Benchmarking Diagnostic Algorithms on an Electrical Power System Testbed

Tolga Kurtoglu [1], Sriram Narasimhan [2], Scott Poll [3], David Garcia [4], Stephanie Wright [5]

[1] *Mission Critical Technologies @ NASA Ames Research Center, Moffett Field, CA, 94035, USA*
tolga.kurtoglu@nasa.gov
[2] *University of California, Santa Cruz @ NASA Ames Research Center, Moffett Field, CA, 94035, USA*
Sriram.Narasimhan-1@nasa.gov
[3] *NASA Ames Research Center, Moffett Field, CA, 94035, USA*
scott.poll@nasa.gov
[4] *Stinger Ghaffarian Technologies @ NASA Ames Research Center, Moffett Field, CA, 94035, USA*
david.garcia@nasa.gov
[5] *Vanderbilt University, Nashville, TN, 37203, USA*
stephanie.l.wright@vanderbilt.edu

ABSTRACT

Diagnostic algorithms (DAs) are key to enabling automated health management. These algorithms are designed to detect and isolate anomalies of either a component or the whole system based on observations received from sensors. In recent years a wide range of algorithms, both model-based and data-driven, have been developed to increase autonomy and improve system reliability and affordability. However, the lack of support to perform systematic benchmarking of these algorithms continues to create barriers for effective development and deployment of diagnostic technologies. In this paper, we present our efforts to benchmark a set of DAs on a common platform using a framework that was developed to evaluate and compare various performance metrics for diagnostic technologies. The diagnosed system is an electrical power system, namely the Advanced Diagnostics and Prognostics Testbed (ADAPT) developed and located at the NASA Ames Research Center. The paper presents the fundamentals of the benchmarking framework, the ADAPT system, description of faults and data sets, the metrics used for evaluation, and an in-depth analysis of benchmarking results obtained from testing ten diagnostic algorithms on the ADAPT electrical power system testbed.

1 INTRODUCTION

Fault Diagnosis in physical systems involves the detection of anomalous system behavior and the identification of its cause. Key steps in the diagnostic inference are fault detection (is the output of the system incorrect?), fault isolation (what is broken in the system?), fault identification (what is the magnitude of the failure?), and fault recovery (how can the system continue to operate in the presence of the faults?). Expert knowledge and prior know-how about the system, models describing the behavior of the system, and sensor data from the system during actual operation are used to develop diagnostic inference algorithms. This problem is non-trivial for a variety of reasons including:

- Incorrect and/or insufficient knowledge about system behavior
- Limited observability
- Presence of many different types of faults (system/supervisor/actuator/sensor faults, additive/multiplicative faults, abrupt/incipient faults, persistent/intermittent faults)
- Non-local and delayed effect of faults due to dynamic nature of system behavior
- Presence of other phenomena that influence/mask the symptoms of faults (unknown inputs acting on system, noise that affects the output of sensors, etc.)

Several communities have attempted to solve the diagnostic inference problem using various methods. Some typical approaches have been:

- Expert Systems - These approaches encode knowledge about system behavior into a form that can be used for inference. Some examples are rule-based systems (Kostelezky *et al.*, 1990) and fault trees (Kavcic and Juricic, 1997).
- Model-based Systems - These approaches use an explicit model of the system configuration and behavior to guide the diagnostic inference. Some examples are "FDI" methods (Gertler and Inc.,

1998), statistical methods (Basseville and Nikiforov, 1993), "AI" methods (Hamscher *et al.*, 1992).

- Data-driven Systems - These approaches use only the data from representative runs to learn parameters that can then be used for anomaly detection or diagnostic inference for future runs. Some examples are IMS (Iverson, 2004), Neural Networks (Sorsa and Koivo, 1998), etc.
- Stochastic Method - These approaches treat the diagnosis problem as a belief state estimation problem. Some examples are Bayesian Networks (Lerner *et al.*, 2000), Particle Filters (de Freitas, 2001), etc.

Despite the development of such a variety of notations, techniques, and algorithms, efforts to evaluate and compare the different diagnostic algorithms (DA) have been minimal (discussed in Section 2). One of the major deterrents is the lack of a common framework for evaluating and comparing diagnostic algorithms. Such a framework would consist of the following:

- Define a standard representation format for the system description, sensor data, and diagnosis results
- Develop a software run-time architecture that can run specific scenarios from actual system, simulation, or other data sources such as files (individually or as a batch), execute DAs, send scenario data to the DA at appropriate time steps, and archive the diagnostic results from the DA
- Define a set of metrics to be computed based on the comparison of the actual scenario and diagnosis results from the DA

Some initial steps in developing such a framework are presented in (Kurtoglu *et al.*, 2009b) as part of the DX Competition initiative (Kurtoglu *et al.*, 2009a). In this paper, we present our efforts to benchmark a set of DAs on the NASA Ames Electrical Power System testbed (ADAPT) using the DXC framework. Section 2 describes other work related to benchmarking of DAs. Section 3 presents the DXC framework in brief. Section 4 describes how the benchmarking was performed including a description of the ADAPT system, the faults injected, the DAs tested and the metrics computed. Section 5 presents the results and detailed analyses of the benchmarking activity. Section 6 lists the limitations and plans for future work. Lastly, section 7 presents the conclusions.

2 RELATED WORK

Several researchers have attempted to demonstrate benchmarking capability on different systems. Among these, (Orsagh *et al.*, 2002) provided a set of 14 metrics to measure the performance and effectiveness of prognostics and health management algorithms for US Navy applications (Roemer *et al.*, 2005). (Bartys *et al.*, 2006) 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 of FDI algorithms on an industrial valve-actuator system. Izadi-Zamanabadi and Blanke (1999) presented a ship propulsion system as a benchmark for autonomous fault control. This benchmark has two main elements. One is the development of an FDI algorithm, and the other is the analysis and implementation of autonomous fault accommodation. Finally, (Simon *et al.*, 2008) introduced a benchmarking technique for gas path diagnosis methods to assess the performance of engine health management technologies.

The approach presented in this paper uses the DXC Framework (Kurtoglu *et al.*, 2009b) (described in Section 3) which adopts some of its metrics from the literature (Society of Automotive Engineers, 2007; Orsagh *et al.*, 2002; Simon *et al.*, 2008) and extends prior work in this area by 1) defining a number of new benchmarking indices, 2) providing a generic, application-independent architecture that can be used for benchmarking different monitoring and diagnostic algorithms, and 3) facilitating the use of real process data on large-scale, complex engineering systems. Moreover, it is not restricted to a single fault assumption and enables the calculation of benchmarking metrics for systems in which each fault scenario may contain multiple faults.

3 FRAMEWORK OVERVIEW

The framework architecture employed for benchmarking diagnostic algorithms is shown in Figure 1. Major elements are the physical system (ADAPT EPS testbed), diagnostic algorithms, scenario-based experiments, and benchmarking software. The physical system description and sample data (nominal and faulty) are provided to algorithm and model developers to build DAs. System documentation in XML format specifies the components, connections, and high-level mode behavior descriptions, including failure modes. A diagram with component labels and connection information is also provided. The documentation defines component and mode identifiers DAs must report in their diagnoses for proper benchmarking. The fault catalog, part of the system documentation, establishes the failure modes that may be injected into experimental test scenarios and diagnosed by the DAs. Benchmarking software is used to quantitatively evaluate the DA output against the known fault injections using predefined metrics.

Execution and evaluation of diagnostic algorithms are accomplished with the run-time architecture depicted in Figure 2. The architecture has been designed to interface DAs with little overhead by using minimalistic C++ and Java APIs or by implementing a simple ASCII-based TCP messaging protocol. The Scenario Loader is the main entry point for DAs; it starts

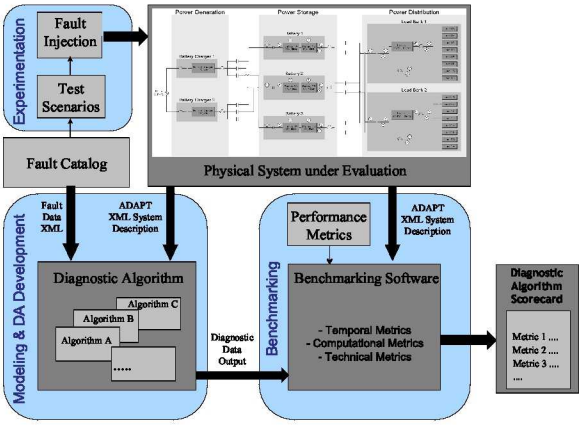

Figure 1: Benchmarking Framework Architecture

and stops other processes and cleans up upon completion of all scenarios. The Scenario Data Source publishes archived datasets containing commands and sensor values following a wall-clock schedule specified by timestamps in the scenario files. The DA uses the diagnostic framework messaging interface to receive sensor and command data, perform a diagnosis, and publish the results; it is the only component that is implemented by DA developers. The Scenario Recorder timestamps each fault injection and diagnosis message upon arrival and compiles it into a Scenario Results file. The Evaluator takes the Scenario Results File and calculates metrics to evaluate DA performance.

Messages in the run-time architecture are exchanged as ASCII text over TCP/IP. DAs may use provided API calls for parsing, sending, and receiving messages or choose to use the underlying TCP/IP. The DA output is standardized to facilitate benchmarking and includes a timestamp indicating when the diagnosis has been issued; a detection signal that indicates whether the DA has detected a fault; an isolation signal that indicates whether a DA has isolated a candidate or set of candidates with associated probabilities; and a candidate fault set that has one, or multiple candidates – each with a single or multiple faults. More details about the framework can be found in (Kurtoglu *et al.*, 2009b).

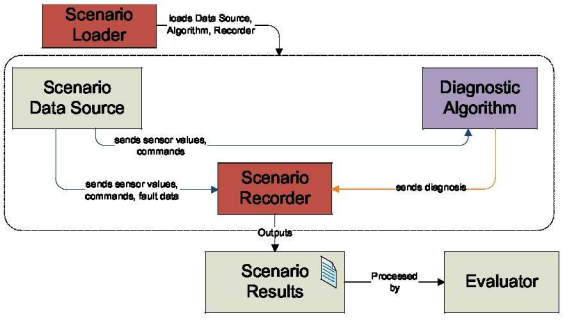

Figure 2: Run-time Architecture

4 BENCHMARKING ON ADAPT EPS TESTBED

4.1 ADAPT EPS

System Description

The physical system used for benchmarking is the Electrical Power System testbed in the ADAPT lab at NASA Ames Research Center (Poll *et al.*, 2007). The ADAPT EPS testbed provides a means for evaluating diagnostic algorithms through the controlled insertion of faults in repeatable failure scenarios. The EPS testbed incorporates low-cost commercial off-the-shelf (COTS) components connected in a system topology that provides the functions typical of aerospace vehicle electrical power systems: energy conversion /generation (battery chargers), energy storage (three sets of lead-acid batteries), power distribution (two inverters, several relays, circuit breakers, and loads) and power management (command, control, and data acquisition). The EPS delivers AC (Alternating Current) and DC (Direct Current) power to loads, which in an aerospace vehicle could include subsystems such as the avionics, propulsion, life support, environmental controls, and science payloads. A data acquisition and control system commands the testbed into different configurations and records data from sensors that measure system variables such as voltages, currents, temperatures, and switch positions. Data is presently acquired at a 2 Hz rate.

The scope of the ADAPT EPS testbed used in the benchmarking study is shown Figure 3. Power storage and distribution elements from the batteries to the loads are within scope; there are no power generation elements. In order to encourage more DA developers to participate in the benchmarking effort, we also included a simplified scope of the system called ADAPT-Lite, which included a single battery to a single load as indicated by the dashed lines in the schematic (Figure 3). The characteristics of ADAPT-Lite and ADAPT are summarized in Table 1. The greatest simplification of ADAPT-Lite relative to ADAPT is not the reduced size of the domain but the elimination of nominal mode transitions. The starting configuration for ADAPT-Lite data has all relays and circuit breakers closed and no nominal mode changes are commanded during the scenarios. Hence, any noticeable changes in sensor values may be correctly attributed to faults injected into the scenarios. By contrast, the initial configuration for ADAPT data has all relays open and nominal mode changes are commanded during the scenarios. The commanded configuration changes result in adjustments to sensor values as well as transients which are nominal and not indicative of injected faults. Finally, ADAPT-Lite is restricted to single faults whereas multiple faults are allowed in ADAPT.

Fault Injection and Scenarios

ADAPT supports the repeatable injection of faults into the system in three ways:

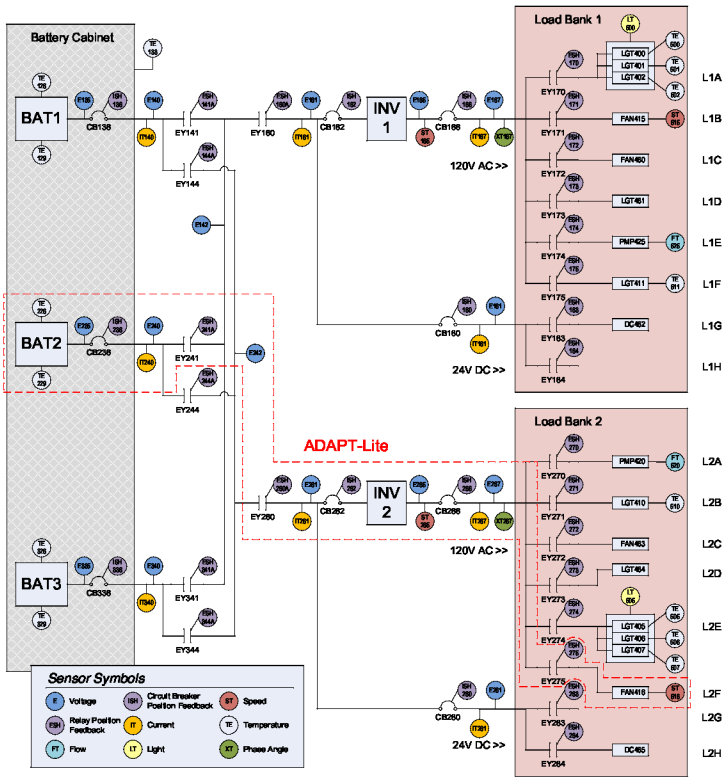

Figure 3: ADAPT EPS Schematic

Table 1: ADAPT EPS Benchmarking System Characteristics

Aspect	**ADAPT-Lite**	**ADAPT**
#Comps/Modes	37/93	173/430
Initial State	Relays closed; circuit breakers closed	Relays open; circuit breakers closed
Nominal mode changes?	No	Yes
Multiple faults?	No	Yes

Hardware-Induced Faults: These faults are 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. 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.

Software-Induced Faults: 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; and 3) altering the testbed sensor data.

Real Faults: In addition the aforementioned two methods, real faults may be injected into the system by using actual faulty components. A simple example includes a blown light bulb. This method of fault injection was not used in this study.

For the results presented in this paper, only abrupt discrete (change in operating mode of component) and parametric (step change in parameter values) faults are considered. Distinct faults types that are injected into the testbed for benchmarking are shown Table 2.

The diagnostic algorithms are tested against a number of diagnostic scenarios consisting of nominal or faulty data of approximately four minutes in length. Some key points and intervals of a notional scenario are illustrated in Figure 4, which splits the scenario into three important time intervals: $\Delta_{startup}$, $\Delta_{injection}$, and $\Delta_{shutdown}$. During the first interval $\Delta_{startup}$, the diagnostic algorithm is given time to initialize, read data files, etc. Note that this is not the time for compilation; compilation-based algorithms compile their models beforehand. Though sensor observations may be available during $\Delta_{startup}$, no faults are injected during this time. Fault injection takes place during $\Delta_{injection}$. Once faults are in-

Table 2: Fault types used for ADAPT and ADAPT-Lite

Component	Fault Description
Battery	Degraded
Boolean Sensor	Stuck at Value
Circuit Breaker	Failed Open Stuck Closed
Inverter	Failed Off
Relay	Stuck Open Stuck Closed
Sensor	Stuck at Value Offset
Pump(Load)	Flow Blocked Failed Off
Fan(Load)	Over Speed Under Speed Failed Off
Light Bulb(Load)	Failed Off
Basic Load	Failed Off

jected they persist until the end of the scenario. Multiple faults may be injected simultaneously or sequentially. Finally, the algorithms are given some time post-injection to send final diagnoses and gracefully terminate during $\Delta_{shutdown}$. The intervals used in this study are 30 seconds, 3 minutes, and 30 seconds for $\Delta_{startup}$, $\Delta_{injection}$, and $\Delta_{shutdown}$, respectively.

Below are some notable points for the example diagnostic scenario from Figure 2:

- t_{inj} - A fault is injected at this time;
- t_{fd} - The diagnostic algorithm has detected a fault;
- t_{ffi} - The diagnostic algorithm has isolated a fault for the first time;
- t_{fir} - The diagnostic algorithm has modified its isolation assumption;
- t_{lfi} - This is the last fault isolation during $\Delta_{injection}$.

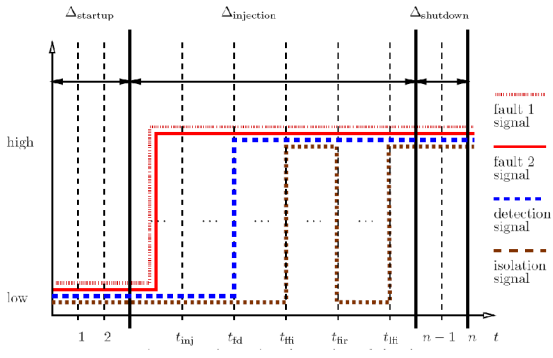

Figure 4: Key time points, intervals, and signals

Nominal and failure scenarios were created using hardware and software-induced fault injection methods according to the intervals specified above and using the faults listed in the fault catalog (Table 2). As shown in Table 3, nominal scenarios comprise roughly half of the ADAPT-Lite and one-third of the ADAPT test scenarios. The ADAPT-Lite fault scenarios are limited to single faults. Half of the ADAPT faults scenarios are single faults; the others are double or triple faults.

Table 3: Number of sample and test scenarios for ADAPT and ADAPT-Lite

	Sample		Competition	
#Scenarios	ADAPT -Lite	ADAPT	ADAPT -Lite	ADAPT
Nominal	32	39	30	40
Single-fault	27	54	32	40
Double-fault	0	19	0	30
Triple-fault	0	1	0	10

Diagnostic Challenges

The ADAPT EPS testbed offers a number of challenges to DAs. It is a hybrid system with multiple modes of operation due to switching elements such as relays and circuit breakers. There are continuous dynamics within the operating modes and components from multiple physical domains, including electrical, mechanical, and hydraulic. It is possible to inject multiple faults into the system. Furthermore, timing considerations and transient behavior must be taken into account when designing DAs. For example, when power is input to the inverter there is a delay of a few seconds before power is available at the output. For some loads, there is a large current transient when the device is turned on. System voltages and currents depend on the loads attached, and noise in sensor data increases as more loads are activated. Measurement noise occasionally exhibits spikes and is non-Gaussian. The 2 Hz sample rate limits the types of features that may be extracted from measurements. Finally, there may be insufficient information and data to estimate parameters of dynamic models in certain modeling paradigms.

4.2 Diagnostic Metrics

A set of 8 metrics has been defined for assessing the performance of the diagnostic algorithms. This set is categorized as temporal, technical, or computational performance metrics. The temporal metrics measure how quickly an algorithm responds to faults in a physical system. The technical metrics measure non-temporal features of a diagnostic algorithm like accuracy. Finally, computational metrics are intended to measure how efficiently an algorithm uses the available computational resources.

In addition, we divide the metrics into 2 main categories:

Detection metrics which deal with temporal, technical, and computational metrics associated with only detection of the fault.

Isolation metrics which deal with temporal, technical, and computational metrics associated with isolation of the fault.

The notation used for the definition of the metrics is as follows:

S - The set of scenarios for a given system description

S_n - The set of nominal scenarios for a given system description

S_f - The set of faulty scenarios for a given system description

t_{fd} - The time when the fault detection signal has been asserted for the first time

t_{fi} - The time when the last persistent fault isolation signal has been asserted

ω_{act} - The true component mode vector (ground truth)

ω_{pre} - The predicted component mode vector (represents the set of candidate diagnoses by the DA)

T_d - Total computation time

M_d - Peak amount of allocated memory

Finally, using the aforementioned notation, the 8 metrics are defined as:

Fault Detection Time (M_{fd})**:** The reaction time for a diagnostic engine in detecting an anomaly (Kurtoglu *et al.*, 2008)

$$M_{\text{fd}} = t_{\text{fd}} \tag{1}$$

Fault Isolation Time (M_{fi})**:** The time for isolating a fault (Kurtoglu *et al.*, 2008). In many applications this metric is less important than the diagnostic accuracy, but it is important in sequential diagnosis, probing, etc.

$$M_{\text{fi}} = t_{\text{fi}} \tag{2}$$

False Positive Rate (M_{fp})**:** The metric that penalizes diagnostic algorithms which announce spurious faults (Kurtoglu *et al.*, 2008). The false positive rate is defined as:

$$M_{\text{fp}} = \frac{\sum_{s \in S} m_{\text{fp}}(s)}{|S|} \tag{3}$$

where for each scenario s the "false positive" function $m_{\text{fp}}(s)$ is defined as:

$$m_{\text{fp}}(s) = \begin{cases} 1, & \text{if } t_{fd} < t_{inj} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where $t_{inj} = \infty$ for nominal scenarios.

False Negative Rate (M_{fn})**:** The metric that measures the ratio of missed faults by a diagnostic algorithm (Kurtoglu *et al.*, 2008).

$$M_{\text{fn}} = \frac{\sum_{s \in S} m_{\text{fn}}(s)}{|S_f|} \tag{5}$$

where for each scenario s the "false negative" function $m_{\text{fn}}(s)$ is defined as:

$$m_{\text{fn}}(s) = \begin{cases} 1, & \text{if } t_{\text{fd}} = \infty \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

Detection Accuracy (M_{FDA})**:** The fault detection accuracy is the ratio of number of correctly classified cases to the total number of cases (Kurtoglu *et al.*, 2008). It is defined as:

$$M_{\text{FDA}} = 1 - \frac{\sum_{s \in S} m_{\text{fp}}(s) + m_{\text{fn}}(s)}{|S|} \tag{7}$$

Classification Errors (M_{ia})**:** Isolation classification error metric measures the accuracy of the fault isolation by a diagnostic algorithm and is defined as the Hamming distance (The Hamming distance between two strings of data values is the number of positions for which the corresponding data values are different) between the true component mode vector ω_{act} and the predicted component mode vector ω_{pre}.

In the calculation of the classification error metric, the data values for the Hamming distance are the respective modes of components comprising a system description. For example, if the true component mode vector of the system is [1,0,0,1,0] and the predicted component mode vector is [1,1,0,0,0], the classification error is 2. If more than one predicted mode vector is reported by a DA, (meaning that the diagnostic output consists of a set of candidate diagnoses), then the classification error is calculated for each predicted component mode vector and weighted by the candidate probabilities reported by the DA.

CPU Load (M_{cpu})**:** This is the average CPU load during the experiment

$$M_{\text{cpu}} = t_s + \sum_{q \in T_d} q \tag{8}$$

where t_s is the startup time of the diagnostic engine and T_d is a vector with the actual CPU time spent by the diagnostic algorithm at every time step in the diagnostic session.

Memory Load (M_{mem})**:** This is the maximum memory size at every step in the diagnostic session. CPU load during the experiment

$$M_{\text{mem}} = \max_{m \in M_d} m \tag{9}$$

where M_d is a vector with the maximum memory size at every step in the diagnostic session.

4.3 Diagnostic Algorithms

A total of ten DAs were tested, nine in full ADAPT and six in ADAPT-Lite. Brief descriptions of each of these algorithms are provided below.

1. FACT - a model-based diagnosis system that uses hybrid bond graphs, and models derived from them, at all levels of diagnosis, including fault detection, isolation, and identification. Faults are detected using an observer-based approach with statistical techniques for robust detection. Faults are isolated by matching qualitative deviations caused by fault transients to those predicted by the model. For systems with few operating configurations, fault isolation is implemented in a compiled form to improve performance (Roychoudhury *et al.*, 2009).
2. Fault Buster - is based on a combination of multivariate statistical methods, for the generation of residuals. Once the detection has been done a neural network performs classification for doing isolation.
3. HyDE-A - HyDE (Hybrid Diagnosis Engine) is a model-based diagnosis engine that uses consistency between model predictions and observations to generate conflicts which in turn drive the search for new fault candidates. HyDE-A uses discrete models of the system and a discretization of the sensor observations for diagnosis (Narasimhan and Brownston, 2007).
4. HyDE-S - uses the HyDE system but runs it on interval values hybrid models and the raw sensor data (Narasimhan and Brownston, 2007).
5. ProADAPT - processes all incoming environment data (observations from a system being diagnosed), and acts as a gateway to a probabilistic inference engine. It uses the Arithmetic Circuit (AC) Evaluator which is compiled from Bayesian network models. The primary advantage to using ACs is speed, which is key in resource bounded environments (Mengshoel, 2007).
6. RacerX - is a detection-only algorithm which detects a percentage change in individual filtered sensor values to raise a fault detection flag.
7. RODON - is based on the principles of the General Diagnostic Engine (GDE) as described by de Kleer and Williams and the G+DE by Heller and Struss. RODON uses contradictions (conflicts) between the simulated and the observed behavior to generate hypotheses about possible causes for the observed behavior. If the model contains failure modes besides the nominal behavior, these can be used to verify the hypotheses, which speed up the diagnostic process and improve the results (Karin *et al.*, 2006).
8. RulesRule - is a rule-based isolation-only algorithm. The rule base was developed by analyzing the sample data and determining characteristic features of fault. There is no explicit fault detection though isolation implicitly means that a fault has been detected.
9. StanfordDA - is an optimization-based approach to estimating fault states in a DC power system. The model includes faults changing the circuit topology along with sensor faults. The approach can be considered as a relaxation of the mixed estimation problem. We develop a linear model of the circuit and pose a convex problem for estimating the faults and other hidden states. A sparse fault vector solution is computed by using l1 regularization (Zymnis *et al.*, 2009).
10. Wizards of Oz - is a consistency-based algorithm. The model of the system completely defines the stable (static) output of the system in case of normal and faulty behavior. Given a new command or new observations, the algorithm waits for a stable state and computes the minimum diagnoses consistent with the observations and the previous diagnoses.

5 RESULTS AND DISCUSSION

The benchmarking results for ADAPT-Lite and ADAPT are shown in Tables 4 and 5, respectively. Figures 5 - 8 are graphical depictions of data in Table 4 and Figures 13 - 16 are graphical depictions of data in Table 5. No DA dominated over all metrics used for benchmarking. For ADAPT-Lite, seven of the nine algorithms tested were best or second best with respect to at least one of the metrics. For ADAPT, five of six algorithms fit this description.

We discuss the ADAPT-Lite results next. Figure 5 shows the false positive rate, false negative rate, and detection accuracy. As is evident from the definition of the metrics in section 4.2, an algorithm that has lower false positive and negative rates will have higher detection accuracy. False positives were counted in the following two situations: for nominal scenarios where the DA declared a fault; and for faulty scenarios where the DA declared a fault before any fault was injected. An error in the rule base of RulesRule led to more false positive indications for the faulty scenarios than for the nominal scenarios and also resulted in a large number of classification errors. Other false positives were due to noise in the data. Figure 9 shows a nominal run in which a current sensor exhibited a spike at approximately 73 seconds. Seven out of nine DAs issued a

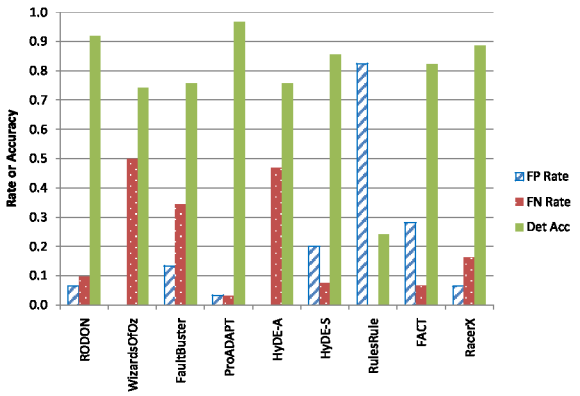

Figure 5: ADAPT-Lite false positive rate, false negative rate, and detection accuracy by DA.

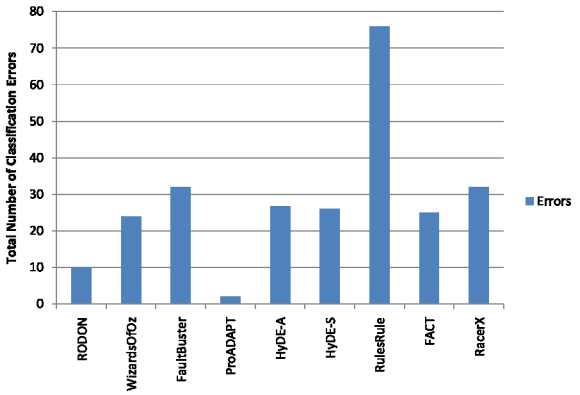

Figure 6: ADAPT-Lite classification errors by DA.

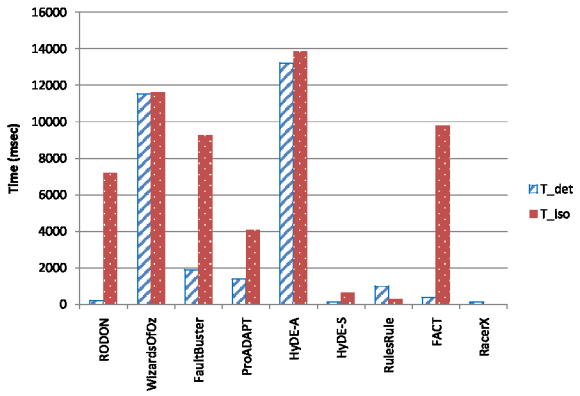

Figure 7: ADAPT-Lite detection and isolation times by DA.

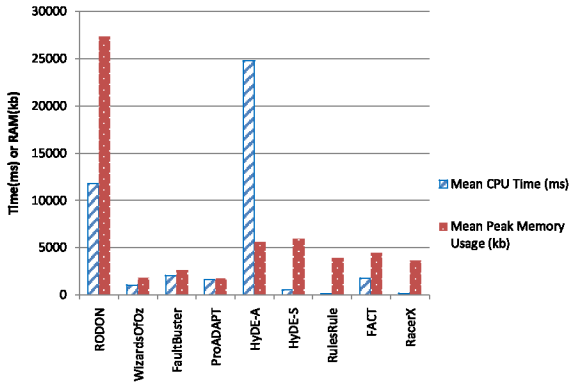

Figure 8: ADAPT-Lite CPU time and peak memory usage by DA.

false positive for this run, only HyDE-A and Wizards of Oz did not declare a fault.

Many false negatives were caused by scenarios in which a sensor reading was stuck within the nominal range of the sensor. Figure 10 shows a voltage sensor that freezes its reading at 118 seconds. Only ProADAPT correctly detected this fault.

The number of classification errors for each DA is shown in Figure 6. ProADAPT was the only DA to correctly detect and classify sensor-stuck faults of the type shown in Figure 10. Furthermore, aside from one run, it also correctly distinguished between sensor-stuck and sensor-offset faults, examples of which are shown in Figure 11. The distinction in the fault behavior is that stuck has zero noise while offset has the noise of the original signal. Also, the sensor-stuck faults were set to the minimum or maximum value of the sensor or held at its last reading. However, we did not specify to DA developers that the stuck values would be limited to these cases. Figure 12 shows number of faults injected and aggregate classification errors for all DAs by component/fault type. The category 'Sensor' includes faults in current, voltage, temperature, phase angle, and switch position sensors. Because of this grouping, the figure shows more sensor faults injected than other types. The number of classification errors in sensor fault scenarios is quite high. Part of the reason is due to the fact that most DAs reported either stuck or offset consistently or they reported both with equal weight. In this benchmarking study, no partial credit was given for correctly naming the failed component but incorrectly isolating the failure mode. We realize however, that isolating to a failed component or line-replaceable-unit (LRU) in maintenance operations is sometimes all that is required. We plan to revisit this metric in future benchmarking work.

In several instances DAs reported component mode IDs which did not match the names specified in the system catalog. For these cases the diagnosis was treated as an empty candidate. This could either negatively or positively impact the classification error metric depending on whether the DA had a correct or incorrect isolation.

The detection and isolation times are shown in Figure 7. RacerX did not have an isolation time as it was a detection-only DA. Second, note the some-

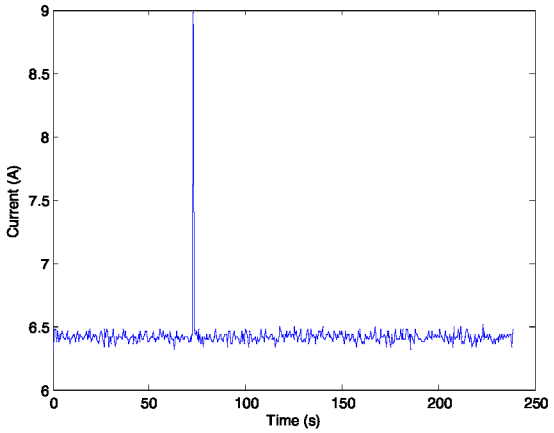

Figure 9: A nominal run with spike in sensor IT240, battery 2 current.

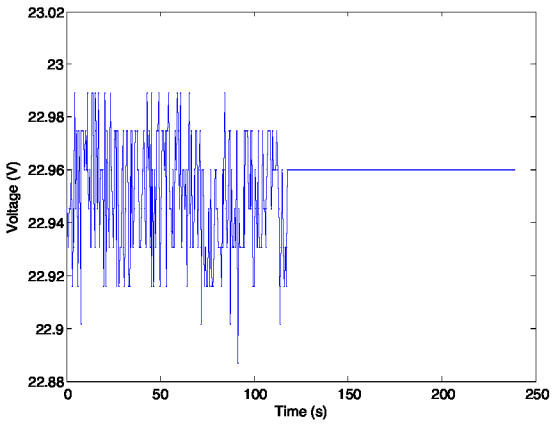

Figure 10: An example of sensor-stuck failure mode for voltage sensor E261, the downstream voltage of relay EY260.

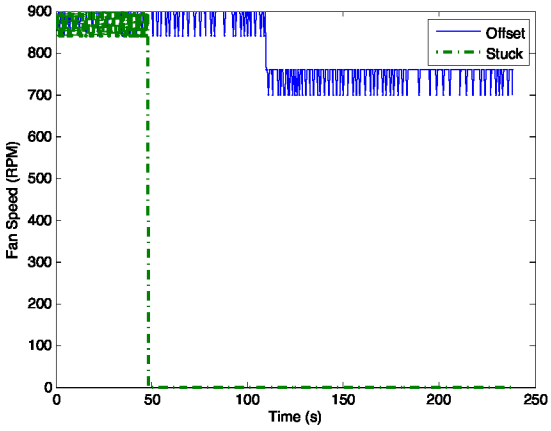

Figure 11: Fan speed sensor ST516 with sensor-offset and sensor-stuck faults.

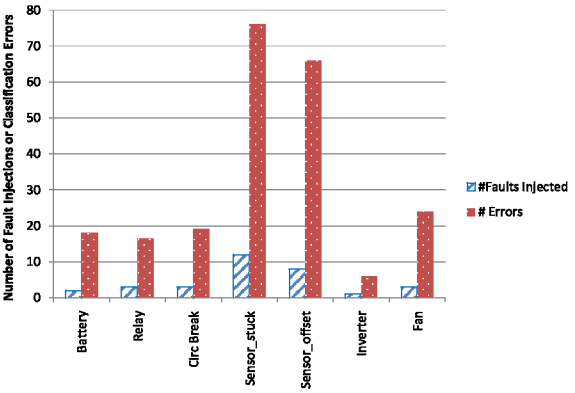

Figure 12: ADAPT-Lite classification errors by component/fault type.

what confusing result that the mean isolation time for RulesRule was less than the mean detection time. This has to do with the way the metrics are calculated. The detection time is undefined for scenarios with a false positive; however, the isolation time is not necessarily undefined and is calculated as discussed in section 4.2. The intent is to account for the situation where a DA retracts a spurious detection signal and subsequently isolates to the correct component. In this case the scenario is declared a false positive but the accuracy and timing of the isolation are calculated with respect to the last persistent diagnosis. Consequently, for DAs with many false positives the detection time may be calculated for fewer scenarios than the isolation time with the result that the mean isolation time for all scenarios could be less than the mean detection time. However, in any scenario where both times are defined, the DA isolation time is always greater than or equal to the detection time, as would be expected.

Note that the same DA was implemented by two different modelers for ADAPT-Lite. HyDE-A was modeled primarily with the larger and more complex ADAPT in mind and had a policy of waiting for transients to settle before requesting a diagnosis. The same policy was applied to ADAPT-Lite as well, even though transients in ADAPT-Lite corresponded strictly to fault events; this prevented false positives in ADAPT but negatively impacted the timing metric in ADAPT-Lite. On the other hand, HyDE-S was modeled only for ADAPT-Lite and did not include a lengthy time-out period for transients to settle. HyDE-S had dramatically smaller mean detection and isolation times (Figure 7) with roughly the same number of classification errors (Figure 6) as HyDE-A. This illustrates the impact that modeling and implementation decisions have on DA performance. While this gives some insight into trade-offs present in building models, in this work we did not define metrics that directly address the ease or difficulty of building models of sufficient fidelity for the diagnosis task at hand.

Figure 8 shows the CPU and memory usage. Note that significant differences were evident in the

peak memory usage metric when run on Linux versus Windows™. The cause for this was not explored due to time constraints, as the method used on Windows for calculating peak memory usage involved a Windows™ API system call, the analysis of which was deemed too expensive. RODON was the only Java DA that was run on Windows, which adversely affected its memory usage metric.

We now discuss the benchmarking results for the larger ADAPT system. Figure 13 shows the false positive rate, false negative rate, and detection accuracy.

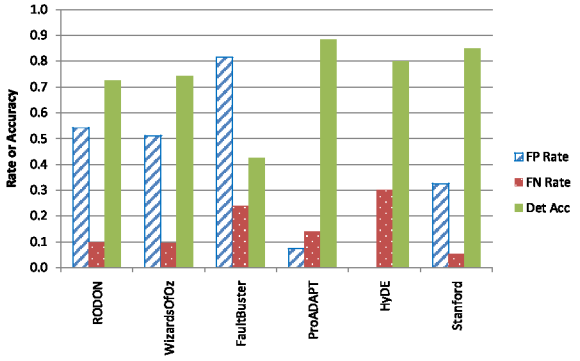

Figure 13: ADAPT false positive rate, false negative rate, and detection accuracy by DA.

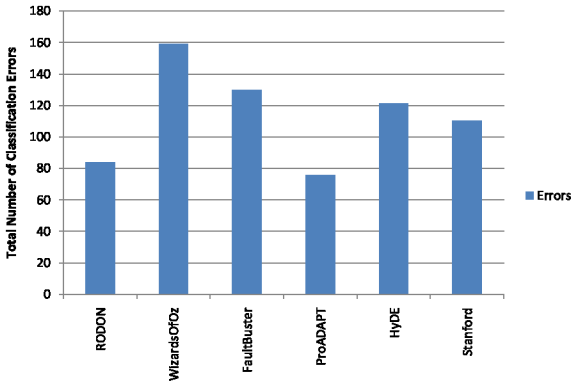

Figure 14: ADAPT classification errors by DA.

The comments in the ADAPT-Lite discussion about noise and sensor stuck apply here as well. Additionally, false positives also resulted from nominal commanded mode changes in which the relay feedback did not change status as of the next data sample after the command. Here is an extract from one of the input scenario files that illustrates this situation:

```
command @120950 EY275_CL = false;
sensors @121001 {...ESH275 = true, ...}
sensors @121501 {...ESH275 = false, ...}
```

A command is given at 120.95 seconds to open relay EY275. The associated relay position sensor does not indicate open as of the next sensor data update 51 milliseconds later. This is nominal behavior for the system. A DA that does not account for this delay will indicate a false positive in this case.

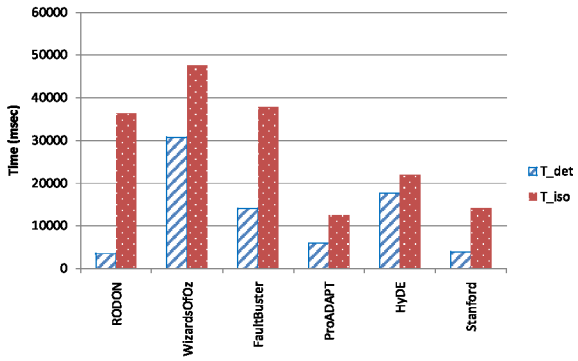

Figure 15: ADAPT detection and isolation times by DA.

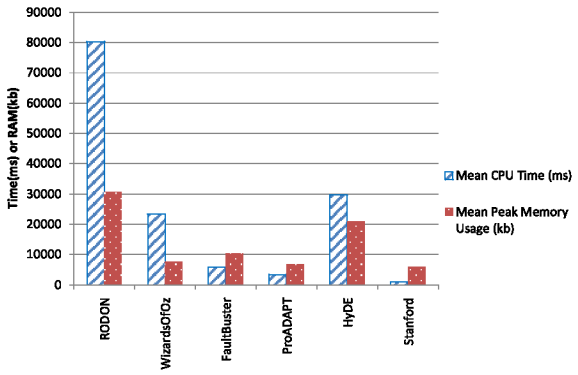

Figure 16: ADAPT CPU time and peak memory usage by DA.

The number of classification errors for each DA is shown in Figure 14. There were significantly more errors when compared to ADAPT-Lite. One reason is simply because there were more fault scenarios overall, resulting in more than four times as many faults injected in ADAPT compared to ADAPT-Lite. Another reason is because of the presence of multiple faults. Figure 17 shows the breakdown of classification errors by the number of faults in the scenario. The errors in the no-fault scenarios were obviously due to false positives. The double and triple faults are lumped into one multiple fault category. Note that there were an equal number of single fault and multiple fault scenarios, 40. Counting the number of faults injected in the scenarios, there were 40 total faults in the single fault scenarios and 90 total faults in the multiple fault scenarios. A DA that did not return any diagnoses at all would have 40 errors for single fault case and 90 errors for multiple faults. This is essentially what happened with FaultBuster because the diagnoses returned were not consistent with the fault catalog and they were treated as empty candidates (the same was true for FaultBuster

ADAPT-Lite). Figure 18 shows the errors by fault type.

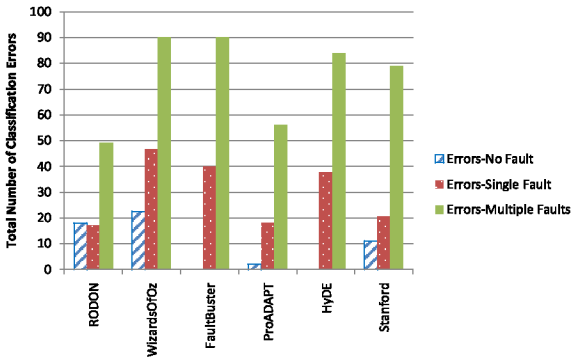

Figure 17: ADAPT classification errors per number of faults by DA.

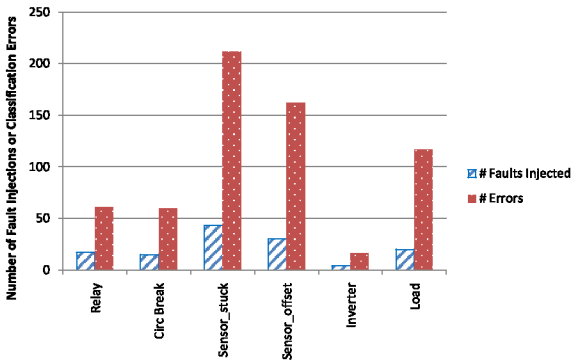

Figure 18: ADAPT classification errors by fault type.

The errors in the multiple fault scenarios were evenly divided among the faults; for example, if there were four classification errors in a scenario where two faults were injected, each fault was assigned two errors. We also did a more thorough assessment in which each diagnosis candidate was examined and classification errors were assigned to fault categories based on an understanding of which sensors are affected by the faults. The results are similar to evenly dividing the errors among the faults and are not shown here. The category 'Loads' includes faults in the fan, lights, pumps, and resistors. The category 'Sensor' includes faults in current, voltage, light, temperature, phase angle, and switch position sensors. Once again, there were many errors due to not distinguishing between offset and stuck faults and not detecting stuck faults that were within the nominal range of the sensor.

Figure 15 shows the detection and isolation times. The times are within the same order of magnitude for the different DAs. Some DAs have isolation times that are similar to its detection times while others show isolation times that are much greater than the detection times. This could reflect differences in reasoning strategies or differences in policies for when to declare an isolation based on accumulated evidence.

The CPU and memory usage are shown in Figure 16. The same comment for RODON mentioned previously in regards to memory usage applies here. The convex optimization approach applied in the Stanford DA(Gorinevsky and Poll, 2009) and the compiled arithmetic circuit in the ProADAPT DA (Ricks and Mengshoel, 2009) lead to very low CPU usages.

6 LIMITATIONS AND FUTURE WORK

We would like to extend the benchmarking efforts in several directions. Some obvious candidates are adding new systems, new DAs, and new metrics to the scope. The primary goal of the work described in this paper was to benchmark the available set of DAs on the ADAPT system. As a result we made several simplifying assumptions. We also ran into several issues during the course of this implementation that could not be addressed. In this section, we try to present those assumptions and issues, which we hope can be addressed in future work.

6.1 ADAPT System Scenarios

We intentionally limited the scope of the scenarios used in the benchmarking in order to focus on the performance of the DAs. Although we would like to include variable loads in future experiments, there were none used in this benchmarking activity. We limited ourselves to abrupt parametric and discrete faults. Faults were inserted assuming uniform probabilities and included component and sensor faults only. We plan to introduce other fault types (incipient, intermittent, and noise) in the future. We also understand that a true test would simulate operating conditions of the real system, i.e. the system operates nominally for long periods of time and failures occur periodically following the prior probability of failure distribution. It was also assumed that all sensor data was available to the DAs at all time steps. In the future we would like to relax this assumption and provide only a subset of the sensor data, possibly at differing sampling rates.

6.2 Metrics

Selecting the set of metrics to be used for benchmarking was a challenging job. We based our decision on the system and kinds of faults we were dealing with. In reality we also need to design metrics more closely associated with the context of use. One common metric is to minimize total cost of repair where cost includes down time to the customer, diagnostician's time, parts, etc. In addition, since we were dealing with abrupt, persistent, and discrete faults, metrics associated with incipient, intermittent, and/or continuous faults were not considered. The metrics listed in this paper do not capture the amount of effort necessary to build models of sufficient fidelity for the diagnosis task at hand. Furthermore, we did not attempt to investigate the ease or

Table 4: ADAPT-Lite DA Results

	RODON	Wizards Of Oz	Fault Buster	Pro-ADAPT	HyDE-A	HyDE-S	Rules-Rule	FACT	RacerX
FP Rate	0.0645	0.0000	0.1333	0.0333	0.0000	0.2000	0.8246	0.2813	0.0645
FN Rate	0.0968	0.5000	0.3438	0.0313	0.4688	0.0741	0.0000	0.0667	0.1613
Det Acc	0.9194	0.7419	0.7581	0.9677	0.7581	0.8548	0.2419	0.8226	0.8871
Class Errors	10.000	24.000	32.000	2.000	26.649	26.000	76.000	25.000	32.000
T_{det}(ms)	218	11530	1893	1392	13223	130	1000	373	126
T_{iso}(ms)	7205	11626	9259	4084	13840	653	282	9796	999999
CPU(ms)	11766	1039	2039	1601	24795	513	117	1767	139
Mem(kb)	26679	1781	2539	1680	5447	5795	3788	4340	3572

Table 5: ADAPT DA Results

	RODON	Wizards Of Oz	Fault Buster	Pro-ADAPT	HyDE	Stanford
FP Rate	0.5417	0.5106	0.8143	0.0732	0.0000	0.3256
FN Rate	0.0972	0.0959	0.2400	0.1392	0.3000	0.0519
Det Acc	0.7250	0.7417	0.4250	0.8833	0.8000	0.8500
Class Errors	84.067	159.248	130.000	76.000	121.569	110.547
T_{det}(ms)	3490	30742	14099	5981	17610	3946
T_{iso}(ms)	36331	47625	37808	12486	21982	14103
CPU(ms)	80261	23387	5798	3416	29612	963
Mem(kb)	29878	7498	10261	6539	20515	5912

difficulty of updating models with new or changed system information. The art of building models is an important practical consideration which is not addressed in the current work.

In future work, we would like to determine a set of application-specific use cases (maintenance, autonomous operation, abort decision etc.) that the DA is supporting and select metrics that would be relevant to that use case. For example, in an abort decision-support use case the detection time would be of utmost importance, however in an autonomous operation use case isolation accuracy would be more important.

6.3 Runtime Architecture

Some practical issues arose in the execution of experiments. Much effort was put into ensuring stable, uniform conditions on the host machines; however, due to time constraints and the unpredictable element introduced by running DAs developed externally, it was necessary to take measures that may have caused slight variability.

7 CONCLUSION

We presented the results from our effort to benchmark a set of diagnostic algorithms on the ADAPT electrical power system testbed. We learned some valuable lessons in trying to complete this effort. One major takeaway is that there is still a lot of work and discussion needed to determine a common comparison and evaluation framework for the diagnosis community. The other key observation is that no DA was able to be best in a majority of the metrics. This clearly indicates that the selection of DAs would necessarily involve a trade-off analysis between various preformance metrics.

As outlined in the previous section we have identified some of the limitations of our work. We are already in the process of planning continued experiments on the ADAPT system. We are also trying to identify other real systems that might be used for benchmarking since we believe that the performance of the DAs is also tied the system being diagnosed. Our long-term goal is to create a database of benchmarking results which presents the performance of a growing set of DAs on a growing set of application domains. We hope that such a database would help mission managers and operations personnel perform trade-off studies to determine which DAs they could consider for deployment on their systems.

ACKNOWLEDGMENTS

We extend our gratitude to Johan de Kleer (PARC), Alexander Feldman (Delft University of Technology), Lukas Kuhn (PARC), Serdar Uckun (PARC), Peter Struss (Technical University Munich), Gautam Biswas (Vanderbilt University), Ole Mengshoel (Carnegie Mellon University), Kai Goebel (University Space Research Association), Gregory Provan (University College Cork), and many others for valuable discussions in establishing the benchmarking framework. In ad-

dition, we extend our gratitude to David Nishikawa (NASA), David Jensen (Oregon State University), Brian Ricks (University of Texas at Dallas), Adam Sweet (NASA), David Hall (Stinger Ghaffarian Technologies), and many others for supporting the benchmarking activity reported here.

REFERENCES

(Bartys *et al.*, 2006) M. Bartys, R. Patton, M. Syfert, S. de las Heras, and J. Quevedo. Introduction to the DAMADICS actuator FDI benchmark study. 2006.

(Basseville and Nikiforov, 1993) M. Basseville and I. Nikiforov. *Detection of Abrupt Changes*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1993.

(de Freitas, 2001) N. de Freitas. Rao-blackwllised particle filtering for fault diagnosis. In *Proceedings of IEEE Aerospace Conference (AEROCONF'01)*, 2001.

(Gertler and Inc., 1998) Janos J. Gertler and NetLibrary Inc. *Fault detection and diagnosis in engineering systems[electronic resource]*. New York: Marcel Dekker, 1998.

(Gorinevsky and Poll, 2009) D. Gorinevsky and S. Poll. Estimation of faults in dc electrical power system. In *Proceedings of American Control Conference*, 2009.

(Hamscher *et al.*, 1992) W. Hamscher, L. Console, and J. de Kleer. *Readings in Model-Based Diagnosis*. Morgan Kaufmann, San Mateo, Ca, 1992.

(Iverson, 2004) D. Iverson. Inductive system health monitoring. In *Proceedings of The 2004 International Conference on Artificial Intelligence (ICAI)*, 2004.

(Karin *et al.*, 2006) L. Karin, R. Lunde, and B. Münker. Model-based failure analysis with RODON. In *Proceedings 17th European Conference on Artificial Intelligence (ECAI'06)*, 2006.

(Kavcic and Juricic, 1997) M. Kavcic and D. Juricic. A prototyping tool for fault tree based process diagnosis. In *Proceedings of 8th International Workshop on Principles of Diagnosis (DX'1997)*, 1997.

(Kostelezky *et al.*, 1990) W. Kostelezky, W. Krautter, R. Skuppin, M. Steinert, and R. Weber. The rule-based expert system Promotex I. Technical Report 2, ESPRIT-Project #1106, Stuttgart, 1990.

(Kurtoglu *et al.*, 2008) T. Kurtoglu, O. J. Mengshoel, and S. Poll. A framework for systematic benchmarking of monitoring and diagnostic systems. In *Annual Conference of the Prognostics and Health Management Society (PHM'08)*, 2008.

(Kurtoglu *et al.*, 2009a) T. Kurtoglu, S. Narasimhan, S. Poll, D. Garcia, L. Kuhn, J. de Kleer, A. van Gemund, and A. Feldman. A framework for systematic benchmarking of monitoring and diagnostic systems. In *Proceedings of 20th International Workshop on Principles of Diagnosis (DX'09)*, pages 383–396, 2009.

(Kurtoglu *et al.*, 2009b) T. Kurtoglu, S. Narasimhan, S. Poll, D. Garcia, L. Kuhn, J. de Kleer, A. van Gemund, and A. Feldman. Towards a framework for evaluating and comparing diagnosis algorithms. In *Proceedings of 20th International Workshop on Principles of Diagnosis (DX-09)*, pages 373–382, 2009.

(Lerner *et al.*, 2000) U. Lerner, R. Parr, D. Koleer, and G. Biswas. Bayesian fault detection and diagnosis in dynamic systems. In *Proceedings of The Seventeenth National Conference on Artificial Intelligence (AAAI'00)*, pages 531–537, 2000.

(Mengshoel, 2007) O. J. Mengshoel. Designing resource-bounded reasoners using bayesian networks: System health monitoring and diagnosis. In *Proceedings of 18th International Workshop on Principles of Diagnosis (DX'07)*, pages 330–337, 2007.

(Narasimhan and Brownston, 2007) S. Narasimhan and L. Brownston. HyDE - a general framework for stochastic and hybrid modelbased diagnosis. In *Proceedings of 18th International Workshop on Principles of Diagnosis (DX'07)*, pages 162–169, 2007.

(Orsagh *et al.*, 2002) R. Orsagh, M. Roemer, C. Savage, and M. Lebold. Development of preformance and effectiveness metrics for gas turbine diagnostic techniques. In *Proceedings of IEEE Aerospace Conference (AEROCONF'02)*, pages 2825–2834, 2002.

(Poll *et al.*, 2007) 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 *Proceedings of 18th International Workshop on Principles of Diagnosis (DX'07)*, 2007.

(Ricks and Mengshoel, 2009) B. Ricks and O. Mengshoel. The diagnostic challenge competition: Probabilistic techniques for fault diagnosis in electrical power systems. In *Proceedings of 20th International Workshop on Principles of Diagnosis (DX'09)*, pages 415–422, 2009.

(Roemer *et al.*, 2005) M. Roemer, J. Dzakowic, R. Orsagh, C. Byington, and G. Vachtsevanos. Validation and verification of prognostic health management technologies. In *Proceedings of IEEE Aerospace Conference (AEROCONF'05)*, 2005.

(Roychoudhury *et al.*, 2009) I. Roychoudhury, G. Biswas, and X. Koutsoukos. Designing

distributed diagnosers for complex continuous systems. 2009.

(Simon *et al.*, 2008) L. Simon, J. Bird, C. Davison, A. Volponi, and R. E. Iverson. Benchmarking gas path diagnostic methods: A public approach. In *Proceedings of the ASME Turbo Expo 2008: Power for Land, Sea and Air, GT*, 2008.

(Society of Automotive Engineers, 2007) E-32 Society of Automotive Engineers, 2007. Health and Usage Monitoring Metrics, Monitoring the Monitor, February 14, 2007, SAE ARP 5783-DRAFT.

(Sorsa and Koivo, 1998) T. Sorsa and H. Koivo. Application of artificial neural networks in process fault diagnosis. 29(4):843–849, 1998.

(Zymnis *et al.*, 2009) A. Zymnis, S. Boyd, and D. Gorinevsky. Relaxed maximum a posteriori fault identification. 2009.

Tolga Kurtoglu is a Research Scientist with Mission Critical Technologies at the Intelligent Systems Division of the NASA Ames Research Center working for the Systems Health Management group. His research focuses on the development of prognostic and health management systems, model-based diagnosis, design automation and optimization, and risk and reliability based design. He received his Ph.D. in Mechanical Engineering from the University of Texas at Austin in 2007 and has an M.S. degree in the same field from Carnegie Mellon University. Dr. Kurtoglu has published over 40 articles and papers in various journals and conferences and is an active member of ASME, ASEE, AIAA, and AAAI. Prior to his work with NASA, he worked as a professional design engineer at Dell Corporation in Austin, Texas.

Sriram Narasimhan is a Computer Scientist with University of California, Santa Cruz working as a contractor at NASA Ames Research Center in the Discovery and Systems Health area. His research interests are in model-based diagnosis with a focus on hybrid and stochastic systems. He is the technical lead for the Hybrid Diagnosis Engine (HyDE) project. He received his M.S and Ph.D. in Electrical Engineering and Computer Science from Vanderbilt University. He also has a M.S in Economics from Birla Institute of Technology and Science.

Scott Poll is a Research Engineer with the National Aeronautics and Space Administration (NASA) Ames Research Center, Moffett Field, CA, where he is the deputy lead for the Diagnostics and Prognostics Group in the Intelligent Systems Division. He is co-leading the evolution of a laboratory designed to enable the development, maturation, and benchmarking of diagnostic, prognostic, and decision technologies for system health management applications. He was previously the Associate Principal Investigator for Prognostics in the Integrated Vehicle Health Management Project in NASA's Aviation Safety Program. He received the BSE degree in Aerospace Engineering from the University of Michigan in 1994, and the MS degree in Aeronautical Engineering from the California Institute of Technology in 1995.

David Garcia is a Computer Programmer with Stinger Ghaffarian Technologies, working at NASA Ames Research Center in the Diagnostics and Prognostics Group (Intelligent Systems Division). He received his BS in Mathematics from Santa Clara University in 2006. He is the software lead for the ADAPT project, and designed and implemented the DXC Framework.

Stephanie Wright obtained her BS degree in Physics from Seattle University in 2008 and is currently a graduate student at Vanderbilt University in the EECS department. At Vanderbilt University she is a research assistant in the MACS (Modeling and Analysis of Complex Systems) group and her research interests are in diagnostics and health management of complex systems.