A Framework for Systematic Benchmarking of Monitoring and Diagnostic Systems

Tolga Kurtoglu, Ole J. 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 assessments of 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 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 Diagnostics and Prognostics Testbed (ADAPT) developed and located at the NASA Ames Research Center. The proposed benchmarking approach shows how to generate realistic diagnostic data sets for 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 and monitoring 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 performance to SHM at NASA.

An important element of SHM systems is monitoring and diagnostic algorithms, which are emphasized in this paper [3-11]. In recent years, a wide range of algorithms, 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 highlight the following two in this paper: the *data challenge* and the *evaluation challenge*.

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 and monitoring technologies (and in particular model-based approaches). This makes it difficult to empirically evaluate integrated systems where software systems 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 alternative technologies, 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 of system health management systems. We consider different 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 [4,11]. The diagnosed system is the Advanced Diagnostics and Prognostics Testbed (ADAPT) [2], a real-world electrical power system (EPS). 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.

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 problem sets for diagnosis and monitoring, including sensor

Manuscript received May 19, 2008. This work was supported in part by the National Aeronautics and Space Administration (NASA) Aeronautics Research Mission Directorate (ARMD) Aviation Safety Program (AvSP) Integrated Vehicle Management (IVHM) Project. Additionally, this material is based upon the work supported by NASA under awards NCC2-1426 and NNA07BB97C.

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).

validation [8,9,10,12]. Second, by introducing a benchmarking architecture and uniform metrics, we enable systematic empirical evaluation of different diagnostic algorithms, which again should lead to improved understanding and comparative analysis of different diagnostic algorithms.

An immediate advantage of our presented method is that it provides a formal framework that can be utilized to generate benchmarking results for a wide spectrum of diagnostic approaches independent of their specific modeling and reasoning schemes (probabilistic/deterministic, model-based/rule-based/ data driven, on-line/off-line, time varying/temporal, etc). Such benchmarking results can eventually be used to select between or integrate different diagnostic approaches in a wide variety of applications. Moreover, our method defines and describes the main elements of the framework so that the benchmarking results can be applied to any arbitrary testbed by constructing the architecture described in the paper.

II. RELATED WORK

The development of monitoring and diagnostic technologies is of great interest to military and commercial 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 clearly and explicitly 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 's "Health and Usage Monitoring Metrics" [13] defines probability of detection and probability of false alarms as key indices 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. In essence, the Kappa Coefficient measures the ability of an algorithm to discriminate among many 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 includes 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-offs in health management systems such as operational cost, scalability, and maintainability 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 study, 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 uses real process data, and demonstrates how the performance indices can be calculated for 19 actuator faults using a single fault assumption.

The benchmarking methodology presented here adopts some of its metrics from [13, 17-19] and extends prior work in this area by defining a number of novel benchmarking indices, by providing a generic, application independent architecture that can be used for benchmarking different monitoring and diagnostic algorithms, and by facilitating the use of real process data on a large-scale, complex engineering system. Moreover, it is not restricted to single fault assumption and enables the calculation of benchmarking metrics for systems, where each component may have multiple fault modes.

III. BENCHMARKING FRAMEWORK AND ADAPT TESTBED 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 introduced including a standardized fault catalog, a library of modular and standardized test scenarios, a test protocol, a common process for processing diagnostic data, and common metrics.

The physical system is the Advanced Diagnostics and

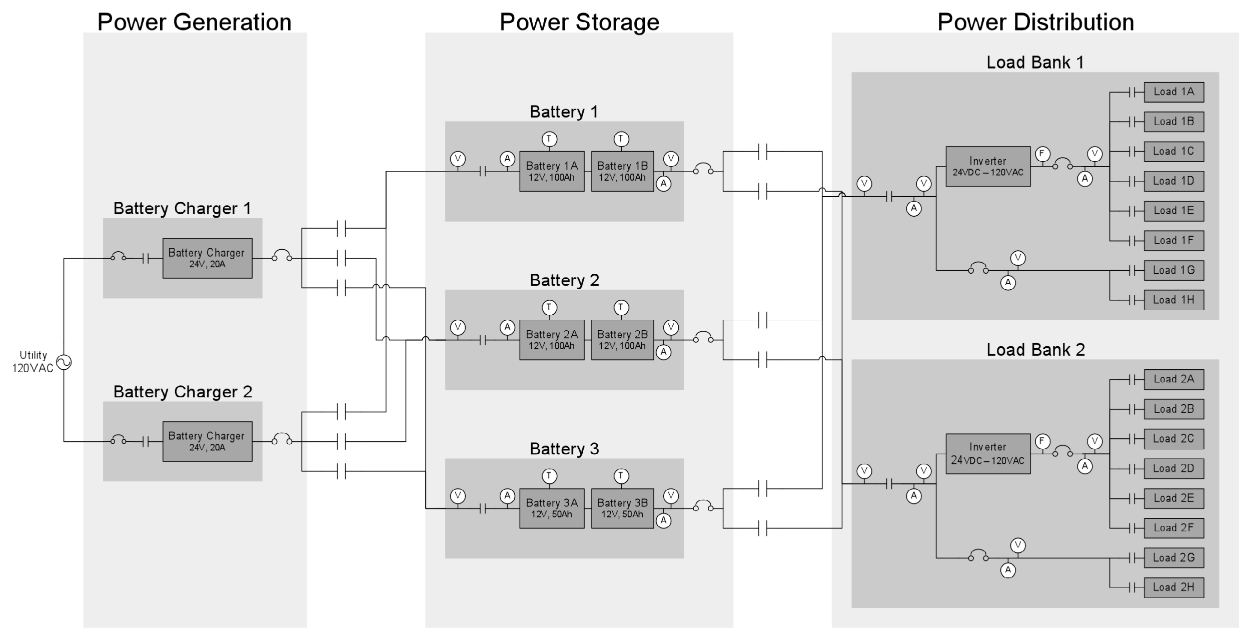


Fig. 1. ADAPT Testbed components and the system configuration

Prognostics Testbed (ADAPT) developed and located at the NASA Ames Research Center. The test-bed is a unique facility designed to test, measure, evaluate, and mature diagnostic and prognostic health management technologies. Reflecting the importance of electrical power systems (EPS) in aerospace [1,2], ADAPT provides a representative EPS subsystem that enables automated diagnosis in a complex domain. The main functions and layout of the ADAPT power system is shown in Fig. 1. ADAPT contains 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 the two battery chargers in the power generation element. Any of the three batteries can be used to power any of the two load banks in the power distribution element. (Loads 1A through Loads 1H in Load Bank 1, and Loads2A through Loads 2H in Load Bank 2). This design gives the ADAPT EPS basic redundancy and reconfiguration capability.

The EPS problem domain exhibits both discrete and continuous fault behaviors, which are defined under a standardized fault catalog. The fault catalog establishes a common ground as to what failure modes and fault behaviors are required to be modeled by diagnostic algorithms. In addition to the probabilistic system discussed in Section VI, ADAPT has worked with diagnostic algorithms from NASA (Hybrid Diagnostic Engine (HyDE) [23], Inductive Monitoring System (IMS) [24], academia (FACT) [25], and industry (TEAMS-RT) [26].

ADAPT's data acquisition and control system sends commands to and receives data from the EPS. 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, a standardized outputting scheme is enforced on the diagnostic algorithms to ensure the generation of common data sets for the calculation of metrics.

The testing procedure, shown in Figure 2, is scenario-based, where each scenario may have faults injected into the system through a scenario loader. To detect faults, each diagnostic algorithm has access to data from the plant (in this case the ADAPT EPS) to generate diagnostic output in real-time. Alternatively, scenarios can be recorded into a database and processed by the algorithms at a later time. In both cases, the data from the plant and the output of the diagnostic system are recorded into a *scenario results file*, and the diagnostic algorithm performance is evaluated using this file according to a predefined set of metrics.

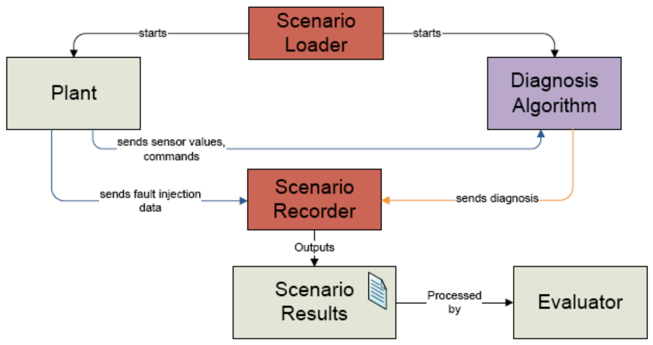


Fig. 2. The test protocol of the benchmarking framework

IV. 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, 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.

Third, a repeatable, standardized process should be developed for running 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. In particular, there needs to be 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 rest of this section followed by a detailed discussion of the performance metrics in Section V.

A. Fault Catalog

Different fault types and behaviors embodied within the ADAPT testbed are defined in 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 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.

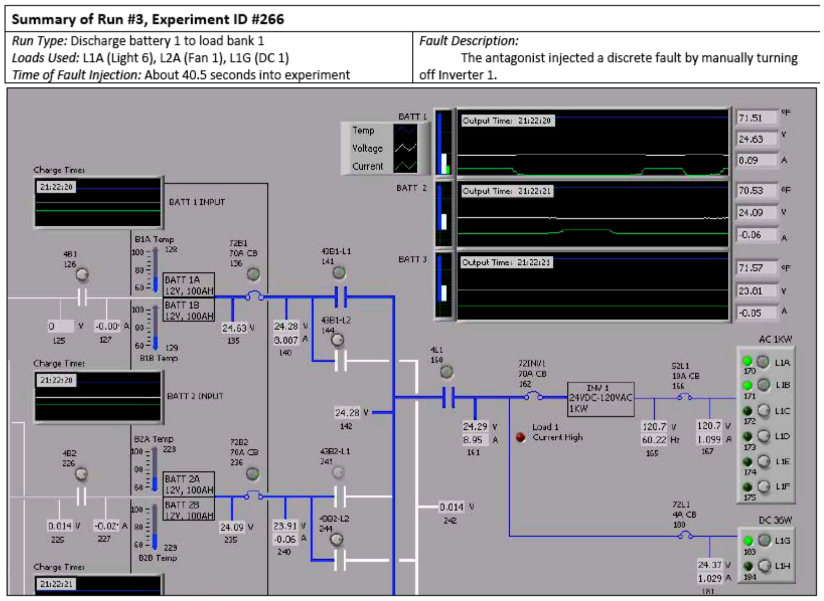
The fault catalog lists a "unique ID" for each fault type, and organizes faulty behavior under generic component types. For example, faults 1-6 in the catalog capture "battery" faults, whereas, faults 7-10 define "circuit breaker" faults. The physical locations where these faults could manifest themselves are also listed in the catalog under each generic component type followed by the class (abrupt vs. incipient) of each fault. Currently, the fault catalog is implemented as an XML document and includes over 90 faults classified under 14 generic component types. In addition, each component type has its own non-faulty or healthy mode(s) defined.

The fault catalog along with the non-faulty component mode definitions serves as a requirements document for 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 the basic connectivity of the major elements of the EPS testbed, i.e. the connectivity between power storage, power generation, and power distribution elements, and the initial state of the system components such as circuit breakers, relays, etc. For example, Battery 1 may be connected to Load Bank 1, 2, or both. Similarly, Battery 1 may be charged by either of the battery chargers. The *load configuration* describes the loads that are used during a test run. Currently, the ADAPT EPS can be configured to have up to 6 AC and 2 DC loads on 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 inserted. For each fault, the fault configuration includes a unique fault-id, injection time and injection location of the fault, any required



Summary of Run #3, Experiment ID #266

Run Type: Discharge battery 1 to load bank 1 *Loads Used:* L1A (Light 6), L2A (Fan 1), L1G (DC 1) *Time of Fault Injection:* About 40.5 seconds into experiment	*Fault Description:* The antagonist injected a discrete fault by manually turning off Inverter 1.

Fig. 3. An illustration of running a test scenario

fault parameters, and the source of the fault that captures whether the fault was generated by the simulation or was induced through a hardware manipulation.

Fig. 3 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 loads are 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 manipulation. 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 Fig. 3.

C. Testing Procedures, Inputs, and Outputs

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 inputs and outputs given to and generated by the ADAPT testbed and the monitoring and diagnostic software for each test scenario.

In order to provide consistency between the diagnostic algorithms, we offer a simple reference architecture shown in Fig. 4. According to this architecture, each diagnostic algorithm is treated as a black box with inputs and outputs as shown in the figure. In this architecture, a diagnostic system is characterized by its model $\mathbf{M_i}$ of the ADAPT testbed, as well as how the inputs to and outputs from 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 testbed from the ADAPT user or other command generating agent. These represent 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 algorithm's output is an estimate of ADAPT's health status $\mathbf{h_i(t)}$, which typically includes the health of ADAPT's sensors. For simplicity, when we only consider a particular algorithm, we use the notation **h(t)** hereafter.

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:

· **Detection Signal** $\mathbf{DS_i(t)}$: A binary value (high or low) as to whether the diagnostic system has detected a fault.

· **Isolation Signal** $\mathbf{IS_i(t)}$: 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.

· **Number of Candidates** $\mathbf{NC_i(t)}$: A candidate is a list of faults, i.e. estimated system components that are diagnosed as "faulty". Each fault is a *string* in the form of (Component_ID. Component_Mode) such as (EY166.StuckClosed). 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. A candidate may have an empty list or may include multiple faults. In the latter case, the listed faults are assumed to be present in the system concurrently. The $\mathbf{NC_i(t)}$ is simply an integer corresponding to the number of candidates an algorithm reports as part of a diagnosis.

· **Candidate Fault Set** $\mathbf{CFS_i(t)}$: A candidate fault set is a list of candidates an algorithm reports as a diagnosis. For example, a candidate fault set may include two candidates each with a single fault such as [{EY166.StuckClosed)}, {(FanA2.failedOverspeed)}]. It is assumed that only one candidate in a candidate fault set can represent the system at any given time.

· **Candidate Probabilities** $\mathbf{CP_i(t)}$: Each candidate in the candidate fault set is required to have an associated probability of occurrence. If there is a single candidate in the CFS, then there will be a single probability equaling to 1. In all other cases, individual candidate probabilities should sum up to 1.

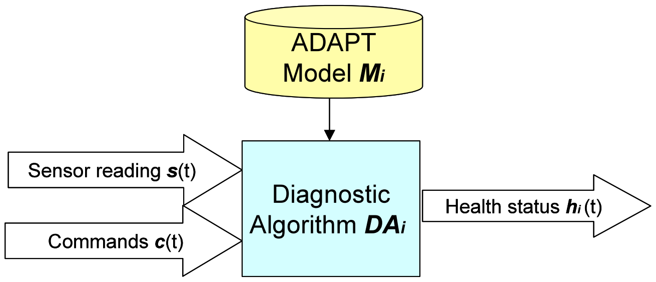


Fig. 4. Diagnostic algorithm reference architecture. Each diagnostic algorithm, DA_i is given a stream of sensory readings and commands. The output from each diagnostic algorithm is a stream of estimates of the health status of the ADAPT testbed.

This output by the diagnostic algorithms can be generated synchronously based on the sampling 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. Currently, there are diagnostic algorithms deployed on the ADAPT testbed that generate outputs using both of the aforementioned methods.

D. Fault Injection

ADAPT supports the repeatable injection of faults into the system in two ways [28]. First, faults may be physically injected into the testbed hardware. A simple example is tripping a circuit breaker using the manual throw bars. Another example 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, a 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. (This is different than the VIRTUAL ADAPT simulation, which we discuss below.) Software fault injection includes one or more of the following: 1) sending commands that were not intended for nominal operations; 2) blocking commands sent to the testbed; and 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. It is also possible to inject intermittent data faults, data spikes, and to combine more than one fault type for a given sensor at the same time. By using these approaches to software fault injection, fault scenarios may be constructed that represent diverse sensor faults.

Since some fault scenarios may be costly, dangerous, or impossible to introduce in the actual ADAPT 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.

V. BENCHMARKING METRIC DEFINITIONS

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

First, the metrics are categorized as either *detection metrics* or *isolation metrics* as shown in Table 1. In defining this categorization, 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.

Table 1. A summary of metrics used for the benchmarking activity

	"Detection" Metrics
	Temporal Performance
	Response Time
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
	Response Time
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***

To produce reliable and realistic benchmarking results for isolation functionality, it is required that a consistent level is 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 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 for the algorithms 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.

The second categorization scheme used for metric definitions is based on the distinction between the temporal and non-temporal aspects of diagnostics. Accordingly, the metrics that measure time response of diagnostic algorithms are grouped under *temporal metrics* and those that measure non-temporal features of a diagnostic algorithm including accuracy, resolution, sensitivity, and stability [18] are grouped under *static metrics*. These metrics are shown in Table 1.

The *temporal metrics*, one for detection (metric 1) and two for isolation (metric 8-9), 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 Fig. 5. In the figure, t_{inj} is the time of fault injection, t_{dsig} is the time of first high detection signal.

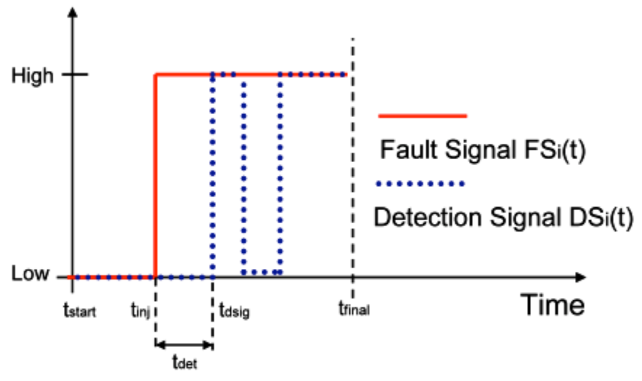


Fig. 5. The definition of "time-to-detect" metric

Time to detect t_{det} then becomes: $t_{det} = t_{dsig} - t_{inj}$, where $t_{det} > 0$. For simplicity, we have assumed both the fault and the detection signal to have values of "high" and "low". (Fig. 5 illustrates the profile of an abrupt fault that is persistent after its injection.)

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

Fig. 6. The decision matrix

Metrics 2-5 are *detection accuracy metrics* and are defined based on the construction of a decision matrix [13]. 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 Fig. 6.

The diagonal in the decision matrix are the correct predictions. The faulty cases equal to **b+d**, and the non-faulty 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 intervals {[0,0.25],

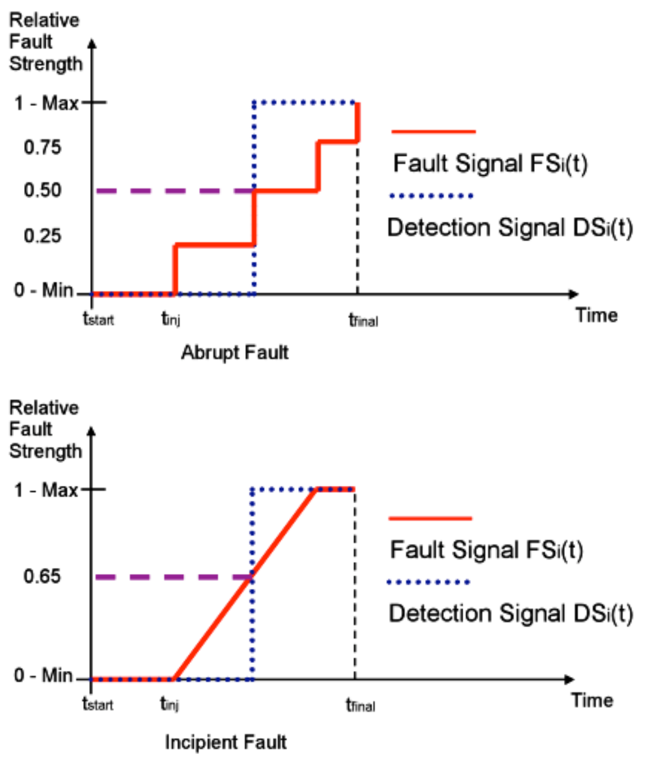


Fig. 7. The definition of "detection-sensitivity-factor" metric

[0.25,0.50], [0.50-0.75], and [0.75,1.00]} (where applicable), 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 Fig. 7.

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

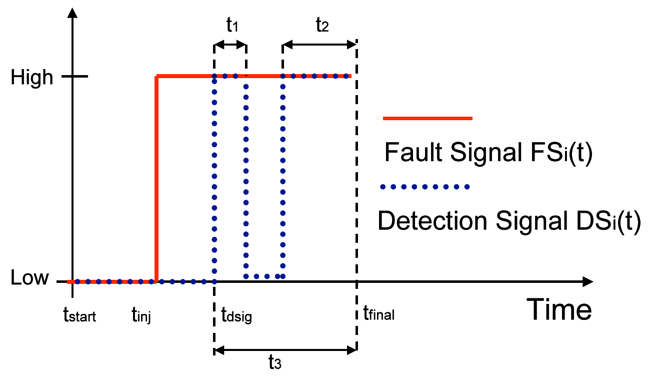


Fig. 8. The definition of "detection-stability-factor" metric

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 from the first "high" detection signal to the end time of the experiment (t_{final} in the figure). For example, if the detection signal remains "high" at all times after the initial detection, the stability factor equals to 1. Otherwise, it takes a value between (0,1) depending on the proportional time of "high" signal versus "low" signal. The metric is illustrated in Fig. 8. For the case shown in the figure, the detection stability factor equals to $\mathbf{(t_1 + t_2)/t_3}$.

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 computation time becomes available as shown in Fig. 9.

In the figure, t_{isig1} and t_{isig2} are the time of isolation signals corresponding to candidate fault sets CFS_1 and CFS_2. In the example given, the algorithm does not have any diagnosis (i.e. empty candidate fault set) until $t=t_{isig1}$ when the algorithm isolates to "inverter1 switched off" or {(Inv 1, f10)} using the standardized diagnostic output format. This remains to be the estimate of the algorithm until $t=t_{isig2}$ when the isolation estimate of the algorithm changes to "circuit_braker_166 failed open" or {(EY166, f7)}. The health estimate of the algorithm then remains at CFS_2 until the end of the experiment.

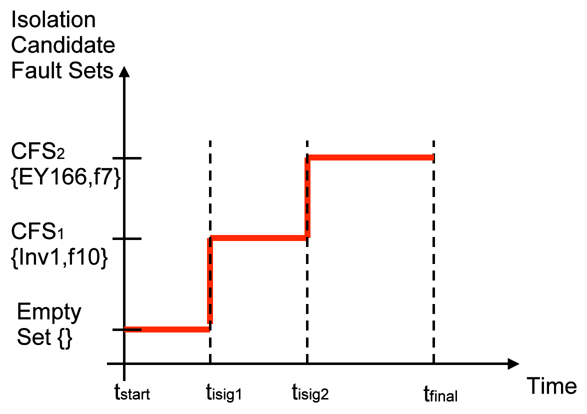


Fig. 9. The definition of isolation convergence estimate

Although isolation estimates are time variant as illustrated in Fig.9, our isolation metric definitions require a *unique* final diagnostic output *for each scenario* to be used for the calculation of the metrics. Defining what constitutes this final diagnostic output may not always be straightforward. Consider the case shown in Fig. 9 again. What is the final diagnostic estimate of the algorithm for the particular experiment shown? Is it CFS_1 or CFS_2? To address this, we define the concept of *isolation convergence set* which will be treated as the final diagnostic output of an algorithm. For simplicity, *an isolation convergence set* is defined as the candidate fault set that corresponds to the latest isolation estimate by the algorithm. Based on this definition, CFS_2 becomes the isolation convergence set for the case used in Fig. 9. Of course, one may take a different approach in defining a unique isolation output. Currently, we are investigating other definitions that take the actual fault injections and other temporal aspects (such as duration) of 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 beginning of the last "high" isolation signal that corresponds to the isolation convergence set. This is shown in Fig. 10. In the figure, t_{inj} is the time of fault injection, and t_{isig1} and t_{isig2} are the time of isolation signals corresponding to CFS_1 or CFS_2 that are previously introduced in Fig. 9. Since CFS_2 is the isolation convergence set, the time to isolate t_{iso} metric is calculated based on the signal that corresponds to CFS_2. The time-to-isolate metric then becomes: $t_{iso} = t_{isig2} - t_{inj}$, where $t_{iso} > 0$.

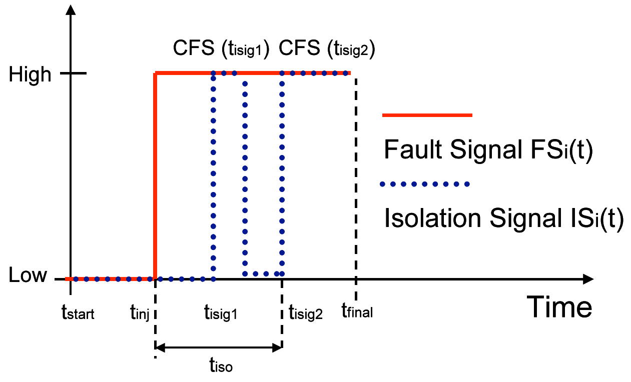


Fig. 10. The definition of "time-to-isolate" metric

Metric 9 - Time to Estimate: The time spent by an algorithm to estimate the physical state of the system. This metric is calculated by summing over and averaging the time periods from the reading of sensor values and commands to the moment of producing an health status estimate as shown in Fig. 4.

Metrics 10-11 are *isolation accuracy metrics* and are defined based on the construction of a confusion matrix [13, 6]. A confusion matrix is an expanded version of a decision matrix that incorporates fault classification as shown in Fig. 11 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.

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

Fig. 11. The confusion matrix

Similar to the decision matrix, the diagonal values in the 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 an estimate of the probability of that case occurring. In addition, the non-faulty row and column can be removed from the matrix to ensure that the matrix represents a measure of discrimination between faults once it has been determined that a fault is actually present. For example, the Kappa Coefficient [13][1], is calculated using a confusion matrix without the no-fault entries.

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 are then defined as:

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

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

These metrics are calculated based on the *isolation convergence set* generated by the algorithm for each scenario. Accordingly, a confusion matrix is built for each system component and the cumulative isolation classification rate of a diagnostic algorithm is calculated by employing a weighted sum over isolation classification rates of individual system components. If there are multiple candidates in an isolation convergence set, the aforementioned calculations are repeated for each candidate and a second weighted sum is taken based on the candidate probabilities as reported by $\mathbf{CP_i(t)}$.

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

Metric 12 – Size of Isolation Set: The number of faults in a candidate that belongs to the isolation convergence set. If there are multiple candidates in an isolation convergence set, an average is taken.

[1] Not used in this study

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 the time elapsed when the isolation signal corresponding to the "isolation convergence set" was "high" (t_2 in Fig. 12) to the time elapsed from the first "high" isolation signal to the end time of the experiment (t_3 in Fig. 12). The metric is illustrated in Fig. 12. For the case shown in the figure, the isolation stability factor equals to $\mathbf{t_2/t_3}$.

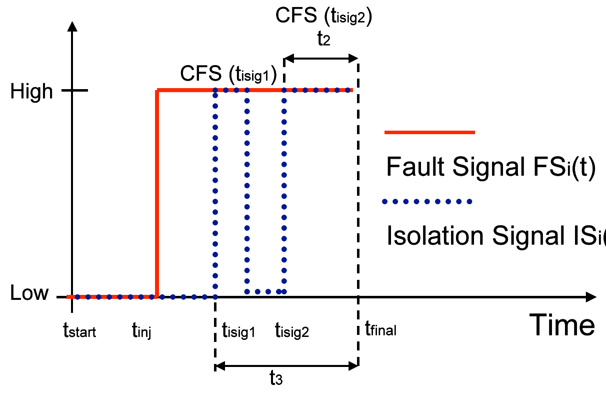


Fig. 12. The definition of "isolation-stability-factor" metric

C. Metric Calculation under Multiple Fault Scenarios

One novelty of the proposed benchmarking approach is its ability to calculate the aforementioned metrics when there are multiple faults in a scenario.

The time to detect metric (metric 1) under multiple scenarios is calculated based on the first fault injected. The second temporal metric, time to isolate (metric 8), is calculated by averaging individual isolation times over the number of faults injected. The third temporal metric, time-to-estimate (metric 9), is generic and calculated as defined earlier independent of the number of faults injected. For detection accuracy metrics (metrics 2-5), multiple fault cases are treated as one and the scenario is classified and logged to the decision matrix accordingly. For isolation accuracy metrics (metrics 10-11), the multiple fault cases are treated separately and logged to the confusion matrices as individual entries. The detection sensitivity metric (metric 6) is applicable to single faults only. To calculate the detection and isolation stability metrics (metrics 7 and 13) under multiple fault scenarios, individual fault signal profiles are first superimposed and the metrics are then calculated as previously defined. Finally, the size of isolation set (metric 12), is generic and calculated as defined earlier independent of the number of faults injected.

VI. CASE STUDY: PROBABILISTIC DIAGNOSTICS

As a case study, we consider a probabilistic approach to diagnosis and state estimation [11]; it is based on Bayesian

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. 13. The summary of experimental scenarios run. "Match" summarizes whether or not the fault(s) inserted into ADAPT exactly corresponds to the computed diagnosis. "Num" gives the number of samples used to compute the following statistics (Mean. Median, and StDev (ms)) for the time to estimate metric.

networks [30] and arithmetic circuits [32,33,36]. 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 (see also [4,11,12]).

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 arithmetic circuit approach [32,33].

Under the tree clustering paradigm, a Bayesian network is transformed into join tree [5]. 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: $\boldsymbol{H_C(t)}$: Represent the health of a system (such as ADAPT) excluding sensors, both failure modes and operational (nominal) modes.
    - Sensor health nodes $\boldsymbol{H_S(t)}$: 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 $\boldsymbol{E_C(t)}$: 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 $\boldsymbol{E_S(t)}$: Sensor readings – such as voltage, current, and temperature for ADAPT. Because of sensor failure, some sensor readings might be incorrect.
- Remaining nodes $\boldsymbol{R(t)}$: Nodes that reflect parts of the system that are not represented by $\boldsymbol{H(t)}$ *or* $\boldsymbol{E(t)}$.

Information from sensors and the environment (user) is incorporated into the probabilistic reasoning process at runtime. More specifically, evidence nodes $\boldsymbol{E(t)}$ 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 $\boldsymbol{H(t)}$ as computed using one or more probabilistic queries. In particular, we are interested in the maximum a posteriori probability over $\boldsymbol{H(t)}$ given evidence instantiation $\boldsymbol{e(t)}$ for $\boldsymbol{E(t)}$, or MAP($\boldsymbol{H(t)}$, $\boldsymbol{e(t)}$). This MAP query can be approximated using the most probably explanation (MPE) or the most likely values (MLV) [11,12]; we will use the notation MAP_{MPE}($\boldsymbol{H(t)}$, $\boldsymbol{e(t)}$) and MAP_{MLV}($\boldsymbol{H(t)}$, $\boldsymbol{e(t)}$) 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].

VII. 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 [4,11,12]) 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 [4]. The ADAPT BN was then compiled, using the ACE system (see http://reasoning.cs.ucla.edu/ace/), into an arithmetic circuit, which was then used for diagnosis. 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 Fig. 13, cover 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 "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 $\boldsymbol{E_C(t)}$ were clamped to *cmdClose* in evidence $\boldsymbol{e(t)}$. Relays were commanded open; the corresponding relay variables in $\boldsymbol{E_C(t)}$ were clamped to *cmdOpen* in $\boldsymbol{e(t)}$. In the initial state, the result of computing MAP_{MPE}($\boldsymbol{H(t)}$, $\boldsymbol{e(t)}$) is that all health nodes $\boldsymbol{H(t)}$ are healthy. Continuous sensor readings in $\boldsymbol{E_S(t)}$ 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 persisted until the end of the experiments.

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.

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 2 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

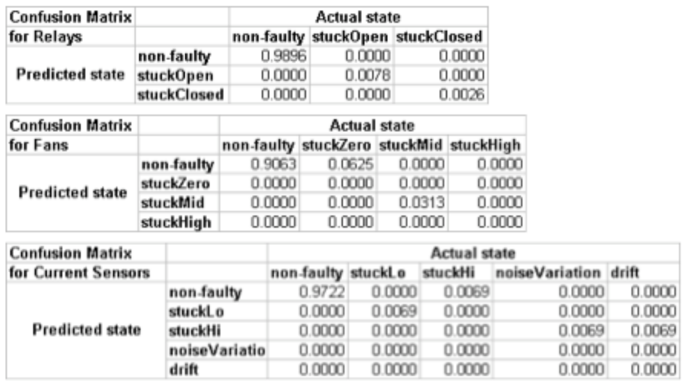
Table 2. The illustration of temporal development of a multiple fault scenario

B. State Estimation and Isolation Times

The results of the experiments with real-world data from ADAPT are summarized in Fig. 13. 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}($\boldsymbol{H(t)}$, $\boldsymbol{e(t)}$). Because $\boldsymbol{H(t)}$ 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 Fig. 13. Diagnostic inference times are generally around 1 ms, making this approach suitable for real-time settings.

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 times (Metric 9).

The "time-to-isolate metric" (Metric 8) is also calculated for



Confusion Matrix		Actual state		
for Relays		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		Actual state			
for Fans		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		Actual state				
for Current Sensors		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. 14. The summary of confusion matrixes for select ADAPT components

a range of experiments and example results are presented in Table 3. 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, giving 500 ms between samples.

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

Table 3. The illustration of temporal development of a multiple fault scenario

C. *Confusion Matrices*

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 Fig. 14. Given these matrices, we can compute isolation classification (Metric 10) and misclassification rates (Metric 11). 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 fans (given the current set of test cases). Interestingly, the accuracy of isolation does not diminish with increasing the number of modes of a component. As can be seen from the results, the classification rate is better for current sensors when compared to fans.

VIII. CONCLUSIONS

In this paper, we introduced a new architecture and a formal framework to be used for systematic benchmarking of monitoring and diagnostic systems. 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 uses 13 benchmarking metrics as a basis of evaluation. These metrics enable the production of comparable performance assessment for different diagnostic technologies.

To illustrate the benchmarking framework, we considered an electrical power system (EPS) called ADAPT [4,11]. 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. 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, which defines 13 analytical performance metrics, 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 when an individual scenario has multiple faults.

There are also several assumptions that the presented framework is based upon. These assumptions pose certain limitations that are left to be addressed in future research. First, the true test of diagnostics is in robustness of operation in the target environment. It is the variability in the target environment that brings about the majority of the non-obvious false alarms. The ADAPT Lab as configured offer some capability that mirror target environments, but there is still variability in every hardware laboratory, which will eventually show up in diagnostic models and the results generated. Second, the current implementation does not account for the likelihood of different failure scenarios. In the 16 runs presented, for example, it is implicitly assumed that all component failure modes for a given component have the same prior probability of occurrence, which may not be the case. Currently, we are working on adding fault probabilities to the fault catalog in order to facilitate better modeling of the failure modes of the system. Moreover, we are working on expanding the scope of the fault types included in the ADAPT fault catalog. Accordingly, we are in the process of adding new continuous faults as well as introducing intermittent fault types. Third, the current isolation metrics evaluate diagnostic performance based on a discrete isolation assumption in which "faults" are isolated to one of the discrete modes of a component defined by the fault catalog. As more continuous type faults are introduced, additional or generalized metrics are required in order to calculate the accuracy of isolation estimates on a continuous scale. Finally, our future work aims at augmenting this work with Internet accessible data files that would enable testing of various approaches and strategies in a web-based, distributed fashion.

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, Eric Barszcz, and Sriram Narasimhan (NASA Ames Research Center) for their contributions to the benchmarking discussions.

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 IEEEE 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., Dzakowic, 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.