on an industrial valve-actuator system. The indices measure the temporal performance of detection and isolation decisions, as well as true and false detection and isolation rates, sensitivity, and diagnostic accuracy. This benchmark study used real process data, and demonstrated how the performance indices can be calculated for 19 actuator faults using a single fault assumption.

III. BENCHMARKING FRAMEWORK AND ARCHITECTURE OVERVIEW

The overall goal of this research is to develop a formal framework to be used for systematic benchmarking of different health management systems and to produce comparable performance assessment for different diagnostic methods. To achieve this goal, a number of standardized specifications are defined which include a standardized fault catalog, a library of modular and standardized test scenarios, a test protocol, a common process for processing diagnostic data and the calculation of metrics.

The detailed architecture of the developed framework is presented in Figure 1. The physical system is a real-world electrical power system (EPS), namely the Advanced Diagnostic and Prognostic Testbed (ADAPT) developed and located at the NASA Ames Research Center. The EPS problem domain contains both discrete and continuous faulty behavior, which are defined under a standardized fault catalog. The fault catalog establishes a common ground as to what failure modes, and faulty behavior are required to be modeled by individual diagnostic algorithm developers. To date, ADAPT has worked with diagnostic algorithms from NASA (Hybrid Diagnostic Engine (HyDE) [23], Inductive Monitoring System (IMS) [24]), academia (FACT) [25]), and industry (such as TEAMS-RT [26]). The testing procedure is usually scenario-based, where each scenario may have faults injected into the system, either through a GUI that manages a fault simulation software or via physical implementation (more on this in Section V). To detect faults, each diagnostic algorithm has access to real-time data from the ADAPT EPS. Moreover, a standardized output scheme is enforced on the diagnostic algorithms to ensure the generation of common data sets for the calculation of metrics. The data from the testbed and this output of the diagnostic system are saved to a database, and the diagnostic algorithm performance is evaluated according to a predefined set of metrics. In the

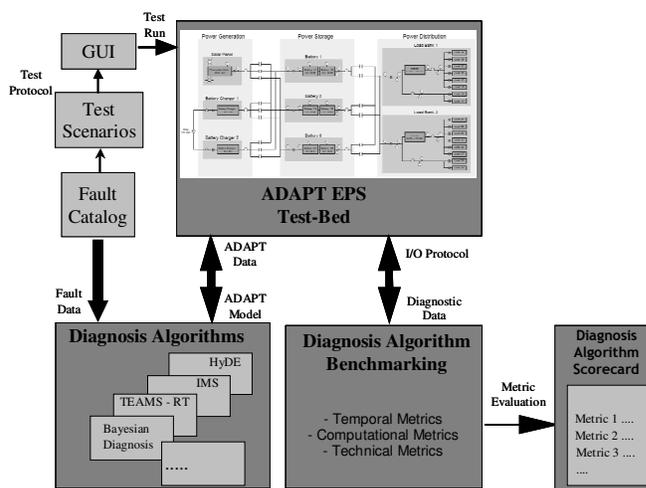


Fig. 1. The architecture of the benchmarking framework

following sections, specific modules of the framework are explained in detail with their interactions and specific roles, but first a brief summary of the ADAPT Testbed is provided.

IV. ADVANCED DIAGNOSTIC AND PROGNOSTIC TESTBED (ADAPT)

The Advanced Diagnostic and Prognostic Testbed (ADAPT) at the NASA Ames Research Center is a unique facility designed to test, measure, evaluate, and mature diagnostic and prognostic health management technologies. It provides a representative physical domain in the form of an electrical power subsystem (EPS) that enable automated diagnosis of complex systems exhibiting hybrid, i.e. both continuous and discrete behavior.

The layout of the ADAPT power system is shown in Figure 2. The EPS includes elements common to many aerospace applications: power storage, power generation, and power distribution. The power storage consists of three battery modules. Each of the three batteries can be charged by one of

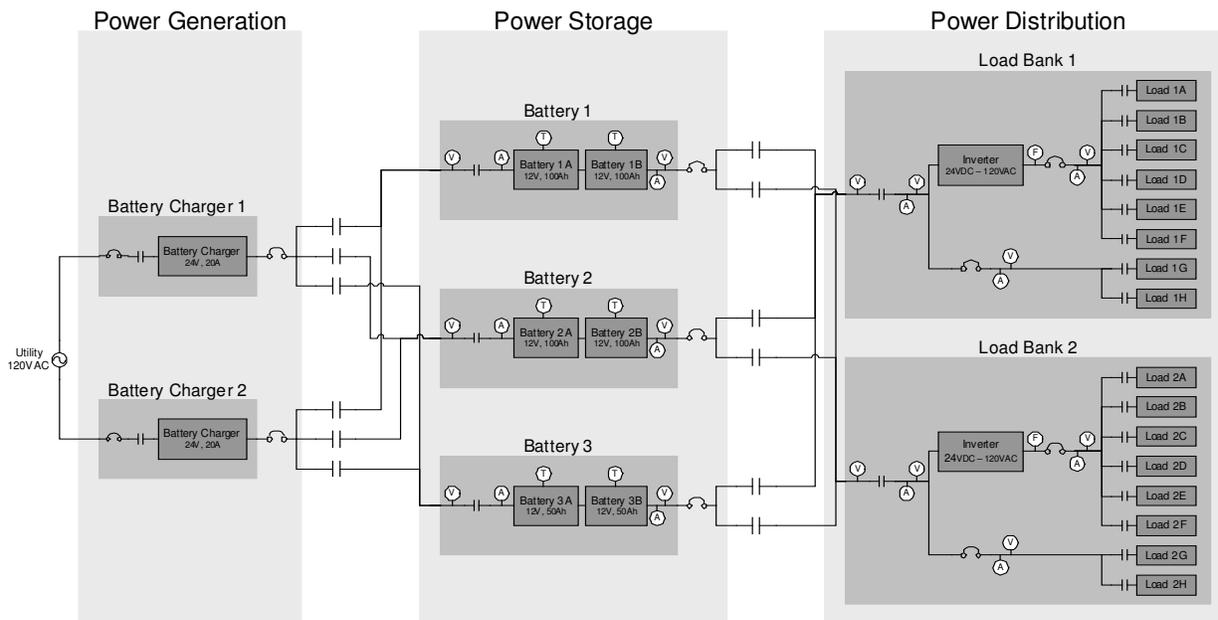


Fig. 2. ADAPT Testbed components and system configuration

the two battery chargers in the power generation element. Finally, any of the three batteries can be used to power any of the two load banks in the power distribution element. This design gives the ADAPT EPS basic redundancy and reconfiguration capability.

A data acquisition and control system sends commands to and receives data from the EPS. The testbed operator stations are integrated into a software architecture that allows for nominal and faulty operations of the EPS, and includes a system for logging all relevant data. In addition, the ADAPT system includes a high-fidelity simulation testbed, called the VIRTUAL ADAPT, that emulates the ADAPT hardware. This environment provides identical interfaces to the application system module through wrappers to the ADAPT network and allows the physical components of the testbed, i.e. the chargers, batteries, relays, and the loads to be replaced by simulation modules that generate the same dynamic behaviors as the hardware testbed. The simulation environment also provides for precise repetition of different operational scenarios that facilitates more rigorous testing and evaluation of different diagnostic algorithms. (More information on the ADAPT testbed can be found in [2].

V. BENCHMARKING FRAMEWORK DESCRIPTION

In order to effectively establish a systematic framework for benchmarking of diagnostic algorithms, our method describes specific tasks and requirements and defines a set of hierarchically organized metrics that can be used for “apples to apples” comparison of different diagnostic technologies. A number of key technical challenges exist in developing this framework.

First, it is required that the criteria which will be used as a basis for comparison are clearly defined. In particular, the

metrics should be measurable, analytically computable, and independent of any specific diagnostic technology.

Second, the description of system characteristics, most notably fault representations and healthy system states, should be standardized. In addition, it is required that a broad collection of fault data is used covering a wide spectrum of fault types and behaviors in order not to bias the benchmarking results towards certain algorithms that may perform well for a specific subset of fault domains.

Third, a repeatable, standardized process should be developed for scenarios and for processing resulting data sets. Part of this challenge is to ensure that a level of consistency is maintained for collection and processing of the diagnostic data. More importantly, it is required to establish a standardized scheme for representing the typical output of diagnostic algorithms, i.e. the estimation of health status of a physical system.

In order to address these challenges, the proposed benchmarking framework provides an architecture that can be used for generating standard, realistic problem sets and associated data for diagnostic analysis. The architecture includes clearly defined performance metrics, a common fault catalog, and standardized representations and procedures required for collecting and processing diagnostic data. The details of the framework are explained in the following sections followed by a detailed discussion of the performance metrics in the next section.

A. Fault Catalog

Different fault types and behaviors embodied within the ADAPT testbed are defined under a fault catalog. There are fundamentally two classes of faults that are being currently used as part of this study: *abrupt* faults and *incipient* faults.

All faults are classified based on the physical components of

ADAPT Fault Catalog			
Fault ID	Component	Fault Mode	Fault Class
Battery			
Component ID's	BAT1A, BAT2A, BAT3A, BAT1B, BAT2B, BAT3B,		
f1		Battery Overheat	abrupt
f2		Voltage Drain	incipient
f3		Voltage Failure	abrupt
f4		Terminal Corrosion/Short	abrupt
f5		Battery Leak/Level Loss	incipient
f6		Reverse Connection	abrupt
Circuit Breaker			
Component ID's	EY136, EY162, EY166, EY180, EY236 EY262, EY266, EY280, EY336		
f7		Failed Open	abrupt
f8		Stuck Closed	abrupt
f9		Failed Closed	abrupt
Inverter			
Component ID's	INV1, INV2		
f10		Switched Off	abrupt
f11		Short	abrupt
f12		Overheat	incipient
f13		Improper Output Voltage	incipient

Fig. 3. An excerpt from the ADAPT Fault Catalog

the ADAPT EPS. These faults are determined by conducting a failure modes and effects analysis (FMEA) [27] on the EPS and by amalgamating component failure information from various reliability documents.

Figure 3 shows an excerpt of the fault catalog. The first column lists the “unique ID” for each fault type, and the second column depicts the generic component type for the faulty behavior. For example, faults 1-6 capture “battery” faults, whereas, faults 7-10 define “circuit breaker” faults. Note that, physical locations where these faults could manifest themselves (labeled as “component ID’s) are also listed in a separate row under each generic component type. Listed next in the fault catalog are the fault modes of components, followed by the fault classes (abrupt vs. incipient). Currently, there are over 90 faults classified under 14 generic component types. In addition, each component type has its own non-faulty mode(s) defined. For example, a “circuit breaker” has healthy nodes of “nominal_open” and “nominal_closed”.

The fault catalog along with the non-faulty component mode definitions serves as a requirements document for individual diagnostic algorithm developers and is intended to guide the modeling of the physical system by establishing a common ground among different diagnostic tools.

B. Scenario Descriptions

An important aspect of the benchmarking process is the ability to test the diagnostic performance over a wide variety of operational scenarios. This is facilitated by the definition of standardized test scenarios.

A test scenario defines the EPS’s initial configuration, load configuration, and fault configurations for a test run. The *system initial configuration* describes basic connectivity of the major elements of the EPS testbed, i.e. the connectivity between power storage, power generation, and power distribution elements. For example, Battery 1 may be connected to Load Bank 1, 2, or both. Similarly, Battery 1 may be charged by the by either of the battery chargers. The *load*

configuration describes the loads that are used during a test run. Currently, the ADAPT EPS system can be configured to have up to 6 AC, and 2 DC loads at each load bank. The AC loads include fans, lights, and pumps with different operating characteristics. Finally, the *fault configuration* describes the fault or faults that are experimented. It includes a unique fault-id, injection time and injection location of the fault, any required fault parameters, and the source of the fault that captures whether the fault was generated by the simulation or was induced through a hardware implementation.

Figure 4 illustrates the execution of a simple scenario. In this test run (experiment ID#266), Battery 1 feeds Load Bank 1. Two AC, and one DC load is used: Light 6 on load location L1A, Fan 1 at L1B, and a DC load at L1G. A single fault is injected on Inverter 1 (fault ID f10 in Figure 3) through hardware set-up. Accordingly, the inverter is switched off at about 41 seconds into the experiment. No fault parameters are required for this fault type. The fault description is also provided in Figure 4.

C. Testing Procedures, Input’s, and Output’s

As stated earlier, a common process is developed for running repeatable, standardized scenarios and for generating and processing common data sets. This section describes the input’s and output’s given to and generated by the ADAPT testbed during each test scenario.

In order to provide a level of consistency between the diagnostic algorithms, we offer a simple reference architecture shown in Figure 5. According to this architecture, each diagnostic algorithm is treated as a black box with inputs and output as shown in the Figure.

In this architecture, a diagnostic system is characterized by its model M of the ADAPT testbed, as well as how the inputs from and outputs to the ADAPT testbed are mapped into the constructs used by the algorithm. The inputs to the diagnostic algorithm are of two main types, namely:

- Commands c(t): Commands at time t to the ADAPT

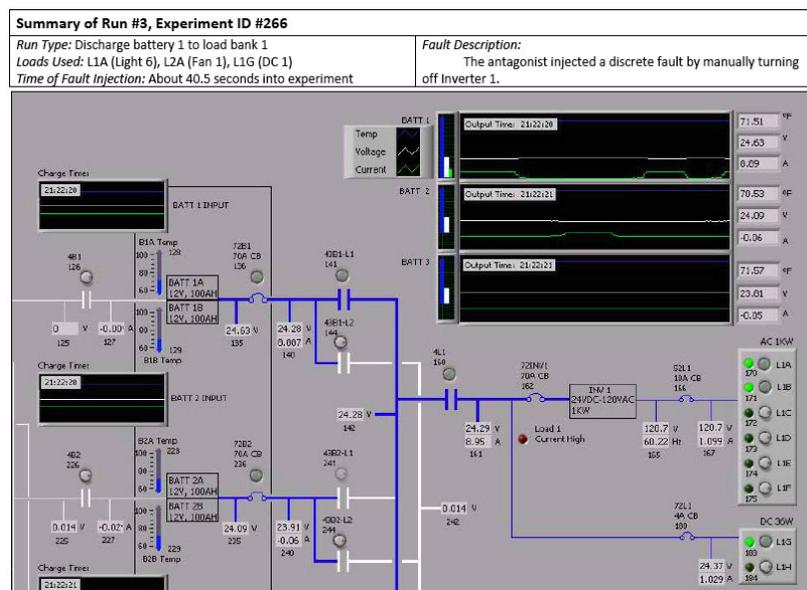


Fig. 4. A summary of a test run

testbed from the ADAPT user. This represents constraints on the desired state of ADAPT.

- Sensor readings $s(t)$: Sensor readings at time t – such as voltage, current, and temperature – from ADAPT. Note that because of sensor failures, some of the readings might be incorrect.

In the architecture, a diagnostic algorithms output is an estimate of ADAPT’s health status $h_{DA(i)}(t)$, which typically includes the health of ADAPT’s sensors and the health of ADAPT excluding sensors.

In addition, the format of $h(t)$ is standardized to facilitate the generation of common data sets and the calculation of the benchmarking metrics which will be introduced in the next section. Accordingly, each diagnostic algorithm is required to output the following through a common API architecture:

- Detection Signal (DS): A binary value (high or low) as to whether the diagnostic system has detected a fault.

- Isolation Signal (IS): A binary value (high, low) as to whether the diagnostic system has isolated a fault or a set of faults. Each “high” isolation signal is associated with a candidate fault set that summarizes the estimate of the health status of the ADAPT system.

- Candidate Fault Set (CFS): A candidate fault set is a list of diagnoses, i.e. estimated system components that are identified as “faulty”. Each element in the candidate fault set is a pair in the form of (Component_ID, Fault_ID). Together the pair captures the particular instance of a component that is diagnosed as faulty, and the associated fault mode as defined by the fault catalog. For example, the candidate set corresponding to the correct diagnosis for the scenario presented in Figure 3 is {(Inverter 1, f10)}.

This output by the diagnostic algorithms can be generated at appropriate time steps based on the data rate of the testbed, or it can be reported at discrete points in time if there is a change in any one of the three values.

D. Fault Injection

ADAPT supports the repeatable injection of faults into the system in one of two ways [28]. First, faults may be physically injected at the testbed hardware. A simple example is tripping a circuit breaker using the manual throw bars. Another is using the power toggle switch to turn off the inverter. Relays may be failed by short-circuiting the appropriate relay terminals. Wires leading to or from sensors may be short-circuited or disconnected. Additional faults include loosening the wire connections in power-bus common blocks. Faults may also be introduced in the loads attached to the EPS. For example, the valve can be closed slightly to vary the back pressure on the pump and reduce the flow rate.

In addition to fault injection through hardware, faults may be introduced via software. Software fault injection includes one or more of the following: 1) sending commands to the testbed that were not intended for nominal operations 2) blocking commands sent to the testbed 3) altering the testbed sensor data. The sensor data can be altered in a number of ways. For a static fault, the data are frozen at previous values and remain

fixed. An abrupt fault applies a constant offset to the true data value. An incipient fault applies an offset that starts at zero and grows linearly with time. Excess sensor noise is introduced by adding Gaussian or uniform noise to the measured value. Future work will add intermittent data faults, data spikes, and the ability to introduce more than one fault type for a given sensor at the same time. By using these three approaches to software fault injection, fault scenarios may be constructed that represent diverse component faults.

Since some fault scenarios may be costly, dangerous, or impossible to introduce in the actual hardware, a simulation module called VIRTUAL ADAPT also provides fault injection capabilities. For example, degradation in the batteries can be simulated as an incipient change in a battery capacitance parameter. Other parametric faults can also be injected and simulated. In addition, VIRTUAL ADAPT permits experimentation with fault scenarios that cannot be realized in the hardware, such as an inverter malfunction. Currently, mostly discrete failures (e.g., relay failures) and sensor errors are introduced into ADAPT, so the simulation provides added functionality by enabling injection of other types of fault scenarios.

VI. BENCHMARKING METRIC DEFINITIONS

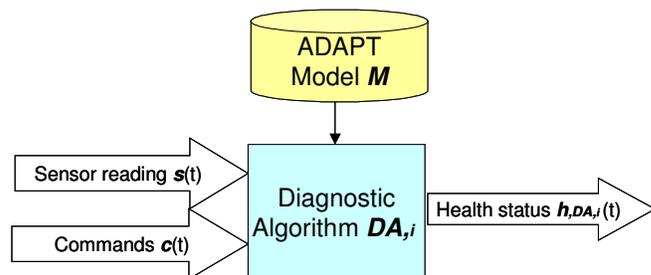


Fig. 5. Diagnostic algorithm reference architecture. Each diagnostic algorithm, DA_i is inputted a stream of sensory readings and a list of commands. The output from each diagnostic algorithm is an estimation of the health status of the ADAPT testbed.

A set of 13 metrics has been defined for assessing the performance of the diagnostic algorithms. These metrics are structured using two different classification schemes.

First, the metrics are classified as either *detection metrics* or *isolation metrics* as shown in Figure 6. In defining this classification, a distinction has been made between two basic functions that can be provided by a diagnostic algorithm [17,19]. According to this distinction, detection is defined as “the indication of malfunction in the system”. By nature, fault detection reasoning is a binary classification of the system state (faulty or non-faulty). On the other hand, isolation is defined as “the determination of the fault mode and location in the system”. Contrary to detection, isolation is a multi-state reasoning problem as there will be multiple candidates for fault modes and locations in a faulty system. This makes the benchmarking of isolation functionality a far more challenging task with several requirements.

To produce reliable and realistic benchmarking results for

isolation functionality, it is required that a consistent level to be defined at which faults in the system are assumed to be located (isolation level). Depending on the application, isolation may be performed at the line replaceable unit (LRU) level as is the case for most maintenance driven ground-based diagnostics, or it may be performed at the component or failure mode level as required by most real-time, on-board diagnostic applications. As one may expect, this selection directly affects the scope of the modeling efforts. To eliminate variations in the scope of different system models, it is also required that a common set of fault definitions is provided that the algorithms are expected to reason about (isolation set). For the benchmarking study reported here, the component failure modes as defined by the fault catalog are used as the common isolation level and the fault catalog along with the non-faulty component mode definitions provides a common isolation set.

Secondly, the metrics are grouped under *temporal metrics* that measure time response of diagnostic algorithms and *static metrics* that measure non-temporal features of a diagnostic algorithm including accuracy, resolution, sensitivity, and stability [18]. These metrics are again shown in Figure 6.

The *temporal metrics*, one for detection (metric 1) and two for isolation (metric 8-9), attempt to measure how quickly the diagnosis algorithms respond to faults in the physical system.

The *static accuracy metrics* (metrics 2-5 for detection accuracy and metrics 10-11 for isolation accuracy) are intended to measure the correctness of the detection and

isolation estimates by an algorithm.

The *static resolution metric* (metric 12) attempts to measure the resolution of isolation estimates. Ideally, an isolation estimate should include all the actual fault cases present in the physical system and nothing more. However, in reality, it is often necessary to lower the resolution setting of diagnostic algorithms for the sake of better accuracy [29]. Practically, this means that an isolation estimate may include other faults in addition to the actual fault cases present in the system.

The *static sensitivity metric* (metric 6) is intended to measure the detection response to the relative strength of faults present in the system.

The *static stability metrics*, one for detection (metric 7) and one for isolation (metric 13), attempt to measure the level of fluctuation in detection and isolation estimates. A detection and isolation estimate that fluctuates is difficult to interpret and often times undesirable [18]. These metrics are designed to favor stable detection and isolation estimates by a diagnostic algorithm.

A. Detection Metrics

The seven detection metrics are defined as:

Metric 1 - Time to Detect: The period of time from the beginning of a fault injection to the moment of the first “high” detection signal as shown in Figure 7. In the figure, t_{inj} is the time of fault injection, t_{dsig} is the time of first high detection signal. Time to detect t_{det} then becomes: $t_{det} = t_{dsig} - t_{inj}$, where $t_{det} > 0$. The metric is calculated for each scenario when it is applicable, i.e. when there is a fault injected. (Figure 7 illustrates the profile of an abrupt fault).

Metrics 2-5 are *detection accuracy metrics* and are defined

"Detection" Metrics	
Temporal Performance	
METRIC 1	Time to Detect
Static Performance	
Accuracy	
METRIC 2	Detection False Positive Rate
METRIC 3	Detection False Negative Rate
METRIC 4	Fault Detection Rate
METRIC 5	Fault Detection Accuracy
Sensitivity	
METRIC 6	Detection Sensitivity Factor
Stability	
METRIC 7	Detection Stability Factor
"Isolation" Metrics	
Temporal Performance	
METRIC 8	Time to Isolate
METRIC 9	Time to Estimate
Static Performance	
Accuracy	
METRIC 10	Isolation Classification Rate
METRIC 11	Isolation Misclassification Rate
Resolution	
METRIC 12	Size of Isolation Set
Stability	
METRIC 13	Isolation Stability Factor

Fig. 6. A summary of metrics used for benchmarking activity

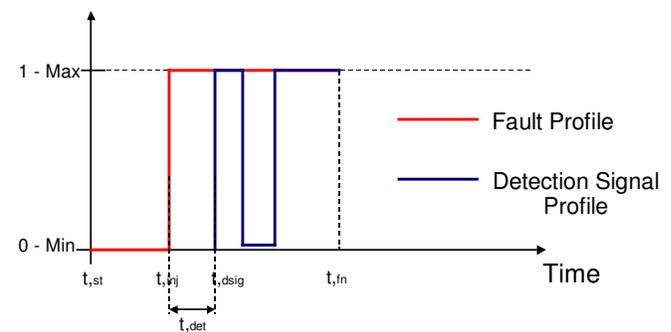


Fig. 7. The definition of “time-to-detect” metric

based on the construction of a decision matrix [1]. A decision matrix is a binary classification matrix that represents the distribution of predicted vs. actual states of faulty and non-faulty cases as shown in Figure 8.

The diagonal in the decision matrix are the correct predictions. The faulty cases equal to b+d, and the non-faulty

		Actual State	
		Non-Faulty	Faulty
Predicted State	Non-Faulty	a	b
	Faulty	c	d

Fig. 8. The decision matrix

cases equal to $a+c$. Based on the decision matrix, metrics 2-5 are defined as:

Metric 2 – Detection False Positive Rate: The ratio of cases where a fault is detected while the system was actually non-faulty which equals to $c/(a+c)$.

Metric 3 – Detection False Negative Rate: The ratio of cases where a fault is missed while the system was actually faulty which equals to $b/(d+b)$.

Metric 4 – Fault Detection Rate: The ratio of cases where a fault is detected while the system was actually faulty which equals to $d/(d+b)$.

Metric 5 – Fault Detection Accuracy: The ratio of correctly classified cases to the total number of cases which equals to $(a+d)/(a+b+c+d)$.

The next metric is intended to measure the *sensitivity of detection* and is defined as:

Metric 6 – Detection Sensitivity Factor: The relative strength of a fault when “detection” occurs [7]. For abrupt faults, fault strength is discretized into four qualitative values {0.25, 0.50, 0.75, and 1.00}, whereas for incipient faults a continuous scale between 0.0 and 1.0 is used to represent fault strength. The sensitivity factor corresponds to the relative level of fault strength where the detection signal becomes “high” as shown in Figure 9. The metric is calculated for each scenario.

The next metric is intended to measure the *stability of detection* and is defined as:

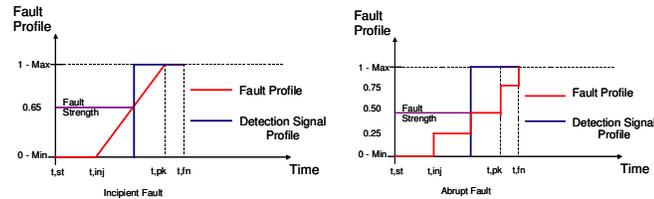


Fig. 9. The definition of “detection-sensitivity-factor” metric

Metric 7 – Detection Stability Factor: The level of stability of the detection signal measured as a percentage of the sum of duration of “high” detection signals to the total time elapsed after fault injection. The metric is calculated for each scenario and is illustrated in Figure 10.

B. Isolation Metrics

As defined earlier, isolation is “the determination of the fault mode and location in the system”. Contrary to detection, isolation is a multi-state reasoning problem as there will be multiple candidates for fault modes and locations in a faulty system. Moreover, the isolation candidates identified by a diagnostic algorithm may change as more data and

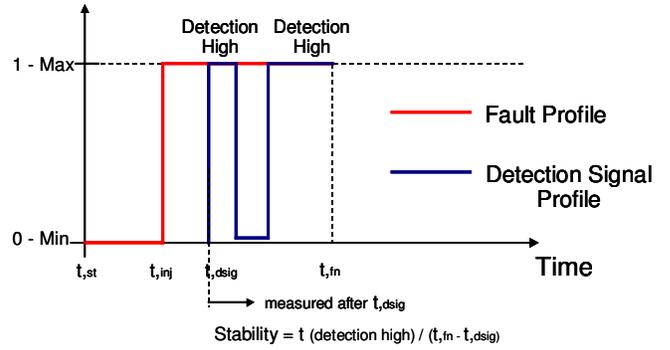


Fig. 10. The definition of “detection-stability-factor” metric

computation time becomes available as shown in Figure 11.

In the figure, t_{inj} is the time of fault injection, t_{isig1} and t_{isig2} are the time of isolation signals corresponding to candidate fault sets 1 and 2. For example, candidate set 1 might correspond to inverter1_switched_off or {(Inverter 1, f10)} using the standardized diagnostic output format, and candidate set 2 might correspond to circuit_braker_166_failed open or {(EY166, f7)}. Due to this time variant nature of isolation estimates, it is required to define what constitutes an algorithms final diagnostic output that is to be used for calculating the isolation metrics.

To address this, we define the concept of *end-of-scenario isolation set* which will be treated as the final diagnostic

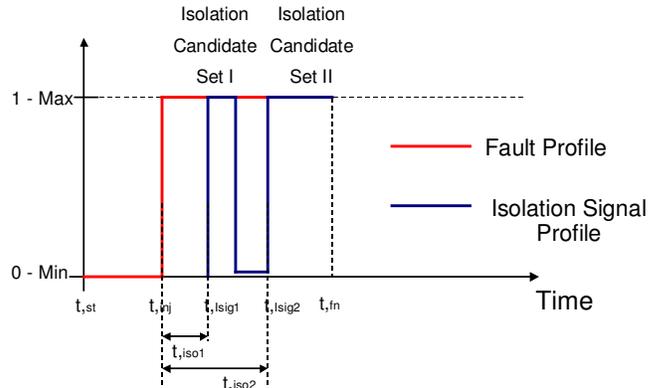


Fig. 11. The definition of “time-to-isolate” metric

output of an algorithm. For simplicity, an *end-of-scenario isolation set* is defined as the candidate fault set that corresponds to the last “high” isolation signal. For example, for the case used in Figure 11, candidate set 2 becomes the end-of-scenario isolation set. We are currently investigating other definitions for the end-of-scenario isolation set that takes fault injections and temporal aspects of different isolation estimates. However, for the purposes of this paper, the aforementioned definition is used to calculate the following 6 isolation metrics:

Metric 8 - Time to Isolate: The period of time from the beginning of a fault injection to the moment of last “high” isolation signal, i.e. the signal that corresponds to the end-of-scenario isolation estimate. Consider Figure 11 again. In the

figure, t_{inj} is the time of fault injection, t_{sig1} and t_{sig2} are the time of isolation signals corresponding to candidate set 1 and 2. Since candidate set 2 is the end-of-scenario isolation set, the time to isolate t_{iso} metric becomes: $t_{iso} = t_{sig2} - t_{inj}$, where $t_{iso} > 0$. Similar to the detection temporal metric, this metric is also calculated for each scenario when it is applicable. When multiple faults are present, time to isolate metric is calculated by averaging individual isolation times over the number of faults injected to the system.

Metric 9 - Time to Estimate: The cumulative time spent by an algorithm to estimate the physical state of the system. This metric is calculated by summing over the time periods from the reading of sensor values and commands to the moment of producing an health status estimate as shown in Figure 5.

Metrics 10-11 are *isolation accuracy metrics* and are defined based on the construction of a confusion matrix [1,6]. A confusion matrix is an expanded version of a decision matrix that incorporates fault classification as shown in Figure 12 for a “relay” component. In this example, the “relay” can have one healthy (non-fault), and two faulty (stuck_open, stuck_closed) modes. The classification problem then becomes determining what mode the relay will be in.

Similar to the decision matrix, the diagonal values in the

		Actual State		
		Relay	Non-Faulty	Stuck_open
Predicted State	Non-Faulty	A	F	K
	Stuck_open	B	G	L
	Stuck_Closed	C	H	M

Fig. 12. The confusion matrix

confusion matrix captures correctly isolated cases, whereas the off-diagonal elements are incorrect diagnoses. Cumulatively, the confusion matrix summarizes an algorithm’s ability to discriminate among multiple fault candidates. In most cases, the confusion matrix is expressed in a normalized form. When normalized, each cell value in the confusion matrix represents the probability of that case occurring. In addition, the non-faulty row and column can be removed from the matrix in order not to bias the results towards no-fault cases and to ensure that the matrix represents a measure of discrimination between faults once it has been determined that a fault is actually present. A common measure, the Kappa Coefficient [1]¹, for example is calculated using a confusion matrix without the no-fault entries.

The confusion matrix can be constructed at the system or individual component level. In this study, a *normalized* confusion matrix is built for each system component, which summarizes the probabilities of a component’s classification cases over a series of scenarios. Metrics 10 and 11 then are defined as:

Metric 10 – Isolation Classification Rate: The rate of correct classification rate by the isolation algorithm. This metric equals to the sum of probabilities along the diagonal.

Metric 11 – Isolation Misclassification Rate: The rate of misclassification by the isolation algorithm. This metric equals to the sum of probabilities along the off-diagonal.

The next metric is intended to measure the *resolution of isolation* and is defined as:

Metric 12 – Size of Isolation Set: The number of candidates in the end-of-scenario isolation set. The metric is calculated for each scenario.

The next metric is intended to measure the *stability of isolation* and is defined as:

Metric 13 – Isolation Stability Factor: The level of stability of the isolation signal measured as a percentage of duration of “end-of-scenario isolation set” to total time elapsed after fault injection. The metric is calculated for each scenario and is illustrated in Figure 13.

VII. CASE STUDY: PROBABILISTIC DIAGNOSTICS

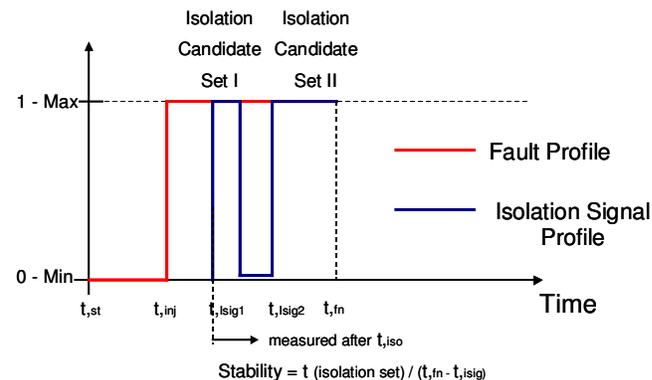


Fig. 13. The definition of “isolation-stability-factor” metric

Our probabilistic approach to diagnosis and state estimation is based on Bayesian networks [30] and arithmetic circuits. Both formalisms have been used to represent and reason with multi-variate probability distributions. Our emphasis in this paper is on their application in ADAPT and benchmarking.

There are two broad classes of approaches to Bayesian network inference: Interpretation and compilation. In interpretation approaches, a Bayesian network is directly used for inference. In compilation approaches, a Bayesian network is (off-line) compiled into a secondary data structure, where the details depend on the approach being used, and this secondary data structure is then used for (on-line) inference. Due to their high level of predictability and fast execution times, compilation approaches are especially suitable for resource-bounded reasoning and real-time systems [31]. Our focus here is on compilation approaches, and in particular the tree clustering (or clique tree, or join tree) approach and the

¹ Not used in this study

arithmetic circuit approach [32,33].

Under the tree clustering paradigm, a Bayesian network is transformed into join tree [5] during compilation. During propagation, evidence is propagated in that join tree, leading to belief updating or belief revision computations as appropriate. In practice, tree clustering often performs very well on relatively sparse BNs as are often developed by formalizing expert knowledge. However, as Bayesian network connectivity (expressed, for example, as the ratio of the number of leaf nodes to the number of non-leaf nodes) increases, the size of the maximal minimal clique size and the total clique tree size can grow dramatically [34,35], and thus care is needed when Bayesian network are designed and compiled.

A second compilation approach is the construction of arithmetic circuits from Bayesian networks [36,32,33]. An arithmetic circuit has a relatively simple structure, but can be used to answer a wide range of probabilistic queries. Compared to tree clustering, the arithmetic circuit approach exploits local structure and often has a longer compilation time but a shorter inference time. In the following we emphasize arithmetic circuits, which have given excellent performance in the ADAPT setting [4,11,12].

We assume a time-sliced Dynamic Bayesian Network (DBN) model M of ADAPT. This DBN represents ADAPT's failure modes, operational modes, as well as other features of the EPS. A DBN is essentially a multi-variate stochastic process, structured as a directed acyclic graph, with discrete time t . Suppose that the set of random variables (nodes in the BN) at time t is $X(t)$; these nodes can be partitioned as follows:

- Health nodes $H(t)$: There are two types of health (or output) nodes in the BN model:
 - Component health nodes: $H_C(t)$: Represent the health of a system (such as ADAPT) excluding sensors, both failure modes and operational (nominal) modes.
 - Sensor health nodes $H_S(t)$: Represent the health of a system's sensors, both their failure modes and operational (nominal) modes.
- Evidence nodes $E(t)$: There are two types of evidence (or input) nodes in the BN model:
 - Command nodes $E_C(t)$: System commands, in our case commands to the ADAPT testbed from the user. This represents the desired, but perhaps not actual, state of the system.
 - Sensor nodes $E_S(t)$: Sensor readings – such as voltage, current, and temperature for ADAPT. Because of sensor failure, some sensor readings might be incorrect.
- Remaining nodes $R(t)$: Nodes that reflect parts of the system that do not do not fit into any of the categories above.

Information from sensors and the environment (user) is incorporated into the probabilistic model and reasoning

process at runtime. More specifically, evidence nodes $E(t)$ are clamped using sensor readings (for time t) and user commands (for time up to time t), thus impacting the status of the health nodes $H(t)$ as computed using one or more probabilistic queries. In particular, we are interested in the maximum a posteriori probability over $H(t)$ given evidence instantiation $e(t)$ for $E(t)$, or $\text{MAP}(H(t), e(t))$. This MAP query can be approximated using the most probably explanation (MPE) or the most likely values (MLV); we will use the notation $\text{MAP}_{\text{MPE}}(H(t), e(t))$ and $\text{MAP}_{\text{MLV}}(H(t), e(t))$ respectively. The benefit of these two approximations is that they are, from a complexity theory perspective, easier (roughly speaking) than the MAP query in the general case [37].

VIII. BENCHMARKING RESULTS: BAYESIAN DIAGNOSTICS OF ADAPT

We now discuss our benchmarking of the current Bayesian network model for ADAPT. (The model was developed in collaboration with Mark Chavira and Adnan Darwiche, UCLA; see also [11,12]) The ADAPT BN currently contains 503 discrete nodes and 579 edges; domain cardinalities range from 2 to 4 with an average of 2.23 and a median of 2. Note that this ADAPT BN was not created manually. Instead, it was auto-generated from a high-level specification of ADAPT. The ADAPT BN was then compiled, using the ACE system (see <http://reasoning.cs.ucla.edu/ace/>), into an arithmetic circuit. The timing measurements reported here were made on a PC with an Intel 4 1.83 GHz processor, 1 GB RAM, and Windows XP.

These experimental scenarios were generated using the ADAPT EPS. These scenarios, which are summarized in Figure 14, cover component failures, sensor failures, and both component and sensor failures. In addition, each scenario contains one, two, or three faults. In order to stress-test our probabilistic reasoner, we did not restrict inserted faults to discrete faults only. We also inserted continuous faults such as “stuck at x ”, “noise StdDev = x ” or “drift slope = x “, where x is a real number. Since our probabilistic models do not contain continuous random variables, experiments with continuous faults cannot be diagnosed exactly, but they are still of great interest and included in many of the experiments reported on below.

In each scenario, ADAPT's initial state was as follows: Circuit breakers were commanded closed; the corresponding command variables in $E_C(t)$ were clamped to *cmdClose* in evidence $e(t)$. Relays were commanded open; the corresponding relay variables in $E_C(t)$ were clamped to *cmdOpen* in $e(t)$. In the initial state, the result of computing $\text{MAP}_{\text{MPE}}(H(t), e(t))$ is that all health nodes $H(t)$ are healthy. Continuous sensor readings in $E_S(t)$ were discretized before being used for clamping the appropriate discrete random variables in our ADAPT model. To keep the experimental protocol consistent across scenarios, all inserted faults were persisted until the end of the experiments.

ID	Faults Inserted in ADAPT	Most Probable Diagnosis - Computed	Match	Num	Mean	Median	StDev
304	Relay EY260 failed open	$Health_relay_ey260_cl = stuckOpen$	Yes	226	1.086	0.534	2.479
305	Relay feedback sensor ESH175 failed open	$Health_relay_ey175_cl = stuckOpen$	Yes	145	1.057	0.543	1.802
306	Circuit breaker ISH262 tripped	$Health_breaker_ey262_op = stuckOpen$	Yes	341	0.791	0.504	0.896
308	Voltage sensor E261 failed low	$Health_e261 = stuckVoltageLo$	Yes	169	1.019	0.534	1.684
309	Battery BATT1 voltage low	$Health_battery1 = stuckDisabled$	Yes	365	0.954	0.502	3.131
310	Inverter INV1 failed off	$Health_inv1 = stuckOpen$	Yes	182	0.994	0.51	1.307
311	Light sensor LT500 failed low	$Health_lt500 = stuckLow$	Yes	158	1.099	0.545	1.64
441	Relay EY160 stuck open	$Health_relay_ey160_cl = stuckOpen$	Partly	195	0.985	0.546	1.28
	Big fan ST515 stuck at 0 RPM						
442	Current sensor IT261 noise StdDev = 5	$Health_it261 = stuckCurrentHi$	Partly	173	2.653	0.455	8.557
	Relay feedback sensor ESH172 stuck at 0	$Health_esh172 = stuckOpen$					
	Current sensor IT140 stuck at 100						
443	Current sensor IT281 drift slope = 2	$Health_it281 = stuckCurrentHi$	Partly	177	2.704	0.532	10.285
	Relay EY244 stuck closed	$Health_relay_ey244_cl = stuckClosed$					
	Big fan ST516 stuck at -10 RPM						
445	Voltage sensor E235 stuck at 0.3	$Health_e235 = stuckVoltageLo$	Partly	175	1.073	0.56	1.348
	Relay feedback sensor ESH344A stuck closed	$Health_relay_ey344_cl = stuckClosed$					
	Inverter INV2 failed off	$Health_inv2 = stuckOpen$					
447	Voltage sensor E161 failed low	$Health_e161 = stuckVoltageLo$	Yes	179	0.961	0.504	1.2
	Current sensor IT167 failed low	$Health_it167 = stuckCurrentLo$					
449	Voltage sensor E140 failed low	$Health_e140 = stuckVoltageLo$	Yes	136	1.007	0.487	1.398
	Voltage sensor E161 failed low	$Health_e161 = stuckVoltageLo$					
450	Inverter INV1 failed off	$Health_inv1 = stuckOpen$	Partly	160	0.994	0.482	1.296
	Big fan ST515 stuck at 600 RPM	$Health_fan1_speed_st515 = stuckMid$					
451	Relay EY171 failed open	$Health_relay_ey171_cl = stuckOpen$	Yes	135	1.016	0.49	1.329
	Light sensor LT500 failed low	$Health_lt500 = stuckLow$					
452	Light bulb TE500 failed off	$Health_load170_bulb1 = stuckDisabled$	Partly	166	0.739	0.358	1.282
	Temperature sensor TE501 failed low						

Fig. 14. The summary of experimental scenarios run on ADAPT testbed

In this paper, we have calculated four (of the six) isolation metrics for the Bayesian diagnostic algorithm using 16 scenarios with single and multiple faults. These metrics are: time-to-isolate, time-to-estimate, isolation classification rate, and isolation misclassification rate. These results are discussed next.

A. Event Table

For illustration purposes, we first present how data is provided in one scenario, namely Experiment 447. This experiment lasts for approximately 80 seconds and has two sequential fault injections as shown in Table 1 below. Sensors are sampled at a 2 Hz rate, and after each sample the probabilistic model is used to compute a diagnosis.

Time	Event
15:56:21.194	Start of scenario
15:56:21.236	Sample of sensors
15:56:21.736	Sample of sensors
...	
15:57:04.736	Sample of sensors
15:57:05.080	Fault injection
15:57:05.236	Sample of sensors
...	

15:57:14.736	Sample of sensors
15:57:15.080	Fault injection
15:57:15.236	Sample of sensors
...	
15:57:42.252	End of scenario

Table 1. The illustration of temporal development of a multiple fault scenario

B. State Estimation and Isolation Times

The results of the experiments with real-world data from ADAPT are summarized in Figure 14. Each scenario is presented in one or more rows of the table, along with the faults inserted and the diagnostic results computed for queries $MAP_{MPE}(\mathbf{H}(t), \mathbf{e}(t))$. Because $\mathbf{H}(t)$ contains 128 variables that provide the health status of 128 EPS components and sensors, we only show the variables found to be non-healthy in Figure 14. The main diagnostic query was also taken towards the end of a scenario.

Our main observations regarding the results from the experiments are as follows. In 10 out of the 16 scenarios, there is an exact match between the faults inserted and the diagnosis. Even in cases where there is not an exact match, the diagnosis is either partly matching or at least quite reasonable. In addition, the table provides statistics on the state estimation

Confusion Matrix for Relays		Actual state		
		non-faulty	stuckOpen	stuckClosed
Predicted state	non-faulty	0.9896	0.0000	0.0000
	stuckOpen	0.0000	0.0078	0.0000
	stuckClosed	0.0000	0.0000	0.0026

Confusion Matrix for Fans		Actual state			
		non-faulty	stuckZero	stuckMid	stuckHigh
Predicted state	non-faulty	0.9063	0.0625	0.0000	0.0000
	stuckZero	0.0000	0.0000	0.0000	0.0000
	stuckMid	0.0000	0.0000	0.0313	0.0000
	stuckHigh	0.0000	0.0000	0.0000	0.0000

Confusion Matrix for Current Sensors		Actual state				
		non-faulty	stuckLo	stuckHi	noiseVariation	drift
Predicted state	non-faulty	0.9722	0.0000	0.0069	0.0000	0.0000
	stuckLo	0.0000	0.0069	0.0000	0.0000	0.0000
	stuckHi	0.0000	0.0000	0.0000	0.0069	0.0069
	noiseVariatio	0.0000	0.0000	0.0000	0.0000	0.0000
	drift	0.0000	0.0000	0.0000	0.0000	0.0000

Fig. 15. The summary of confusion matrixes for select ADAPT components

times.

The “time-to-isolate metric” is also calculated for a range of experiments and the results are presented in Table 2. In calculating this metric, we distinguish between sequential and simultaneous fault insertion. The sequential case is more complicated, in that there are two isolation times (since two faults are inserted for all scenarios shown). Since the diagnostic inference time is on the order of one millisecond, much of the isolation time here is due to the relatively slow sample rate of 2Hz.

ID	Faults	Fault Insertion	Time to Isolate (ms)
447	2	Sequential	158.35
449	2	Simultaneous	63.85
451	2	Sequential	720.89

Table 2. The illustration of temporal development of a multiple fault scenario

C. Confusion Matrixes

To illustrate our computation of confusion matrices, we consider three components types, namely Relays (of which there are 24 in ADAPT), Fans (2 in ADAPT), and Current Sensors (9 in ADAPT). The matrices for these components types are illustrated in Figure 15. Given these matrices, we can compute isolation classification and misclassification rates. For relays, the classification rate is 1.0; for fans it is 0.9375; while for current sensors it is 0.9792. In other words, according to the classification rate, the performance is strongest for the relays, while it is weakest for the fan faults (given the current set of test cases).

IX. CONCLUSIONS

In this paper, we introduced a new architecture and a formal framework to be used for systematic benchmarking of

monitoring and diagnostic systems and for producing comparable performance assessment for different diagnostic technologies.

The framework defines a number of standardized components, which include a fault catalog, a library of modular test scenarios, and a common protocol for gathering and processing diagnostic data. In addition, it introduces 13 benchmarking metrics that are used as a basis of evaluation.

To illustrate the benchmarking framework, we considered probabilistic model-based diagnosis of an electrical power system (EPS) called ADAPT. ADAPT is a real-world electrical power system that resides at the NASA Ames Research Center. Our testing procedure is scenario-based; each scenario is nominal (non-faulty) or involves faults being injected into ADAPT. Testing proceeds in a controlled manner such that benchmarking is enabled. Finally, we discussed how diagnostic data is generated and presented results for a selected subset of our defined metrics for a probabilistic model-based diagnosis algorithm.

There are several important characteristics of the developed framework. First, it uses complex, real-world data taken from the ADAPT EPS. Second, the framework defines generic requirements and details important elements for creating a benchmarking architecture that can be used for empirical evaluation of monitoring and diagnostic systems. It emphasizes the use of a common fault catalogue and common metrics, which together enable “apples to apples” assessments of the effectiveness of different technologies. Third, the framework defines 13 analytical performance metrics that provides a systematic way to perform benchmarking of diagnostic algorithms for realistic fault scenarios. Moreover, contrary to other benchmarking examples in the literature, it enables the calculation of these metrics in multiple fault scenarios.

ACKNOWLEDGEMENT

The authors would like to thank Ann Patterson-Hine and Dougal Maclise (NASA Ames Research Center) for their role in the development of the ADAPT testbed, and David Hall, Adam Sweet, John Ossenfort, Serge Yentus, David Garcia and David Nishikawa (NASA Ames Research Center) for their contributions to the benchmarking discussions and for running many of the ADAPT experiments.

REFERENCES

- [1] R. M. Button and A. Chicatelli, "Electrical Power System Health Management", In Proc. 1st International Forum on Integrated System Health Engineering and Management in Aerospace, November 2005, Napa, CA.
- [2] S. Poll, A. Patterson-Hine, J. Camisa, D. Garcia, D. Hall, C. Lee, O. J. Mengshoel, C. Neukom, D. Nishikawa, J. Ossenfort, A. Sweet, S. Yentus, I. Roychoudhury, M. Daigle, G. Biswas, and X. Koutsoukos, "Advanced Diagnostics and Prognostics Testbed", In Proc. of the 18th International Workshop on Principles of Diagnosis (DX-07), Nashville, TN, May 2007.
- [3] F. Figueroa and J. Schmalzel, "Rocket Testing and Integrated System Health Management", In Condition Monitoring and Control for Intelligent Manufacturing, W. Gao (ed), Springer Verlag, 2006, pp. 373-392.
- [4] O. J. Mengshoel, M. Chavira, K. Cascio, S. Poll, A. Darwiche, and S. Uckun, "Probabilistic Model-Based Diagnosis: An Electrical Power System Case Study", Submitted to IEEE Transactions on Systems, Man and Cybernetics, Part A, 2008.
- [5] S. Lauritzen and D. J. Spiegelhalter, "Local Computations with Probabilities on Graphical Structures and their Application to Expert Systems (with Discussion)", Journal of the Royal Statistical Society series B, Vol. 50, No. 2, 1988, pp. 157-224.
- [6] U. Lerner, R. Parr, D. Koller, and G. Biswas, "Bayesian fault detection and diagnosis in dynamic systems", In Proc. of the Seventeenth National Conference on Artificial Intelligence (AAAI-00), 2000, pp. 531-537.
- [7] E. Liu and D. Zhang, "Diagnosis of Component Failures in Space Shuttle Main Engines using Bayesian Belief Networks: A Feasibility Study", In Proc. 14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI-02), 2002.
- [8] R. L. Bickford, T. W. Bickmore, and V. A. Caluori, "Real-Time Sensor Validation for Autonomous Flight Control", In Proc. 33rd Joint Propulsion Conference and Exhibit, Seattle, WA, July 1997.
- [9] T. W. Bickmore, "A Probabilistic Approach to Sensor Data Validation", In Proc. 28th Joint Propulsion Conference and Exhibit, Nashville, TN, July 1992.
- [10] W. A. Maul, K. J. Melcher, A. K. Chicatelli, and T. S. Sowers, "Sensor Data Qualification for Autonomous Operation of Space Systems", In AAAI Fall Symposium on Spacecraft Autonomy: Using AI to Expand Human Space Exploration, Arlington, VA, October 2006.
- [11] O. J. Mengshoel, A. Darwiche, K. Cascio, M. Chavira, S. Poll, and S. Uckun, "Diagnosing Faults in Electrical Power Systems of Spacecraft and Aircraft", In Proc. of the Twentieth Innovative Applications of Artificial Intelligence, Conference (IAAI-08), Chicago, IL, 2008.
- [12] O. J. Mengshoel, A. Darwiche, and S. Uckun, "Sensor Validation using Bayesian Networks", In Proc. of the 9th International Symposium on Artificial Intelligence, Robotics, and Automation in Space (ISAIRAS-08), Los Angeles, CA, 2008.
- [13] SAE (Society of Automotive Engineers) E-32, 2007, "Health and Usage Monitoring Metrics, Monitoring the Monitor", February 14, 2007, SAE ARP 5783-DRAFT.
- [14] Depold, H., Siegel, J., and Hull, J., 2004, "Metrics for Evaluating the Accuracy of Diagnostic Fault Detection Systems", ASME GT2004-54144, IGTI Turbo Expo, June 2004, Vienna, Austria.
- [15] Depold, H., Rajamani, R., Morrison, W.H., and Pattipati, K.R., 2006, "A Unified Metric for Fault Detection and Isolation in Engines", ASME GT2006-91095, IGTI Turbo Expo, May 2006, Barcelona, Spain.
- [16] Metz, C.E., 1978, "Basic Principles of ROC Analysis", Journal of Nuclear medicine, 1978, Vol 8 (4), pp. 283-298.
- [17] Orsagh R.F., Roemer, M.J., Savage, C.J., and Lebold, M., 2002, "Development of Performance and Effectiveness Metrics for Gas Turbine Diagnostic Techniques", Aerospace 2002 IEEE Conference Proceedings, 2002, Vol6, pp. 2825-2834.
- [18] Roemer, M.J., Dzakovic, J., Orsagh R.F., Byington, C.S., and Vachtsevanos, G., 2004, "Validation and Verification of Prognostic Health Management Technologies", Aerospace 2005 IEEE Conference Proceedings, 2005, paper 1344.
- [19] Bartys, M., Patton, R., Syfert, m., de las Heras, S., and Quevedo, J., 2006, "Introduction to the DAMADICS Actuator FDI Benchmark Study", Control Engineering Practice, 2006, Vol 14, pp.577-596.
- [20] Williams, Z., 2006, "Benefits of IVHM: An Analytical Approach", Proceedings of IEEE Aerospace Conference, Big Sky, Montana, 2006.
- [21] J. Kurien, and Maria Dolores R-Moreno, 2008, "Costs and Benefits of Model-Based Diagnosis", Aerospace 2008 IEEE Conference Proceedings, 2008, Paper #1280.
- [22] Hoyle, C., Mehr, A.F., Tumer, I.Y., and Chen, W., 2007, "Cost-benefit analysis of ISHM in aerospace systems," International Design Engineering Technical Conferences; Computers in Engineering Conference (IDETC/CIE), Las Vegas, NV.
- [23] Narasimhan, S., Dearden, R., and Benazera, E., "Combining Particle Filters and Consistency-based Approaches for Monitoring and Diagnosis of Stochastic Hybrid Systems," 15th International Workshop on Principles of Diagnosis (DX04), Carcassonne, France, 2004.
- [24] D. Iverson, 2004, "Inductive System Health Monitoring", Proceedings of the 2004 International Conference on Artificial Intelligence (IC-AI'04), Las Vegas, Nevada, June 2004.
- [25] M. Daigle, Roychoudhury, I., Biswas G., Koutsoukos, X., Patterson-Hine, A., Poll, S., 2008, "A Comprehensive Diagnosis Methodology for Complex Hybrid Systems: A Case Study on Spacecraft Power Distribution Systems", IEEE Transactions on Systems, Man, and Cybernetics, Part B, 2008, Volume 38 (2).
- [26] Ghoshal, S., Azam, M., and Malepati, V., SBIR Phase III "Comprehensive Fault Detection, Isolation, and Recovery (FDIR) on the ADAPT Test Bed," Progress Report 1, Contract No. NNA06AA51Z, Oct. 2006.
- [27] Vesely, W. E., Goldberg, F. F., Roberts, N. H. and Haasi, D. F., The Fault Tree Handbook, US Nuclear Regulatory Commission, NUREG 0492, 1981.
- [28] S. Poll, A. Patterson-Hine, J. Camisa, D. Nishikawa, L. Spirkovska, Garcia, D. Hall, C. Lee, O. J. Mengshoel, C. Neukom, J. Ossenfort, A. Sweet, S. Yentus, I. Roychoudhury, M. Daigle, G. Biswas, and X. Koutsoukos, 2007, "Evaluation, Selection, and Application of Model-Based Diagnosis Tools and Approaches", In Proc. of the AIAA Infotech at Aerospace Conference and Exhibit, Rohnert Park, CA, May 2007.
- [29] Andrej Rakar, and Juricic, D., 2004, "Matching the Requirements in Model-Based Fault Diagnosis", 2004, Proceeding of the 15. International Workshop on Principles of Diagnosis (DX-04), Carcassonne, France, June 2004.
- [30] J. Pearl, "Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference", Morgan Kaufmann, 1988.
- [31] O. J. Mengshoel, "Designing Resource-Bounded Reasoners using Bayesian Networks: System Health Monitoring and Diagnosis", In Proc. of the 18th International Workshop on Principles of Diagnosis (DX-07), Nashville, TN, May 2007.
- [32] M. Chavira and A. Darwiche, "Compiling Bayesian Networks Using Variable Elimination", In Proc. of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07), January 2007, pp. 2443 - 2449.
- [33] M. Chavira, M. and A. Darwiche, "Compiling Bayesian Networks with Local Structure", In Proc. of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05), 2005, 1306-1312.
- [34] O. J. Mengshoel, "Macroscopic Models of Clique Tree Growth for Bayesian Networks". In Proc. of the 22nd National Conference on Artificial Intelligence (AAAI-07). July 2007, Vancouver, Canada, pp. 1256-1262.
- [35] O. J. Mengshoel, D. C. Wilkins, and D. Roth, "Controlled Generation of Hard and Easy Bayesian Networks: Impact on Maximal Clique Tree in Tree Clustering". Artificial Intelligence, 170(16-17), October 2006, pp. 1137-1174.

- [36] A. Darwiche, "A Differential Approach to Inference in Bayesian Networks", Journal of the ACM, Volume 50, Number 3, pp. 280-305, 2003.
- [37] J. D. Park and A. Darwiche, "Complexity Results and Approximation Strategies for MAP Explanations", Journal of Artificial Intelligence Research (JAIR), Vol. 21, 2004, pp. 101-133.