First International Diagnosis Competition – DXC'09

Tolga Kurtoglu*, Sriram Narasimhan**, Scott Poll***, David Garcia****, Lukas Kuhn[†], Johan de Kleer[†], Arjan van Gemund[‡], Alexander Feldman[†,‡]

* Mission Critical Technologies @ NASA Ames Research Center
** University of California, Santa Cruz @ NASA Ames Research Center
*** NASA Ames Research Center
**** Stinger Ghaffarian Technologies @ NASA Ames Research Center
[†] Palo Alto Research Center
[‡] Delft University of Technology

Abstract: A framework to compare and evaluate diagnosis algorithms (DAs) has been created jointly by NASA Ames Research Center and PARC. In this paper, we present the first concrete implementation of this framework as a competition called DXC'09. The goal of this competition was to evaluate and compare DAs in a common platform and to determine a winner based on diagnosis results. 12 DAs (model-based and otherwise) competed in this first year of the competition in 3 tracks that included industrial and synthetic systems. Specifically, the participants provided algorithms that communicated with the run-time architecture to receive scenario data and return diagnostic results. These algorithms were run on extended scenario data sets (different from sample set) to compute a set of pre-defined metrics. A ranking scheme based on weighted metrics was used to declare winners. This paper presents the systems used in DXC'09, description of faults and data sets, a listing of participating DAs, the metrics and results computed from running the DAs, and a superficial analysis of the results.

1. INTRODUCTION

The DX community meets every year at the International Workshop on the Principles of Diagnosis[1] to discuss the latest developments in the field of model-based diagnosis. Various diagnostic modeling approaches, associated reasoning algorithms, and applications to real and toy systems are presented. However, efforts to compare and evaluate diagnosis algorithms (DAs) on a common platform have been far and few in between. Other diagnosis communities have also not been actively involved in creating such platforms.

There have been attempts to benchmark DAs by computing performance metrics/indices (Orsagh et al., 2002, Bartys et al., 2006, Simon, et al., 2008). However these approaches were focused on specific domains and lack a general-purpose representation. In an effort to bridge this gap a framework called the DXC framework (Kurtoglu et al., 2009) was created to provide a level playing field to evaluate and compare DAs. This framework tried to establish a general purpose representation for system description, scenario data format, and diagnostic result format. A run-time architecture was created to execute DAs under similar conditions and compute performance metrics based on diagnostic output and ground-truth data.

In this paper, we present the first concrete implementation of the DXC framework called the DX Competition[2] (DXC'09). 12 DAs (model-based and otherwise) competed in 3 tracks that included industrial and synthetic systems. Initially the participants were provided with system descriptions and a sample data set that included nominal and fault scenarios. The participants had to provide algorithms that communicated with the run-time architecture to receive scenario data and return diagnostic results. For the competition we ran all the algorithms on extended scenario data sets (different from sample set) to compute a set of pre-defined metrics. The metrics (with associated weighting) were used to rank the DAs.

The rest of this paper details the constituent pieces of this competition. Section 2 describes the tracks and systems used in the competition. Section 3 lists the classes of faults that were injected, how they were injected, and the sensor data sets that were generated as a result. Section 4 defines all the metrics used in the competition. Section 5 describes the conditions of the competition including information on how the algorithms were started and executed. Section 6 lists and briefly describes the participating DAs. Section 7 provides results of the competition and an analysis of the performance of the DAs. Section 8 introduces the assumptions that were made in implementing this competition, issues that were identified, and scope for possible extensions. Section 9 presents the conclusion and looks forward to the continuation of this competition in future years.

2. TRACKS & SYSTEMS

One of the primary goals of DXC'09 is to facilitate the development of domain independent diagnostic software. Furthermore, diagnostic software should be stress tested with difficult cases to determine its strengths and weaknesses and to pose a challenge. A diagnostic problem may be interesting due to its practical importance or it may be challenging due to

[1] http://www.isy.liu.se/dx09/
[2] http://dx-competition.org/

its size and complexity. To facilitate all this we included multiple DXC tracks, and (optionally) multiple tiers in each track. The DXC'09 tracks and tiers are summarized in Table 1.

Table 1. Tracks, systems, and tiers in DXC'09

Track	Tier	Systems	Description
Industrial	1	ADAPT-Lite	Basic faults injected into a simplified EPS (Electrical Power System) testbed
	2	ADAPT	More complex faults injected into the full EPS distribution system
Synthetic	1	ISCAS85	Multiple faults injected into the circuits from the ISCAS85 benchmarks

2.1 Industrial Track

The hardware system for the DXC'09 Industrial Track 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.

The scope of the ADAPT EPS testbed used for DXC Industrial Track is shown in Fig. 1. Tier 1 has the reduced scope as indicated. The nomenclature in the figure is consistent with the system description provided to all participants, which provides component, connection, and mode information. The characteristics of Tier 1 and Tier 2 are summarized in Table 2. The greatest simplification of Tier 1 relative to Tier 2 is not the reduced size of the domain but the elimination of nominal mode transitions. The starting configuration for Tier 1 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 Tier 2 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.

Table 2. Industrial track tier characteristics

Aspect	Tier 1	Tier 2
#Comps/Modes	37 / 93	173 / 430
Initial State	Relays closed; circuit breakers closed	Relays open; circuit breakers closed
Nominal mode changes?	No	Yes

2.2 Synthetic Track

For the synthetic track, we have used the well-known benchmark models of ISCAS85 circuits (Brglez and Fujiwara, 1985). These circuits are purely combinational, i.e., they contain no flip-flops or other memory elements. Note that the high-level structure of the ISCAS85 circuits, which can be beneficial to Model-Based Diagnosis (MBD) analysis, has been flattened out. A reverse engineering effort had resulted in high-level Verilog models (Hansen et al., 1999). Table 3 summarizes the circuits used in the synthetic DXC'09 track. Note that for many tasks of MBD (e.g., computing MFMC (Max-Fault Min-Cardinality) observations (Feldman et al., 2008)), the number of components in the ISCAS85 circuits can be reduced by performing cone identification (Siddiqi and Huang, 2007, de Kleer, 2008). The number of components in the reduced circuits is shown in the rightmost column of Table 3. We have left the decision if to identify cones to the competitors, i.e., we distribute the non-reduced circuits. In this first year of the competition we injected the complete fault at one instant. For example, if 3 components are faulted, the first observation provided to the DA is the result of all three faults injected simultaneously.

Table 3. ISCAS85 models (V and C denote the total number of variables and clauses, respectively)

	original					reduced
sys	|IN|	|OUT|	|COMPS|	V	C	|COMPS|
74182	9	5	19	47	75	6
74L85	11	3	33	77	118	15
74283	9	5	36	81	122	14
74181	14	8	65	144	228	15
c432	36	7	160	356	1028	59
c499	41	32	202	445	1428	58
c880	60	26	383	826	2224	77
c1355	41	32	546	1133	3220	58
c1908	33	25	880	1793	4756	160
c2670	233	140	1193	2695	6538	167
c3540	50	22	1669	3388	9216	353
c5315	178	123	2307	4792	13386	385
c2688	32	32	2416	4684	14432	1456
c7552	207	108	3512	7232	19312	545

3. FAULT INJECTION AND SCENARIOS

3.1 Industrial Track

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

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 the first DX competition.

In addition, the software architecture described in (Kurtoglu et al., 2009) allows the injection of multiple faults into the system. Distinct faults types that are injected into the testbed for the DX Competition are shown Table 4 and summarized in Table 5.

Table 4. Fault types used for the industrial tracks of DXC'09

Component	Fault Description
Battery	Degraded
Boolean Sensor	Stuck at Value
Circuit Breaker	Tripped
	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

As shown in Table 5, nominal scenarios comprise roughly half of the Tier 1 and one-third of the Tier 2 competition scenarios. The Tier 1 fault scenarios are limited to single faults. Half of the Tier 2 faults scenarios are single faults; the others are double or triple faults. For both tiers once faults are injected they persist until the end of the scenario. In the case of multiple faults, they may be injected simultaneously or sequentially. In the first year of the competition the fault types are limited to additive parametric (abrupt changes in parameter values) and discrete (unexpected changes in system state).

Table 5. Number of sample and competition scenarios for industrial track

	Sample		Competition	
#Scenarios	Tier 1	Tier 2	Tier 1	Tier 2
Nominal	32	39	30	40
Single-fault	27	54	32	40
Double-fault	0	19	0	30
Triple-fault	0	1	0	10

3.2 Synthetic Track

To present the scenario generation algorithm with the appropriate level of formality we need a number of definitions.

Definition 1. (Diagnostic System). A diagnostic system DS is defined as the triple DS = SD, COMPS, OBS , where SD is a propositional theory over a set of variables V , COMPS ⊂ V, OBS ⊂ V, COMPS is the set of assumables, and OBS is the set of observables.

We partition the set of observable variables OBS into inputs IN and outputs OUT such that OBS = IN ∪ OUT and IN ∩ OUT = ∅.

Definition 2. (Diagnosis). Given a diagnostic system DS = SD, COMPS, OBS , an observation over some variables in OBS, and a health assignment , is a diagnosis iff SD ∧ ∧ is consistent.

Definition 3. (Minimal Diagnosis). A diagnosis is minimal if no diagnosis ' exists such that NL(') ⊂ NL( ), where NL( ) is the set of negative literals in .

Definition 4. (Cardinality of a Diagnosis). The cardinality of a diagnosis, denoted as ^ ,^is defined as the number of negative literals in .

A minimal cardinality diagnosis is a minimal diagnosis, but the opposite does not hold. There are minimal diagnoses which are not minimal cardinality diagnoses.

The purpose of Alg. 1 is to generate observations leading to diagnoses of increasing minimal cardinality.

Algorithm 1: A greedy stochastic scenario generation algorithm

```
function MAKEALPHAS(DS, N ) returns set of terms
  inputs:
      DS = <SD, COMPS, OBS>, diag. system
      OBS = IN ∪ OUT, IN ∩ OUT = ∅
      N, integer, observations per cardinality
  local variables:
      , , n, fault, terms
      i, c, integers,
      R, set of terms, result, initially ∅
1:    for i = 1 ... N do
2:        ← RANDOMINPUTS(IN)
3:        ← COMPUTENOMINALOUTPUTS(DS, )
4:        c ← 0
```

```
5:          forall v ∈ OUT do
6:              n ←  ^ FLIP ( , v)
7:              fault = MCFAULT( n)
8:              if |fault| > c then
9:                  c ← |fault|
10:                 R ← R ∪ <fault, n>
11:             end if
12:         end for
13:     end for
14:     return R
    end function
```

Algorithm 1 uses a number of auxiliary functions. RANDOMINPUTS in line 2 assigns uniformly distributed random values to each input. Given the "all healthy" assignment, and the diagnostic system, COMPUTENOMINALOUTPUTS (line 3) propagates the inputs and computes values for each output variable in OUT. The loop in lines 5 – 12 increases the cardinality by greedily flipping the values of the output variables. For each new candidate observation $_n$, Alg. 1 uses the diagnostic oracle MCFAULT in line 7 to compute the minimal cardinality of the diagnosis resulting from $_n$. If the cardinality of the diagnosis increases, the observation and the diagnoses are added to the result set (line 10).

By running Alg. 1 we get up to N observations leading to faults of cardinality 1, 2, ..., n, where n is the cardinality of the MFMC diagnosis for the respective circuit. Alg. 1 clearly shows a bootstrapping problem. In order to create "difficult" scenarios for a DA we need the DA (in line 7) to be able to solve those "difficult" scenarios. To overcome this problem we have used subset-minimal diagnoses instead of MC diagnoses. Our approach is similar to (Feldman et al., 2008).

4. EVALUATION METRICS

A set of 9 metrics has been defined for assessing the performance of the diagnostic algorithms. For DXC we make a distinction between temporal, technical, and 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 including accuracy and diagnostic cost/utility. 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 9 metrics are listed in Table 6. The notation used for the definition of the metrics is as follows:

Table 6. Metrics summary

Symbol	Name	Description	Class/Category/ Tracks Used
"Per System Description" Metrics			
M_{FPR}	False Positives Rate	Spurious faults rate	Technical / Detection/I
M_{FNR}	False Negatives Rate	Missed faults rate	Technical / Detection/I
M_{FDA}	Detection Accuracy	Correctness of the detection	Technical / Detection/I
"Per Scenario" Metrics			
M_{fd}	Fault Detection Time	Time for detecting a fault	Temporal / Detection/I,S
M_{fi}	Fault Isolation Time	Time for last persistent diagnosis	Temporal / Isolation/I,S
M_{ia}	Classification Errors	Number of mode classification errors	Technical / Isolation/I
M_{utl}	Diagnostic Utility	Cost related to component replacements due to incorrect diagnosis	Technical / Isolation/S
M_{cpu}	CPU Load	CPU time spent	Computational / Detection & Isolation/I,S
M_{mem}	Memory Load	Memory allocated	Computational / Detection & Isolation/I,S

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

$_{act}$ – The true component mode vector (ground truth)

$_{pre}$ – 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

C – All possibly faulted components

D – Faulted components in $_{pre.}$

I – Faulted components in $_{act.}$

Finally, using the aforementioned notation, the 9 metrics are defined as:

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

$$M_{fd} = t_{fd} \quad (1)$$

M_{fi} – Fault Isolation Time: 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.

$$M_{fi} = t_{fi} \quad (2)$$

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

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

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

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

where $t_{inj} = \infty$ for a nominal scenario

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

$$M_{FNR} = \frac{\sum_{s \in S_f} m_{fn}(s)}{|S_f|} \quad (5)$$

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

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

M_{FDA} – Detection Accuracy: 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:

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

M_{ia} – Classification Errors: Isolation classification error metric measures the accuracy of the fault isolation by a diagnostic algorithm and is defined as the Hamming distance between the true component mode vector $_{act}$ and the predicted component mode vector $_{pre.}$[3]

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 candidate probabilities reported by the DA.

M_{utl} – Diagnostic Utility: The intuition behind the metric is to charge a DA for every incorrect component replacement it required to restore the circuit to functioning. For example, the correct diagnosis should always receive a perfect score. The diagnosis all components bad has a cost of the number of components. Consider a single fault and the DA reports all components good. Finding the faulty component would require on average replacing component by component until the system was functioning correctly (on average half the components). More generally:

(8)

Where c(n,m) is defined as the expected number of trials needed to isolate n out of m. If n is much smaller than m, then it is approximately:

(9)

For example, to find 1 fault in m has cost m/2. To find 2 faults in m is 2m/3. Similarly to the classification metric, if more than one predicted mode vector is reported by a DA, then error is calculated for each predicted component mode vector and weighted by candidate probabilities reported by the DA.

M_{cpu} – CPU Load: This is the average CPU load during the experiment

$$M_{CPU} = t_s + \sum_{q \in T_d} q \quad (10)$$

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

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

$$M_{mem} = \max_{m \in M_d} m \quad (11)$$

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

[3] The Hamming distance between two strings of data values (of equal length) is the number of positions for which the corresponding data values are different.

5. COMPETITION SETUP AND SCORING

Version 1.1 of the DXC Framework, implemented as specified in (Kurtoglu et al., 2009), was used to run the competition. Two computers with identical hardware[4] were set up, one running Windows™ and the other Linux. The choice of target operating system was left to DA developers.

System profiling was performed on the machines over a period of days to ensure stable experiment conditions.

DAs were run on competition datasets over a period of two weeks. The Evaluator was then run on the full results set, assigning relative rankings for each metric. Since there were multiple systems in the Synthetic Track, the metrics computed for each system were aggregated before assigning relative rankings. The per scenario metrics were averaged over all scenarios and aggregated over all systems. For each of the Industrial track tiers there was only one system, so no aggregation was necessary.

A DA that ranked first place in a given metric was awarded 10 points, second place was awarded 8, third 7, etc. This score was then multiplied by a metric weight, shown in Tables 8, 9, and 10, and added to the DA's total.

Metric weights for the Industrial Track were determined by considering a number of use cases in which the importance of each metric was subjectively assessed. For example, in an abort use case high importance was given to the mean time to detect a fault whereas in a maintenance use case more weight was given to the ability to isolate a fault. Similar considerations were given to use cases such as real-time recovery and control, ground support operations, and resource limited applications. Since a use case was not specified as part of the competition scenarios, we simply averaged over all of the use cases to arrive at the final metric weights.

6. DIAGNOSTIC ALGORITHMS

The teams that participated in the First International Diagnosis Competition are listed in Table 7.

Table 7. DXC participating DAs

Team Name	Track(s)	Algorithm Type
FACT	I1	Model-based
Fault Buster	I1, I2	Statistical
HyDE-A	I1, I2	Model-based
HyDE-S	I1	Model-based
Lydia	S	Model-based
NGDE	S	Model-based
ProADAPT	I1, I2	Probabilistic
RacerX	I1	Change detection
RODON	I1, I2, S	Model-based
RulesRule	I1	Rule-based
StanfordDA	I2	Optimization
Wizards of Oz	I1, I2	Model-based

[4] Intel® XEON™ 2x2.20Ghz, 3.60 GB RAM

A total of twelve DAs participated, nine in Tier 1 of the Industrial Track, six in Tier 2, and three in the Synthetic Track. 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. Lydia – is a declarative modeling language specifically developed for Model-Based Diagnosis (MBD). The language core is propositional logic, enhanced with a number of syntactic extensions for ease of modeling. The accompanying toolset currently comprises a number of diagnostic engines and a simulator tool (Feldman et al., 2006).
6. NGDE – Allegro Common Lisp implementation of the classic GDE. Uses a minimum-cardinality candidate generator to construct diagnoses for the competition.
7. 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).
8. RacerX – is a detection-only algorithm which detects a percentage change in individual filtered sensor values to raise a fault detection flag.
9. 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).

10. 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.
11. 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 l_1 regularization (Zymnis et al., 2009).
12. 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.

7. RESULTS AND DISCUSSION

7.1 Industrial Track

The results for the Industrial Track are shown in Table 8 and Table 9 for Tier 1 and Tier 2, respectively. The overall winner for both tracks was ProADAPT. RODON placed second in Tier 1 and third in Tier 2. The StanfordDA, which did not participate in Tier 1, placed second in Tier 2. However, ProADAPT and StanfordDA benefitted from previous funded experience with ADAPT so RODON was the official winner of both tiers. The distribution of first or second ranks within each metric was spread out among the DAs, no DA ranked first or second for all of the metrics. Note that the final scores and ranks depend on the weights applied to each metric. Different weights, corresponding to different use cases, would affect the results. The sensitivity of the results to the metrics and weights is left for future study.

Figures 2-9 are graphical depictions of the data in Tables 8 and 9. A few observation follow. 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. For other DAs, false positives also resulted from nominal commanded mode changes in Tier 2 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 and examples were provided in the sample data. A DA that does not account for this delay will likely indicate a false positive in this case.

In several instances DAs reported diagnosis mode IDs which did not match the names specified in the system catalog. For these cases the diagnosis was treated as an empty candidate.

Table 8. Industrial track tier 1 results

	Weight	RODON	Wizards OfOz	Fault Buster	ProADAPT	HyDE-A	HyDE-S	RulesRule	FACT	RacerX
FP Rate	1.3	0.0645	0.0000	0.1333	0.0333	0.0000	0.2000	0.8246	0.2813	0.0645
Ranking		4	1	6	3	1	7	9	8	4
Points		6	9	4	7	9	3	1	2	6
FN Rate	1.3	0.0968	0.5000	0.3438	0.0313	0.4688	0.0741	0.0000	0.0667	0.1613
Ranking		5	9	7	2	8	4	1	3	6
Points		5	1	3	8	2	6	10	7	4
DetAcc	0.3	0.9194	0.7419	0.7581	0.9677	0.7581	0.8548	0.2419	0.8226	0.8871
Ranking		2	8	6	1	6	4	9	5	3
Points		8	2	3.5	10	3.5	6	1	5	7
Class Errors	2.2	10.000	24.000	32.000	2.000	26.649	26.000	76.000	25.000	32.000
Ranking		2	3	7	1	6	5	9	4	7
Points		8	7	2.5	10	4	5	1	6	2.5
T_det (ms)	2.2	218	11530	1893	1392	13223	130	1000	373	126
Ranking		3	8	7	6	9	2	5	4	1
Points		7	2	3	4	1	8	5	6	10
T_iso (ms)	1.5	7205	11626	9259	4084	13840	653	282	9796	999999
Ranking		4	7	5	3	8	2	1	6	9
Points		6	3	5	7	2	8	10	4	1
CPU (ms)	0.6	11766	1039	2039	1601	24795	513	117	1767	139
Ranking		8	4	7	5	9	3	1	6	2
Points		2	6	3	5	1	7	10	4	8
Mem (kb)	0.6	26679	1781	2539	1680	5447	5795	3788	4340	3572
Ranking		9	2	3	1	7	8	5	6	4
Points		1	8	7	10	3	2	5	4	6
FINAL SCORES:		59.850	46.300	35.750	72.800	31.750	59.500	51.800	50.400	51.850
FINAL RANK:		2	7	8	1	9	3	5	6	4

Table 9. Industrial track tier 2 results

	Weight	RODON	Wizards OfOz	Fault Buster	ProADAPT	HyDE	Stanford
FP Rate	1.3	0.5417	0.5106	0.8143	0.0732	0.0000	0.3256
Ranking		5	4	6	2	1	3
Points		5	6	4	8	10	7
FN Rate	1.3	0.0972	0.0959	0.2400	0.1392	0.3000	0.0519
Ranking		3	2	5	4	6	1
Points		7	8	5	6	4	10
DetAcc	0.3	0.7250	0.7417	0.4250	0.8833	0.8000	0.8500
Ranking		5	4	6	1	3	2
Points		5	6	4	10	7	8
Class Errors	2.2	84.067	159.248	130.000	76.000	121.569	110.547
Ranking		2	6	5	1	4	3
Points		8	4	5	10	6	7
T_det (ms)	2.2	3490	30742	14099	5981	17610	3946
Ranking		1	6	4	3	5	2
Points		10	4	6	7	5	8
T_iso (ms)	1.5	36331	47625	37808	12486	21982	14103
Ranking		4	6	5	1	3	2
Points		6	4	5	10	7	8
CPU (ms)	0.6	80261	23387	5798	3416	29612	963
Ranking		6	4	3	2	5	1
Points		4	6	7	8	5	10
Mem (kb)	0.6	29878	7498	10261	6539	20515	5912
Ranking		6	3	4	2	5	1
Points		4	7	6	8	5	10
FINAL SCORES:		70.500	51.400	52.400	83.200	61.000	81.500
FINAL RANK:		3	6	5	1	4	2

This could either negatively or positively impact the classification error metric depending on whether the DA had a correct or incorrect isolation. Participants were encouraged to run their DA output through the evaluator code that was distributed with the sample data sets to check for and correct these syntax errors.

There are a few remarks in regards to the timing metrics listed in Table 6 and shown graphically in Fig. 4. First, RacerX did not have an isolation time as it was a detection-only DA. Second, note the somewhat 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. 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 is 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.

Tier 1 had the interesting circumstance that the same DA was implemented by two different modelers. HyDE-A was modeled primarily with Tier 2 in mind and had a policy of waiting for transients to settle before requesting a diagnosis. The same policy was simply applied to Tier 1 as well, even though transients in Tier 1 corresponded strictly to fault events. On the other hand, HyDE-S was modeled only for Tier 1 and did not include a lengthy time-out period for transients to settle. HyDE-S had dramatically smaller mean detection and isolation times (see Fig. 4) with roughly the same number of classification errors (Fig. 3) as HyDE-A. This illustrates the kind of impact that modeling and implementation decisions have on DA performance.

7.2 Synthetic Track

As can be seen in Table 7 all synthetic track DAs are model-based. Lydia uses a stochastic approach to identify diagnoses while RODON and NGDE use the familiar GDE-like approaches. Their overall utility scores are not dramatically different.

The results for the Synthetic Track are presented in Table 10. Based on the overall metric NGDE was first, Lydia second, and RODON third. Lydia was used to generate the scenario sets and therefore is disqualified. Furthermore the designers of Lydia and NGDE both participated in the design of DXC, and are thus disqualified. So RODON is the official winner. RODON scored reasonably well on the smaller circuits but failed to return any diagnoses for the 4 larger circuits.

Use of computational resources varied dramatically over the systems. Lydia used an order of magnitude fewer resources than either RODON or NGDE and thus ranked first along the memory and CPU metrics. RODON and NGDE are very similar in resource usage, with RODON edging out NGDE.

Fig. 10 shows the DA utility for each of the circuits. Note that the utility score decreases significantly with circuit size. This decrease is not a result of poor performance or algorithm design. Rather, an oracle could not do much better as a large

number of faults can exhibit the same input-output behavior and no DA could isolate the injected fault out of the large ambiguity groups. The challenge presented by large ambiguity groups is discussed further in the following section and the NGDE and Lydia papers included in this collection.

8. ASSUMPTIONS, ISSUES, AND EXTENSIONS

The primary goal of this competition was to demonstrate an end-to-end implementation of the DXC framework and create a foundation for future DX competitions. 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 competitions.

Although the competition was a success, it only addresses a small set of the types of diagnostic tasks, which occur in practice. It would be unfortunate for the DX community to focus only on the tasks of this competition. Our goal is to continually expand the coverage of diagnostic challenges experienced in the field. Our hope is that every successive year will expand the set of tasks in the competition and in doing so produce an ever growing repository DX researchers have available to evaluate their own algorithms.

8.1 Competition Scope

In the first year of the diagnostic competition, the fault signatures were limited to abrupt parametric and discrete types. Faults were inserted assuming uniform probabilities and included component and sensor faults. In future years, we will provide the failure rates of components and use these to evaluate the precision of diagnoses. For the Industrial Track, other fault types are presently possible to inject in the testbed – including incipient, intermittent, and noise – and could be included in future work. Additional ideas for future research include giving DAs reduced sensor sets, introducing multi-rate sensor data, injecting transient faults, allowing for autonomous transitions, adding variable loads, and extending the scope and complexity of the physical system. For the synthetic track, all the systems were known a priori. This means researchers could optimize for these circuits. We don't believe this happened this year, but to avoid this in future years we will include entirely novel circuits along with the familiar ones. This year we sampled only one observation time. We will provide multiple observations. This will evaluate a DAs ability to merge information from multiple times. An important component of troubleshooting is introducing probe points. In future years, we can evaluate the number of probes needed to isolate the fault. This year the input vector was supplied. The diagnostician could construct the input vector, which was most informative. This year the Synthetic Track focused on combinatorial circuits. In subsequent years we hope to introduce troubleshooting of sequential circuits. Finally, digital circuits are convenient to model and conveniently illustrate many aspects of diagnostic algorithms. In future years, we will extend the types of systems to include. Two comparatively easy types of systems to add are reprographic engines as we have a tool available to

Table 10. Synthetic track results

		Lydia			NGDE			RODON		
circuit	#comp	cpu	mem	utl	cpu	mem	utl	cpu	mem	utl
74182	19	51	154	0.4137	6335	11540	0.4793	3043	19773	0.4448
74L85	33	68	223	0.2433	6365	11784	0.3098	3888	20979	0.1952
74283	36	60	229	0.1580	6385	12231	0.1553	5351	20637	0.1147
74181	65	64	401	0.1504	6619	14625	0.1931	12527	25432	0.1417
c432	160	115	878	0.0871	7520	17868	0.2096	22621	36811	0.0906
c499	202	130	1094	0.0622	20347	32649	0.0699	23504	39872	0.0089
c880	383	203	1945	0.0483	13718	28622	0.0401	20347	43687	0.0182
c1355	546	296	2759	0.0295	22550	37930	0.0246	23253	33530	0.0012
c1908	880	538	4134	0.0179	26171	39843	0.0150	27718	38557	0.0180
c2670	1193	937	5867	0.0647	20537	61722	0.1076	35680	43063	0.0442
c3540	1669	1674	7900	0.0319	27022	82045	0.0407	0	0	0.0000
c5315	2307	3091	11316	0.0165	30926	93116	0.0275	0	0	0.0000
c6288	2416	3530	12037	0.0008	17483	102420	0.0563	0	0	0.0000
c7552	3512	11817	16679	0.0317	37989	125910	0.0283	0	0	0.0000
	Averaged	1613	4687	0.0969	17855	48022	0.1255	12709	23024	0.0770
	Per Metric Rank	1	1	2	3	3	1	2	2	3
	Points	10	10	8	7	7	10	8	8	7
	Metric Weight	1.5	1.5	7	1.5	1.5	7	1.5	1.5	7
	Final Scores		86			91			73	
	Final Rank		2			1			3	

generate such models, and analog circuits.

8.2 Metrics

Selecting the set of metrics to be used for evaluation 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 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.

The isolation accuracy metric used for the industrial track was not suitable for the synthetic track. A DA which reported nothing wrong on every scenario would come close to winning the competition based on this metric. The main problem with this metric is that the number of faulty components is always small with respect to the size of the system. As a result we cannot differentiate adequately between a few faults and no faults.

Isolation classification error was also not suitable for the synthetic track. This metric still suffers from the problem that all good is scored too high: The Hamming distance between a single fault and every component good is very small!

Ideally we would like to use a SAT solver to evaluate the accuracy of a DA's diagnosis. However, we did not have time to implement it so as an alternate we selected utility as the isolation accuracy metric for the synthetic track. One of the major flaws of this metric is that average expected utility scores decrease with system size, thereby implicitly de-weighting diagnoses of larger circuits. We also considered and rejected a classification error metric which would assign a high score to any fault from an ambiguity group which we considered a bigger flaw than the flaw for the utility metric.

Finally, 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. 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.

8.3 Competition Setup

Some practical issues arose in the execution of competition 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 external DA submissions, it was necessary to take measures that may have caused slight variability. One example was the manual examination of ongoing experiment results for quality assurance. Future releases of the DXC Framework can address this by being more robust to unexpected DA behavior, and sending email notifications in the event of such.

Additionally, for Java DAs, 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.

The problem was bypassed by running all Java DAs on Linux. This worked for all save one, RODON. When it was determined that any change in RODON's peak memory usage score would not affect the final rankings in any way, the issue was waived.

9. CONCLUSIONS

We presented the successful implementation of the DXC framework called DXC'09. We learned some valuable lessons trying to run this competition. One major takeaway is that there is still a lot of work and discussion needed to determine common comparison and evaluation framework for the diagnosis community.

We hope to continue the work next year by running DXC'10. We have identified several ways to extend the systems used in the current competition some of which can be achieved in a year's time. We also hope to add other systems to the fold, which may pose different diagnostic challenges.

ACKNOWLEDGMENTS

We extend our gratitude to David Nishikawa (NASA), David Jensen (Oregon State University), Brian Ricks (University of Texas at Dallas), Ole Mengshoel (Carnegie Mellon University), Adam Sweet (NASA), David Hall (Stinger Ghaffarian Technologies), all DXC'09 competitors, the DX'09 organizers and many others for valuable discussions, criticism and help.

REFERENCES

Bartys, M., R. Patton, M. Syfert, S. de las Heras, and J. Quevedo (2006). Introduction to the DAMADICS Actuator FDI Benchmark Study, Control Engineering Practice, vol 14, pp. 577-596.

Brglez, F., and H. Fujiwara (1985). A neutral netlist of 10 combinational benchmark circuits and a target translator in Fortran. In: Proc. ISCAS'85, pages 695–698.

de Kleer, J (2008). An improved approach for generating Max-Fault Min-Cardinality diagnoses. In: Proc. DX'08, pp. 247–252.

Feldman A., J. Pietersma, and A. van Gemund (2006). All roads lead to fault diagnosis: Model-based reasoning with LYDIA. In: Proc. BNAIC'06.

Feldman, A., G. Provan, and A. van Gemund (2008). Computing observation vectors for Max-Fault Min-Cardinality diagnoses. In: Proc. AAAI'08, pp. 919–924.

Hansen, M., H. Yalcin, and J. Hayes (1999). Unveiling the ISCAS-85 benchmarks: A case study in reverse engineering. IEEE Design & Test, 16(3):72–80.

Karin L., R. Lunde, and B. Münker. (2006). Model-Based Failure Analysis with RODON, In: Proc. ECAI'06.

Kurtoglu, T., S. Narasimhan, S. Poll, D. Garcia, L. Kuhn, J. de Kleer, A. van Gemund, and A. Feldman (2009). Towards a Framework for Evaluating and Comparing Diagnosis Algorithms. In: Proc. DX'09.

Mengshoel O.J. (2007). Designing resource-bounded reasoners using Bayesian networks: System health monitoring and diagnosis, In: Proc. DX'07, pp. 330-337.

Narasimhan, S., and L. Brownston (2007). HyDE – A General Framework for Stochastic and Hybrid Model-based Diagnosis. In: Proc. DX'07, pp. 162-169.

Orsagh R.F., M.J. Roemer, C.J. Savage, and M. Lebold, (2002). Development of Performance and Effectiveness Metrics for Gas Turbine Diagnostic Techniques. Aerospace 2002 IEEE Conference Proceedings, Vol6, pp. 2825-2834.

Poll, S., 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 (2007). Advanced Diagnostics and Prognostics Testbed. In: Proc. DX'07.

Roychoudhury I., G. Biswas, and X. Koutsoukos (2009). Designing Distributed Diagnosers for Complex Continuous Systems, IEEE Transactions on Automation Science and Engineering, vol. 6, no. 2, pp. 277-290.

Siddiqi, S. and J. Huang (2007). Hierarchical diagnosis of multiple faults. In: Proc. IJCAI'07, pp. 581–586.

Simon L., J. Bird, C. Davison, A. Volponi, R. E. Iverson, (2008). Benchmarking Gas Path Diagnostic Methods: A Public Approach, Proceedings of the ASME Turbo Expo 2008: Power for Land, Sea and Air, GT2008.

Zymnis A., S. Boyd, and D. Gorinevsky (2009). Relaxed maximum a posteriori fault identification, Signal Processing, vol. 89, no. 6, 2009, pp. 989–999.

Appendix A. FIGURES.

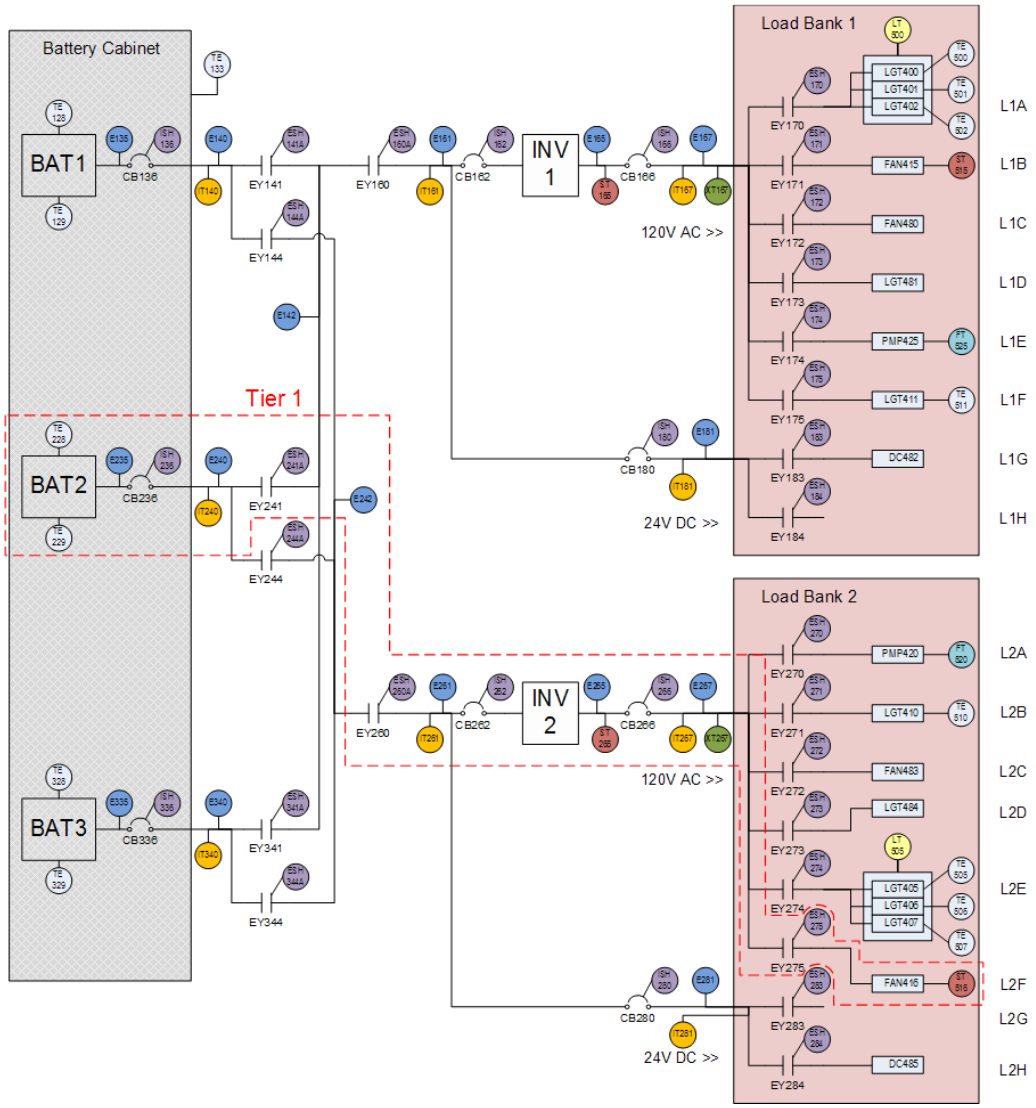

Fig. 1. The ADAPT EPS (Electrical Power System)

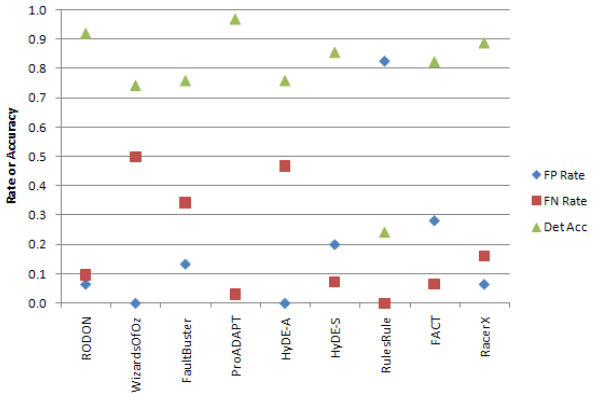

Fig. 2. Industrial track tier 1 false positive rate, false negative rate, and detection accuracy by DA

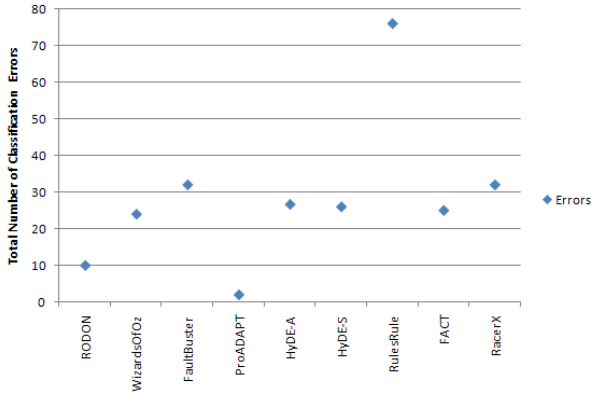

Fig. 3. Industrial track tier 1 classification errors by DA

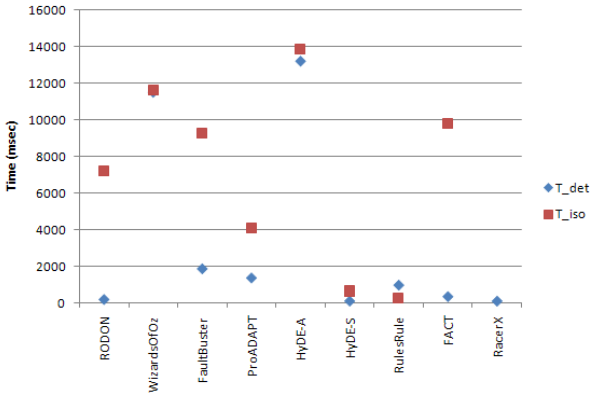

Fig. 4. Industrial track tier 1 detection and isolation times by DA

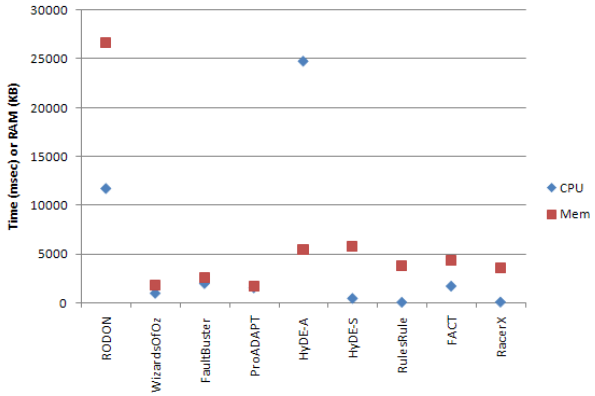

Fig. 5. Industrial track tier 1 CPU time and peak memory usage by DA

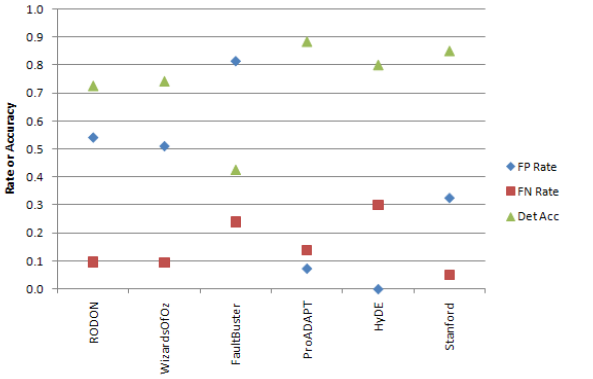

Fig. 6. Industrial track tier 2 false positive rate, false negative rate, and detection accuracy by DA

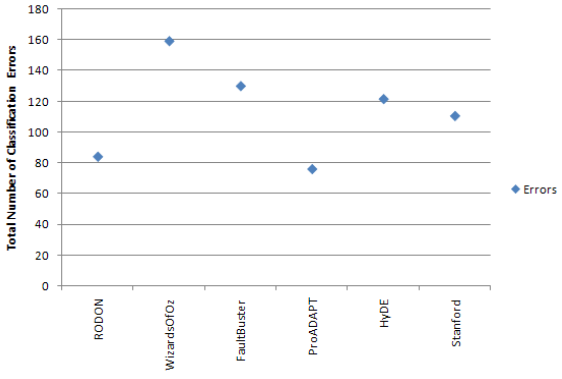

Fig. 7. Industrial track tier 2 classification errors by DA

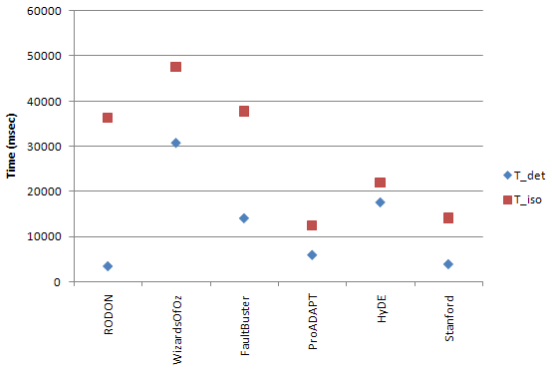

Fig. 8. Industrial track tier 2 detection and isolation times by DA

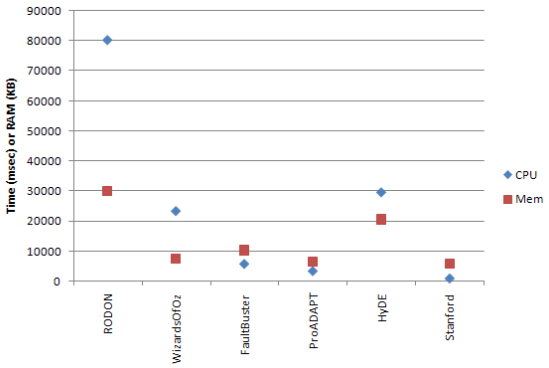

Fig. 9. Industrial track tier 2 CPU time and peak memory usage by DA

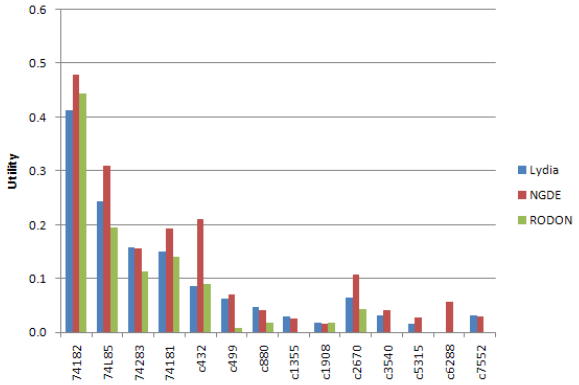

Fig. 10. Synthetic track DA utility scores by circuit