Second International Diagnostic Competition – DXC'10

Scott Poll[1], Johan de Kleer[2], Alexander Feldman[3], David Garcia[2], Tolga Kurtoglu[2] and Sriram Narasimhan[4]

[1] *NASA Ames Research Center, Moffett Field, CA, 94035, USA*
scott.poll@nasa.gov

[2] *Palo Alto Research Center, Palo Alto, CA, 94304, USA*
dekleer@parc.com
dgarcia@parc.com
kurtoglu@parc.com

[3] *Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud, CH-1401 Yverdon-les-Bains, Switzerland*
alexander.feldman@heig-vd.ch

[4] *University of California, Santa Cruz @ NASA Ames Research Center, Moffett Field, CA, 94035, USA*
sriram.narasimhan-1@nasa.gov

ABSTRACT

A framework to compare and evaluate diagnosis algorithms (DAs) has been created jointly by NASA Ames Research Center, Palo Alto Research Center, and Delft University of Technology. In this paper, we present the second implementation of this framework in a competition called DXC'10. The overall goal of this competition is to evaluate the performance of different diagnostic methods. In order to accurately mimic diagnostic technology use in a real-world context, we have defined diagnostic problems driven by use cases representing different roles of diagnosis results. In the end, the competition pitted seven DAs competing in two diagnostic problems. The paper presents the systems used in DXC'10, a description of faults and data sets used for each diagnostic problem, a listing of participating DAs, the performance metrics and results computed from running the DAs with the framework, and an analysis of the results.

1 INTRODUCTION

The problem of detecting and isolating faults (also called diagnosis) in physical systems has led to various solution approaches including expert, model-based, data-driven, and stochastic reasoning methods. However, there have been few efforts to evaluate and compare these different approaches in a standardized way. NASA Ames Research Center (ARC), Palo Alto Research Center (PARC), and Delft University of Technology decided to combine efforts to create a generalized framework that would establish a common platform to evaluate and compare diagnosis algorithms (Kurtoglu *et al.*, 2009a). The objectives for developing this framework are to accelerate research in theories, principles, and computational techniques for monitoring and diagnosis of complex systems, to encourage the development of software platforms that promise more rapid, accessible, and effective maturation of diagnostic technologies, and to provide a forum that can be utilized by algorithm developers to test and validate their technologies on real-world physical systems.

The First International Diagnostic Competition (DXC'09) was the first implementation of the above-mentioned framework (Kurtoglu *et al.*, 2009b). The overall goal of the competition was to systematically evaluate diagnostic technologies and to produce comparable performance assessments for different diagnostic methods. DXC'09 pitted 12 diagnosis algorithms competing in 3 different tiers on 2 different tracks (industrial and synthetic). In each tier, the DAs were provided a description of the system being diagnosed and sample data sets from nominal and faulty runs. Several metrics that covered timing, technical, and computational performance were computed and a single final ranking score was calculated to determine the winners in each tier.

Continuing the effort, we conducted the Second International Diagnostic Competition (DXC'10) this year. Based on the feedback from last year's

participants we instituted several changes. The primary change was in the evaluation criteria. Last year we used several metrics which were consolidated into a single ranking score. In order to accurately represent real-life problems, we defined use cases indicating what the role of the diagnosis results would be. We chose a decision support use case where the diagnosis result would be used to decide what recovery action(s) should be performed in order to minimize mission costs, which may include loss of mission objectives and/or loss of vehicle. We also added incipient faults and intermittent faults to make the problems more challenging. This paper describes the competition and presents the results.

Section 2 gives an introduction the DXC framework. Section 3 describes the diagnostic problems that were presented to the competitors. Section 4 lists which kinds of faults were injected. Section 5 explains how the evaluation was done to decide winners. Section 6 presents the results including metrics used in last year's competition.

2 DXC FRAMEWORK

The DXC framework allows systematic evaluation and comparison of diagnostic algorithms under identical experimental conditions. The key components of this framework include representation languages for the physical system description, sensor data and diagnosis results, a runtime architecture for executing DAs and diagnostic scenarios, and an evaluation component that computes performance metrics based on the results from diagnostic algorithm execution.

We provide a summary of the DXC framework (DXF) in this paper with emphasis on additions for DXC'10 and refer the reader to (Kurtoglu *et al.*, 2009a; Feldman *et al.*, 2010) for additional details. Figure 1 shows an overview of the DXC software components and the primary information flows. All communication is ASCII-based, and all the modules communicate via TCP ports by using a simple message-based protocol. Next, we provide a brief description of each software component.

Scenario Loader: Executes the Scenario Data Source, Recorder, and Diagnostic Algorithm. Scenario Loader ensures system stability and clean-up upon scenario completion. This is the main entry point for performing a diagnostic experiment.

Scenario Data Source: Provides scenario data from previously recorded datasets. The provenance of the data (whether hardware or simulation) depends on the system in question. A scenario dataset contains sensor readings, commands (note that the majority of classical MBD literature does not distinguish commands from observations), and fault injection information (to be sent exclusively to SR). Scenario Data Source publishes data following a wall-clock schedule specified by timestamps in the scenario files.

Scenario Recorder: Receives fault injection data and diagnosis data into a scenario results file. The results file contains a number of time-series which are used by the evaluation module for the computation of metrics. Scenario Recorder is the main timing authority, i.e., it timestamps each message upon arrival before recording it to the results file.

Diagnosis Algorithm: A DA receives sensor and command data, performs diagnosis, and sends the diagnosis results back. As long as the DAs comply to the provided API, there are no restrictions on a DA; for example, a DA may read precompiled data, or use external (user supplied) libraries, etc.

Diagnostic Oracle: Provides a querying capability to the DAs in one of two ways: 1) It takes a diagnostic output produced by a DA and returns the lowest cost action(s) associated with the provided diagnosis, or 2) it takes a diagnostic output and specific actions produced by a DA and returns the corresponding cost.

Evaluator: Takes scenario result file and applies metrics to evaluate DA performance. The metrics and evaluation procedures are detailed in Section 5.

Figure 2 shows a diagnostic session sequence diagram. Note the introduction in this year's competition of the oracle and the messages with which a DA requests recovery information/costs and recommends recovery actions.

3 DIAGNOSTIC PROBLEMS

Three diagnostic problems were announced for DXC'10, two industrial track problems and one synthetic. We report only on the industrial track diagnostic problems (DP-I, DP-II) in this paper since no teams participated in the synthetic track.

The hardware system for DXC'10 diagnostic problems I and II 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.

We have created two systems from the same physical testbed, ADAPT-Lite and ADAPT, for use in diagnostic problems I and II, respectively.

ADAPT-Lite: Includes a single battery, two AC loads and one DC load as shown in Figure 3. The initial configuration for ADAPT-Lite has all relays and circuit breakers closed and no nominal mode changes are commanded during the scenarios. Hence, any significant changes in sensor values may be correctly attributed to faults injected into the scenarios. ADAPT-Lite is restricted to single faults. For this year's competition we introduce simplified representations of drift and intermittent faults in addition to the abrupt parametric (abrupt changes in parameter values) and discrete (unexpected changes in system state) faults used in the First International Diagnostic Competition. Figure 4 illustrates the parametric fault profiles used in ADAPT-Lite.

ADAPT: Includes all batteries and loads in the EPS as shown in Figure 5. The initial configuration for ADAPT 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. Multiple faults may be injected in ADAPT. However, the fault types are restricted to abrupt parametric and discrete, similar to last year's competition.

DXC'10 introduces multi-rate data of 1, 2, and 10 Hz for both systems (rather than all data at 2 Hz) and includes fewer sensors than DXC'09. Reducing the sensor set resulted in ambiguity groups, for which it was not possible to isolate the injected fault. For example, in diagnostic problem I there were four ambiguity groups: (i) AC483 failed off and EY272 stuck open, (ii) FAN416 failed off and EY275 stuck open, (iii) DC485 failed off and EY284 stuck open, and (iv) INV2 failed off and CB262 failed open. In each case however, the recovery action is the same for both faults in the ambiguity group. Ruling out guessing, a perfect DA would have non-zero classification errors because of the ambiguity groups. Table 1 summarizes the main characteristics of the diagnostic problems, which we describe next.

Table 1: Diagnostic problem characteristics

Aspect		**DP-I**	**DP-II**
system		ADAPT-Lite	ADAPT
operational scenario		single-string UAS mission	redundant systems UAS mission
diagnostic use case		abort rec.	fault recovery rec.
#comps		25	96
#modes		102	306
initial relay state		closed	open
initial circuit breaker state		closed	closed
nominal mode changes		no	yes
multiple faults		no	yes
fault types	offset	yes	yes
	drift (incipient)	yes	no
	intermittent offset	yes	no

3.1 Diagnostic Problem I

Diagnostic problem I mimics the use of the ADAPT-Lite system in a single-string UAS (Unmanned Aircraft System) mission. The primary objective of the diagnostic algorithm in this operational scenario is to provide decision support to a remote pilot or an autonomous controller by making an "abort" recommendation. An abort recommendation would result in aborting the mission and landing the UAS vehicle.

This problem has a non-redundant EPS configuration which is supplying power to vehicle systems and payloads necessary for successful mission completion. There is only one path from one power source to two AC loads and one DC load. There are two possible recommendations for the nominal and single-fault scenarios in this diagnostic problem: "none" (or no-abort) and "abort".

The correct recommendation for a scenario depends on the injected failure mode and, for certain failure modes, the fault parameters. Any failure which cuts off power to any of the three loads results in an abort recommendation. Other failure modes may lead to

degraded performance which can be tolerated if the fault magnitude is below some threshold. For these scenarios, giving the correct recommendation requires isolating the failure mode and estimating the fault parameter(s).

3.2 Diagnostic Problem II

Diagnostic problem II mimics the use of the ADAPT system in a redundant systems UAS mission. The primary objective of the diagnostic algorithm in this operational scenario is to provide decision support to a remote pilot or an autonomous controller by making recommendations for fault recovery actions.

This problem has redundant power configurations of the EPS system that support mission and vehicle critical loads. There are multiple possible paths from power sources to the loads. Some loads are critical to vehicle operations, some are critical for mission success, and others are considered non-critical. For example, a vehicle critical load may be an avionics computer, while a mission critical load may be a sensor payload such as an IR camera.

Nominal configurations for this problem provide power to five critical loads and four non-critical loads. The system employs passive redundancy for the critical load functions such that there are two identical loads for each critical function located on opposite load banks. Only one of two redundant loads will be on in any given scenario and will remain on for the duration of a nominal scenario while the non-critical loads may be turned on and off.

The correct recommendation for a scenario restores power to all critical loads, which may require multiple recovery actions, or suggests an abort when it is not possible to do so.

4 FAULT INJECTION AND SCENARIOS

Experimental scenarios of approximately four minutes in length were acquired using the ADAPT EPS testbed for diagnostic problems I and II. The testbed allows for the repeatable injection of faults into the system in three ways: hardware-induced faults (e.g., turning off inverters, tripping circuit breakers, manipulating loads); software-induced faults (e.g., sending extraneous relay commands or blocking intended relay commands); introduction of faulty components (e.g., inserting a burned out light bulb). The first two methods were used for DXC'10.

The ADAPT-Lite experiments for DP-I include 20 nominal and 134 single-fault scenarios. The parametric fault profiles illustrated in Figure 4 are injected for sensor faults as well as AC and DC load faults. For the latter, a programmable electronic load was used to vary the AC and DC load resistances. Of the 154 scenarios, 65 are "abort" and 89 are "none" cases. Table 2 summarizes the faults and scenarios used for DP-I.

Table 2: Diagnostic problem I faults

Type	Subtype	Fault	#	abort / none
nominal		no fault	20	0 / 20
battery		degraded	1	0 / 1
circ. breaker		failed open	4	4 / 0
inverter		failed off	1	1 / 0
load	fan	failed off	1	1 / 0
		over speed	1	1 / 0
		under speed	1	0 / 1
	AC load	resistance offset	8	4 / 4
		resistance drift	8	4 / 4
		intermittent res. offset	7	3 / 4
		failed off	1	1 / 0
	DC load	resistance offset	8	4 / 4
		resistance drift	9	4 / 5
		intermittent res. offset	8	4 / 4
		failed off	1	1 / 0
relay		stuck open	5	5 / 0
sensor	position	stuck	2	0 / 2
	current, temp., voltage	offset	20	8 / 12
		stuck	8	4 / 4
		drift	20	8 / 12
		intermittent offset	20	8 /12
		Totals:	154	65 / 89

The ADAPT experiments for DP-II include 20 nominal and 100 single, double, and triple-fault scenarios. The scenarios required 0 to 4 recovery commands or the abort command as shown in Table 3. The faults used for DP-II are summarized in Table 4.

Table 3: Diagnostic problem II scenario distribution, fault cardinality, and recovery commands

		#Recovery Commands (A=abort)					
Card	**#scn**	0	1	2	3	4	A
no fault	20	20	0	0	0	0	0
single fault	31	12	11	7	1	0	0
double fault	35	11	4	7	6	2	5
triple fault	34	10	4	8	6	1	5
Totals:	120	53	19	22	13	3	10

Table 4: Diagnostic problem II faults

Type	**Subtype**	**Fault**	**#**
nominal		no fault	20
battery		degraded	4
circ. breaker		failed open	22
inverter		failed off	9
load	basic	failed off	8
	fan	failed off	3
		over speed	4
		under speed	2
	light bulb	failed off	18
	pump	failed off	4
		flow restricted	2
relay		stuck closed	4
		stuck open	37
sensor	position	stuck	10
	current, temp., voltage	offset	38
		stuck	38
		Totals:	223

5 EVALUATION

Diagnostic algorithms are ranked using a decision cost metric, M_{dc}. The decision cost is the cost incurred had the DA's recommendation(s) been acted upon. Additionally, we compute the metrics used in DXC'09 to evaluate how changes introduced in DXC'10 affect DA performance.

For DP-I, the two main categories of costs are cost of losing the vehicle and cost of not completing the mission. In this use case the DA is only responsible for deciding if a mission should be aborted or not. Hence there are 4 outcomes (2 answers from the DA versus 2 actual situations). Let the cost of losing the mission be $c_{mission}$ and the cost of losing the vehicle be $c_{vehicle}$. Table 5 computes the costs incurred in each of the 4 possible outcomes.

Table 5: Diagnostic problem I decision costs, M_{dc}

DA rec. \ Actual Case	abort	non-abort
abort	0	$c_{mission}$
non-abort	$c_{mission} + c_{vehicle}$	0

For DP-II, in addition to $c_{mission}$ and $c_{vehicle}$, there is a cost associated with performing each recovery action, c_{action}. When a DA recommends a set of recovery actions, it automatically incurs the cost of performing those actions. In addition, it may incur the cost of losing the mission or losing the mission and vehicle if the recovery actions do not mitigate the effects of the faults. Note that for DP-II we treat the abort action as incurring a cost equal to the cost of the mission, so a correct recommendation may have non-zero cost, in contrast to DP-I.

For both DP-I and DP-II evaluations, $c_{mission}$ is set to 25, $c_{vehicle}$ is set to100, and each action, c_{action} is set to 1. The metrics used for evaluating DAs in DXC'09 are summarized in Table 6. Please see (Kurtoglu *et al.*, 2009; Feldman *et al.*, 2010) for detailed definitions and related discussion. Note that DXC'09 metric M_{ia} has been renamed M_{err} in the table below. The metrics in the table are per scenario metrics. To calculate "per system" metrics, an unweighted average is taken over all scenarios and indicated with an overbar.

Table 6: DXC'09 metrics summary

Metric	**Name**	**Class**
M_{fd}	fault detection time	detection
M_{fn}	false negative scenario	detection
M_{fp}	false positive scenario	detection
M_{da}	scenario detection accuracy	detection
M_{fi}	fault isolation time	isolation
M_{err}	classification errors	isolation
M_{cpu}	CPU load	computation
M_{mem}	memory load	computation

Diagnostic algorithms were evaluated using the DXC framework on two computers with identical hardware (Intel® XEON™ 2x2.20Ghz, 3.60 GB RAM), one running Windows™ and the other Linux. The choice of target operating system was left to DA developers.

6 RESULTS

Using the evaluation approach described in the previous section, we computed metrics and rankings for 7 diagnostic algorithms that participated in the Second International Diagnostic Competition.

6.1 Diagnostic Algorithms

The teams that participated in diagnostic problems I and II are listed in Table 7. In what follows we provide a brief description of each DA.

Table 7: DXC'10 participating DAs

DA	DP	Algorithm Type
AdaptedFACT	I	Model-based
HyDE-A	I, II	Model-based
ProADAPT	I, II	Probabilistic
QED	I	Model-based
SystemicsC	I	Model-based
TARDEC	I	Rule-based
TRT4ADAPT	II	Flow-models

1. AdaptedFACT: Uses a model-based diagnosis approach. The front end nominal model was mostly static with a more dynamic battery model. Incoming data for each sensor along with predictions from the model are combined with a Generalized Likelihood Ratio detector that is designed to detect a Gaussian shift-in-mean. The fault isolation uses pre-derived fault signatures (the signatures were generated by inspecting the model and system equations), much like the TRANSCEND approach (Mosterman and Biswas, 1999). The signatures are combined with special condition checking as well as heuristics, to differentiate between partial signatures.
2. 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 (Narasimhan and Brownston, 2007). A HyDE model may use Boolean, discrete, real, or interval-valued variables to describe the behavior of a system. HyDE-A uses discrete models of the system and a discretization of the sensor observations for diagnosis.
3. ProADAPT: Uses a probabilistic approach to diagnosis, based on Bayesian networks (BNs). Real and virtual sensors are represented in a BN, as are health nodes which provide the health status of the sensors and components that are diagnosed. Off-line, the BN is compiled to an equivalent arithmetic circuit. On-line, the arithmetic circuit is used to quickly compute, based on sensor inputs, the posterior distribution over the health nodes, which is used to compute diagnoses. (Ricks and Mengshoel, 2009; Ricks and Mengshoel, 2010)
4. QED: A model-based diagnosis system based on qualitative event-based fault isolation. Statistically significant deviations of measured from model-predicted values imply the presence of faults. These deviations are abstracted into symbolic event-based descriptions of fault-induced behavior, which are compared to predicted event sequences to isolate faults. Fault identification uses quantitative methods to compute fault parameters and further refine fault hypotheses (Daigle and Roychoudhury, 2010).
5. SystemicsC: A multi-modal reasoning system that combines Case-Based Reasoning (CBR) with Model-Based Reasoning (MBR) in the context of diagnostic problem solving process. SystemicsC is based on the principles of the General Diagnostic Engine (GDE) as described by (de Kleer and Williams, 1987). Contradictions (conflicts) between the simulated and the observed behavior are used to generate hypotheses about possible causes for the observed behavior. The failure modes of the model components together with previously recorded faults are then used to verify the hypotheses and speed up the diagnostic process.
6. TARDEC: A real-time, two-tier system which first utilizes statistical models to detect and categorize sensor faults, then aggregates all detected sensor-level faults and associated characteristics and, through a rule-based expert system, identifies the electrical system component most likely to have produced those faults, as well as estimates of the significant parameters of its failure.
7. TRT4ADAPT: A model-based diagnostic system based on a casual dependency graph of fault causes and fault effect propagation paths. It is a combination of the Testability Engineering and Maintenance System (TEAMS) software created by Qualtech Systems Incorporated (QSI) and custom wrapper code that facilitates the exchange of data between test results and the real-time reasoning engine. It uses a set of custom designed tests that facilitates the transformation of a graphical flow model of the system into a matrix representation describing the relationship between faults and test points for a given mode of the system. This representation contains the basic information needed to interpret test results and diagnose failures during operations (Wilson and Kurtoglu, 2010).

6.2 Diagnostic Problem I

The results for DP-I are shown in Table 8; diagnostic algorithm TARDEC had the lowest cost and is the winner. For comparison, a DA that always recommends abort would have received a cost of 89 * 25 = 2225; a DA that always recommends no-abort would have received a cost of 65 * 125 = 8125. HyDE-A used the same model as last year, which did not classify drift, intermittent, offset or stuck faults (except when the value went to 0) so it is not surprising that it is close to a no-abort DA.

We show the breakdown of costs by fault type for each DA in Figure 6. Offset, drift, and intermittent faults include hardware (AC483, DC485) and sensor (e.g., IT267, IT281, etc.) fault injection scenarios. Category "other" includes battery, circuit breaker, inverter, fan, and AC and DC load failed-off fault scenarios. It is interesting to note that TARDEC did not incur any costs in the category "other". While it did not classify all faults in this category correctly (see Figure 7), the misclassifications resulted in the correct recommendation.

Table 8: Diagnostic problem I scores

DA	Cost	Rank
AdaptedFACT	2250	2
HyDE-A	6950	6
ProADAPT	4925	5
QED	2350	3
SystemicsC	2400	4
TARDEC	2000	1

The metrics used for scoring DAs in the First International Diagnostic Competition are computed and shown in Table 10. Similar to last year, no DA dominates all metrics. The fault detection and isolation times are noticeably higher than last year, primarily because of the need to accumulate more evidence in the case of drift and intermittent fault types. Diagnostic algorithm QED had much smaller CPU load compared to the other algorithms.

Note that the ranking of the DAs with respect to the detection accuracy metric exactly matches, from top to bottom, the ranking by the cost metric. The DA with the best detection accuracy, TARDEC, had the lowest cost. Furthermore, TARDEC also had the fewest classification errors. It is logical that the algorithm which most often correctly detects and classifies the injected fault in the scenario would receive the proper fault response from the Oracle, and consequently have the lowest cost.

Figure 7 shows the breakdown of classification errors by fault type. In a scenario, the number of classification errors is the number of misclassified components. Figure 8 shows the contribution to classification errors from scenario detection categories false positive (FP), false negative (FN), and true positive (TP). DAs HyDE and TARDEC had no false positives. We noticed some scenarios in which DAs SystemicsC and TARDEC had false negatives and a CPU load of 0 for the same scenarios. We were not able to investigate this more thoroughly prior to publication; this could indicate some errors in the evaluation.

In general, drift faults presented the most difficulty to DAs. In some drift fault scenarios the fault was isolated to the incorrect component or incorrect failure mode, which resulted in a recovery recommendation that was not appropriate for the actual injected fault. In other drift fault scenarios, the isolation was correct but the estimation of the slope or the fault injection time was inaccurate. In particular, SystemicsC appears to have consistently overestimated the slope for positive drift faults, most likely the outcome of a simple calculation error.

Sensor ST516 proved especially difficult to diagnose correctly due to the low signal-to-noise ratio resulting from the quantized sensor readings. This sensor was responsible for many of the DA false positive scenarios, especially for QED, in which the incorrect diagnosis was ST516=stuck. Stuck faults were incorrectly indicated by a few DAs for other sensors as well. This likely reflects slight differences between the training data set and competition data set, which resulted in thresholds for counts of consecutive identical sensor values being too small.

Diagnostic algorithm ProADAPT had some calculation or output error which set the offset parameter to 0 for all resistance offset faults, resulting in several missed aborts.

6.3 Diagnostic Problem II

The results for DP-II are shown in Table 9; diagnostic algorithm ProADAPT had the lowest cost and is the winner. The minimum possible score for DP-II is 364, the sum of the recovery action costs for all scenarios. The breakdown of cost by the number of actions required to restore critical functionality (or abort in the case critical functionality cannot be restored) is shown in Figure 9. Note that the costs for the category "0 cmd" are negligible but non-zero (HyDE-A = 10, ProADAPT = 11, TRT4ADAPT = 3).

Table 9: Diagnostic problem II scores

DA	Cost	Rank
HyDE-A	3471	3
ProADAPT	1590	1
TRT4ADAPT	1676	2

Additional DA performance metrics are shown in Table 11. Similar to the first diagnostic problem, the

Table 10: Diagnostic problem I additional metrics

DA	$\overline{M}_{fd}$ (s)	$\overline{M}_{fn}$	$\overline{M}_{fp}$	$\overline{M}_{da}$	$\overline{M}_{fi}$ (s)	$\overline{M}_{err}$	$\overline{M}_{cpu}$ (ms)	$\overline{M}_{mem}$ (kb)
AdaptedFACT	21.462	0.069	0.040	0.901	151.746	98.000	37189	9656
HyDE-A	27.717	0.873	0.000	0.240	29.355	136.030	1550	6463
ProADAPT	15.990	0.179	0.019	0.825	64.711	171.000	6356	4373
QED	7.307	0.015	0.105	0.882	115.499	71.752	239	5364
SystemicsC	9.390	0.134	0.026	0.856	13.860	73.000	229057	3151
TARDEC	162.638	0.090	0.000	0.922	162.638	58.000	8979	3211

Table 11: Diagnostic problem II additional metrics

DA	$\overline{M}_{fd}$ (s)	$\overline{M}_{fn}$	$\overline{M}_{fp}$	$\overline{M}_{da}$	$\overline{M}_{fi}$ (s)	$\overline{M}_{err}$	$\overline{M}_{cpu}$ (ms)	$\overline{M}_{mem}$ (kb)
HyDE-A	31.788	0.360	0.000	0.700	29.127	216.416	4706	14172
ProADAPT	6.970	0.040	0.075	0.892	35.878	151.000	11405	10482
TRT4ADAPT	10.766	0.062	0.265	0.684	37.853	220.995	6135	16819

DA with the lowest cost, ProADAPT, also had the best detection accuracy and fewest classification errors.

The fact that isolation time can be less than detection time for DP-II, as it is for HyDE-A, requires explanation. There is one detection time possible per fault scenario, which is measured from the time the first fault is injected to the time when the DA sets the fault detection Boolean high. For multiple fault scenarios, we have allowed for more than one isolation time as follows. Suppose in a double fault scenario the first fault is injected at 100 seconds and the second fault is injected at 200 seconds. Furthermore, the DA detects a fault at 110 seconds. By performing a string comparison of the candidate set, it is determined that the DA converges on a diagnosis at 115 seconds. It then changes its candidate set after the second fault is injected and converges to a new diagnosis at 201 seconds, which persists to the end of the scenario. There are two isolation times defined for this scenario, 15 seconds (115 – 100) and 1 second (201 – 200). An average of all isolation times is taken over the scenario, in this case 8 seconds. The detection time is 10 seconds. Hence, in this hypothetical scenario, the isolation time is less than the detection time.

Figure 10 shows the contribution to classification errors from scenario detection categories false positive (FP), false negative (FN), and true positive (TP). We noticed a few scenarios in which the DA had no classification errors but yet incurred a cost of losing the vehicle and mission. Upon closer inspection, these scenarios terminated after the allotted time and before the DA could communicate the recovery actions to the Scenario Recorder. Allowing more time for the interactions between DA, Oracle, and Scenario Recorder should solve this problem. We also observed that repeated queries to the Oracle results in system lag.

7 CONCLUSION

We presented the implementation of the Second International Diagnostic Competition, DXC'10. We noted recommendations from the first competition and tried to make changes accordingly. Identifying realistic use case scenarios was difficult and we spent a lot of time discussing different possibilities. This represents just one use case among many different applications of diagnosis results.

We hope that this work can be continued moving forward by identifying new physical systems and new diagnostic problems. The DXC framework provides an easy way to create an evaluation platform for new problems. Our sincere hope is that the framework is adopted by a growing number of people and applied to a wide variety of physical systems including diagnosis algorithms from several different research communities. The long-term goal is to create a database of performance evaluation results which will allow system designers to choose the appropriate DA for their system given the constraints and metrics in their application.

ACKNOWLEDGMENT

We extend our gratitude to Michael Wilson (Embry-Riddle Aeronautical University), Adam Sweet (NASA), David Nishikawa (NASA), Craig Harrison (University of Maine), Ole Mengshoel (Carnegie Mellon University), all DXC'10 competitors, and many others for valuable discussions, criticism and help.

REFERENCES

(Daigle and Roychoudhury, 2010) M. Daigle and I. Roychoudhury. Qualitative Event-based Diagnosis: Case Study on the Second International Diagnostic Competition. In *Proceedings of 21st International Workshop on Principles of Diagnosis*, Portland, OR,

2010.

(de Kleer and Williams, 1987) J. de Kleer and B. C. Williams. Diagnosing Multiple Faults. *Artificial Intelligence*, 32(1):97-130, 1987.

(Feldman *et al.*, 2010) A. Feldman, T. Kurtoglu, S. Narasimhan, S. Poll, D. Garcia, J. de Kleer, L. Kuhn, A. van Gemund. Empirical Evaluation of Diagnostic Algorithm Performance Using a Generic Framework. In *International Journal of Prognostics and Health Management, Vol. 1 (2)*, 2010.

(Kurtoglu *et al.*, 2009a) 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*, Stockholm, Sweden, 2009.

(Kurtoglu *et al.*, 2009b) T. Kurtoglu, S. Narasimhan, S. Poll, D. Garcia, L. Kuhn, J. de Kleer, A. van Gemund, A. Feldman. First International Diagnosis Competition – DXC'09. In *Proceedings of 20th International Workshop on Principles of Diagnosis*, Stockholm, Sweden, 2009.

(Mosterman and Biswas, 1999) P. J. Mosterman and G. Biswas. Diagnosis of Continuous Valued Systems in Transient Operating Regions. In *IEEE Trans. on Systems, Man and Cybernetics, vol. 29, no. 6*, pp. 554-565, Nov. 1999.

(Narasimhan and Brownston, 2007) S. Narasimhan and Lee Brownston. HyDE – A General Framework for Stochastic and Hybrid Model-based Diagnosis. In *Proceedings of 18th International Workshop on Principles of Diagnosis*, Nashville, TN, 2007.

(Ricks and Mengshoel, 2009) B. Ricks and O. J. 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)*, Stockholm, Sweden, pp. 415–422, 2009.

(Ricks and Mengshoel, 2010) B. Ricks and O. J. Mengshoel. Diagnosing Intermittent and Persistent Faults using Static Bayesian Networks. In *Proceedings of 21st International Workshop on Principles of Diagnosis (DX-10)*, Portland, OR, 2010.

(Wilson and Kurtoglu, 2010) M. Wilson and T. Kurtoglu. TRT4ADAPT – A Model-Based Algorithm for the Second International Diagnostic Competition. In *Proceedings of 21st International Workshop on Principles of Diagnosis*, Portland, OR, 2010.

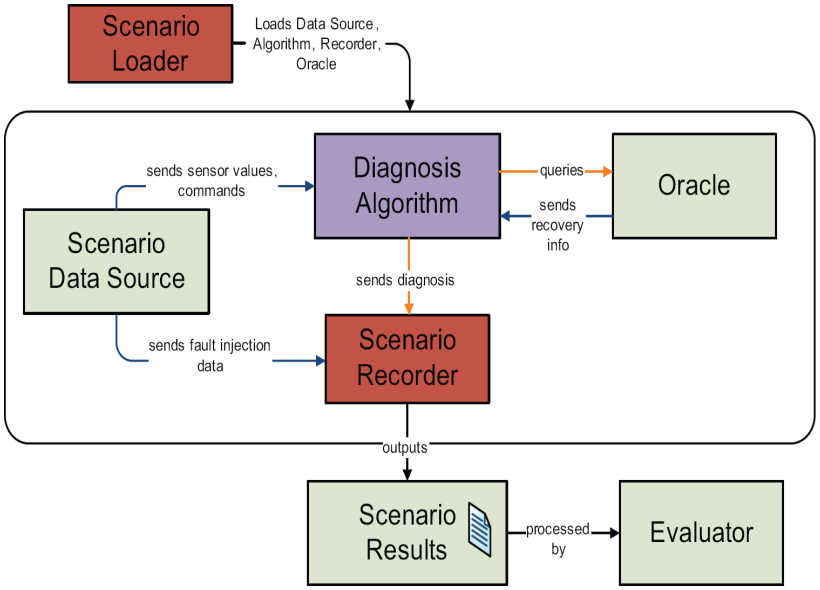


Figure 1: DXF run-time architecture.

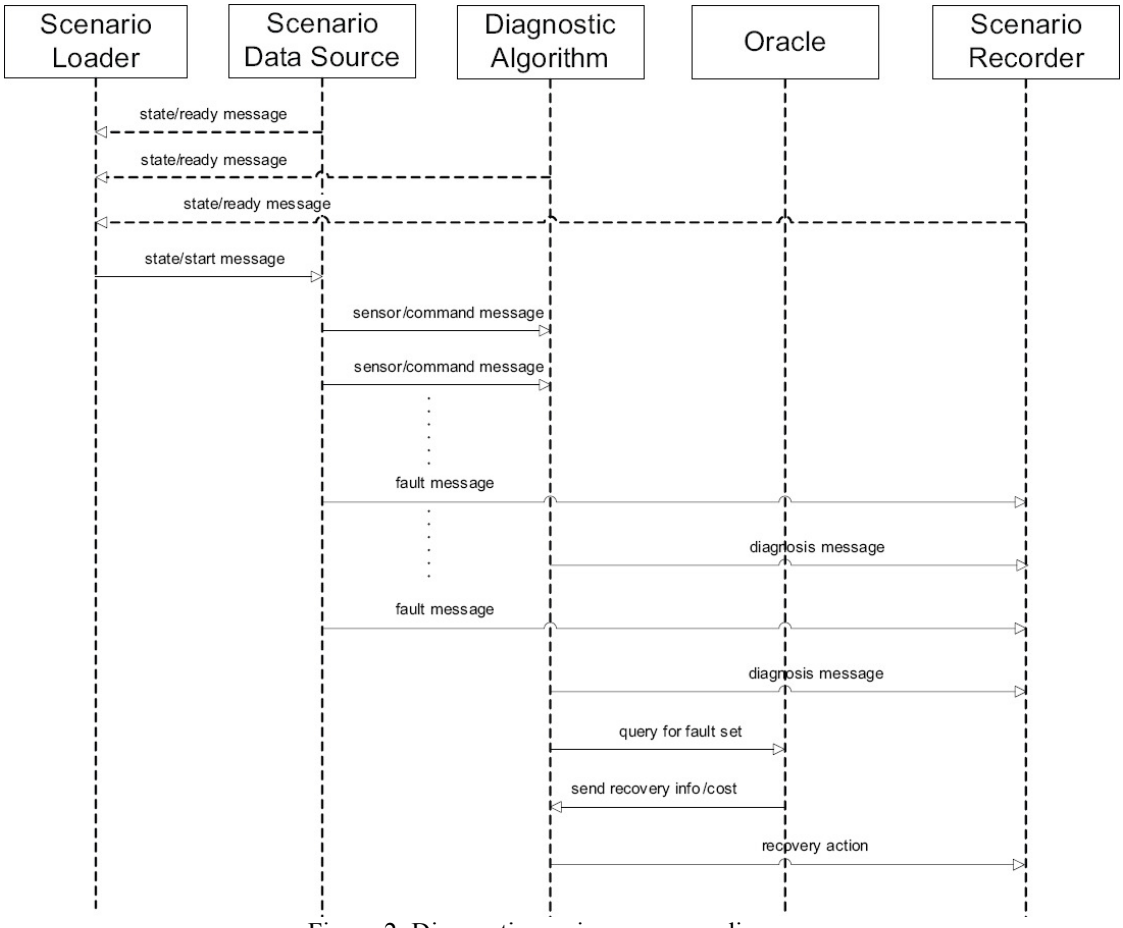


Figure 2: Diagnostic session sequence diagram.

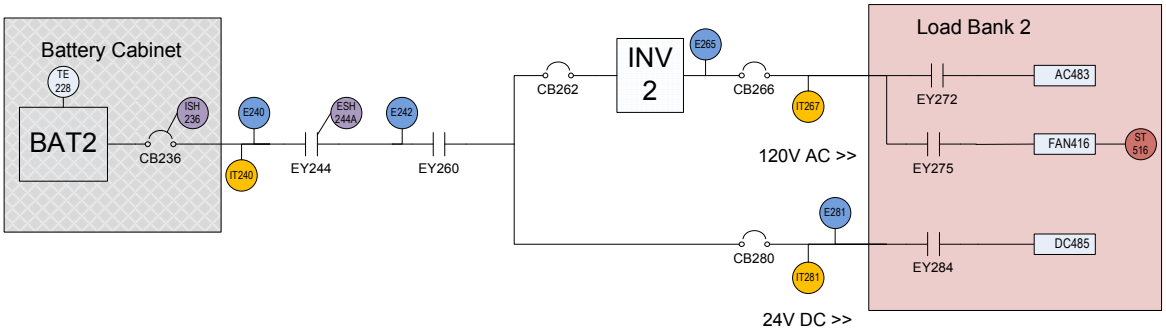
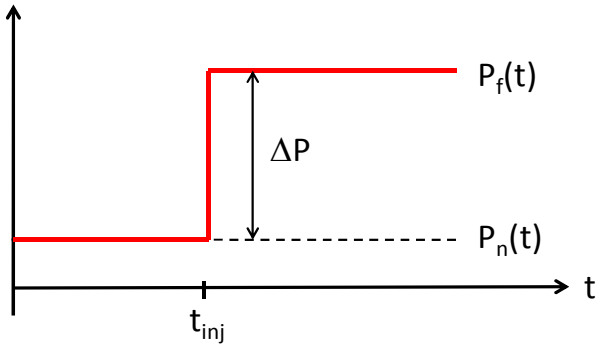
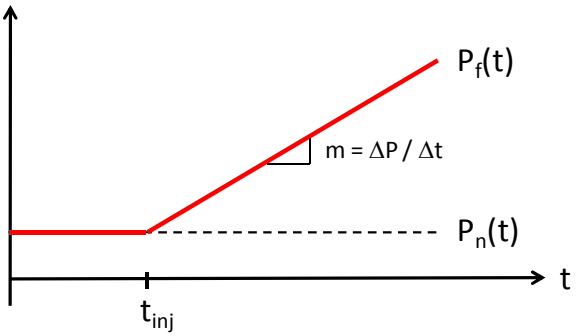


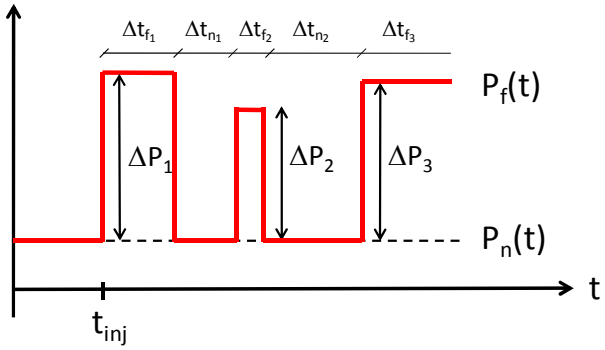
Figure 3: ADAPT-Lite system for diagnostic problem I.



a) Offset.



b) Drift.



c) Intermittent Offset.

Figure 4: Diagnostic problem I fault profiles.

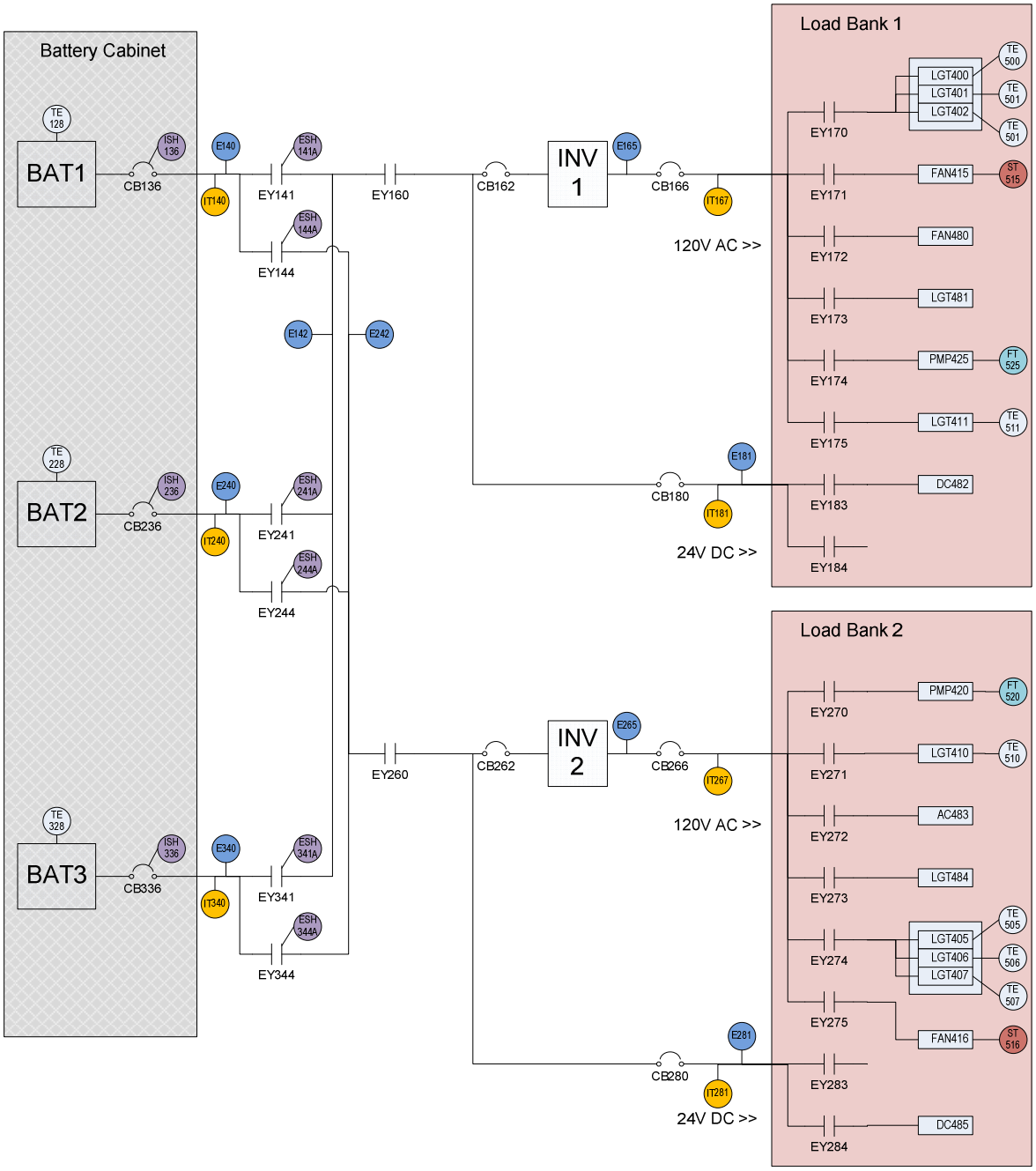


Figure 5: ADAPT system for diagnostic problem II.

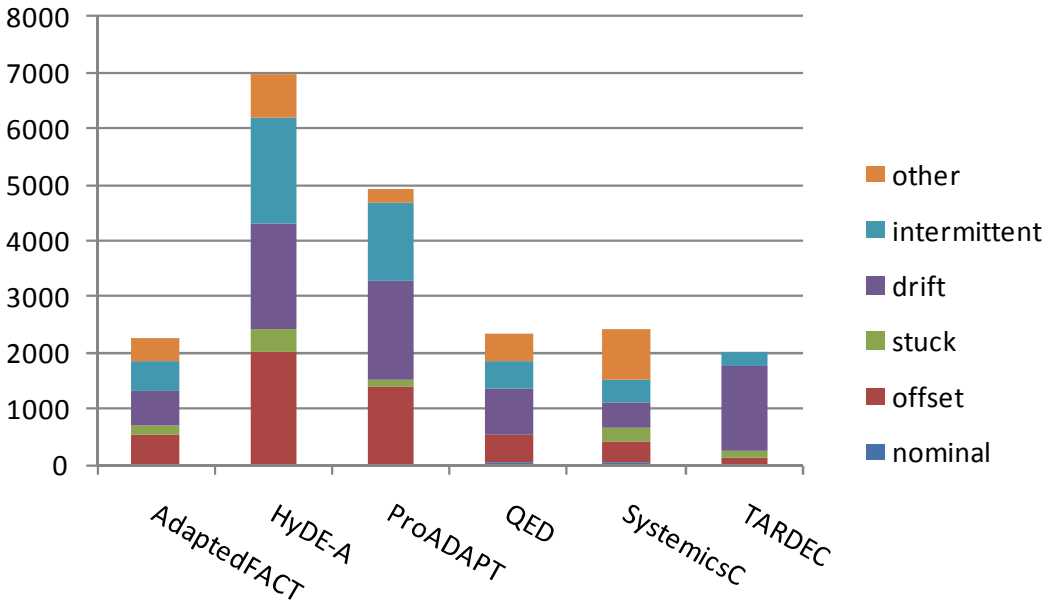


Figure 6: DP-I cost breakdown by scenario fault type.

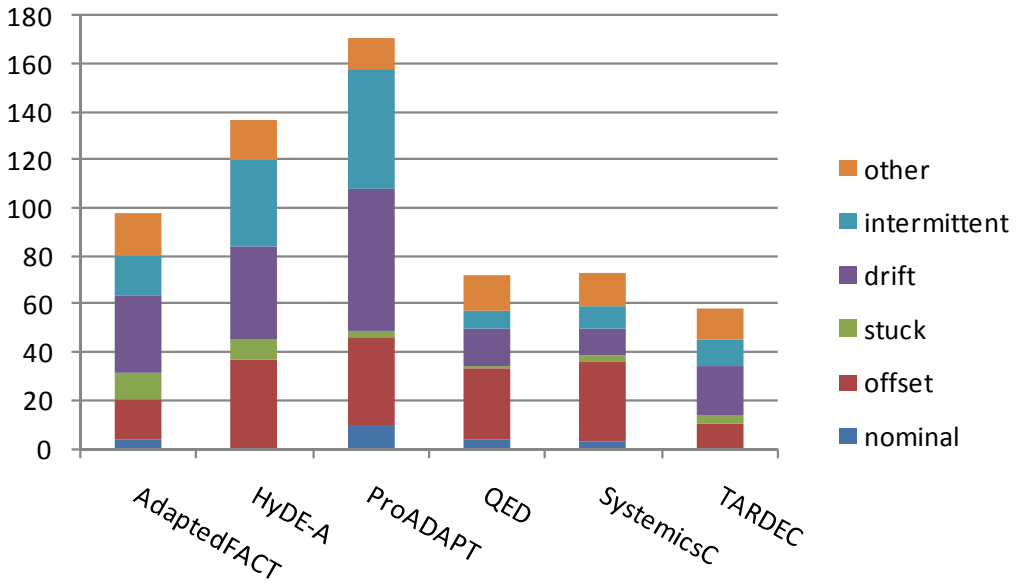


Figure 7: DP-I classification error breakdown by scenario fault type.

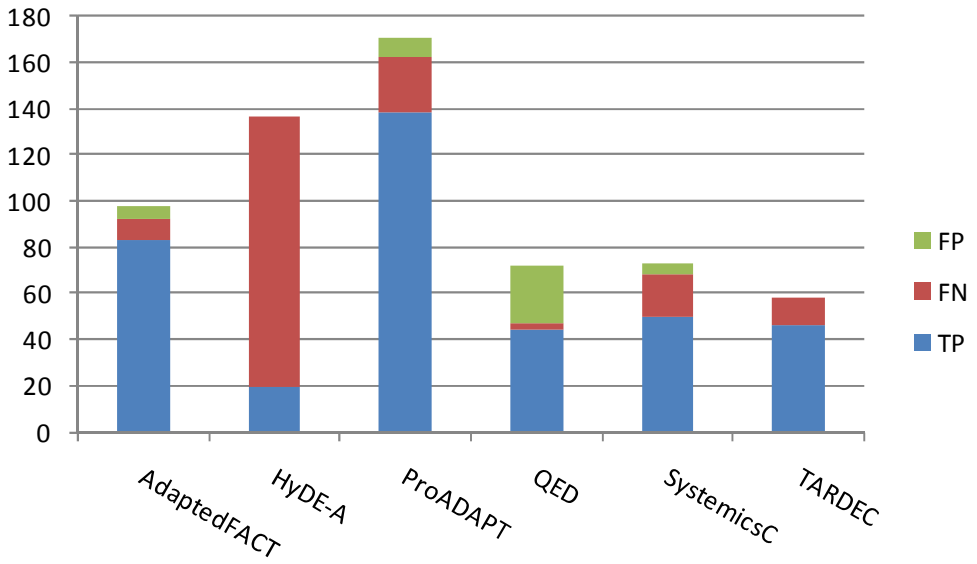


Figure 8: DP-I classification error breakdown by scenario detection type.

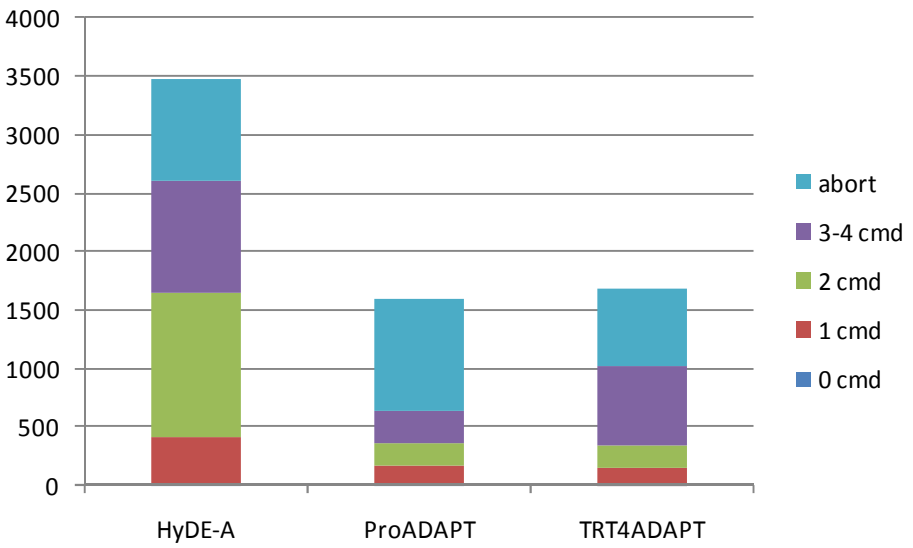


Figure 9: DP-II cost breakdown by scenario recovery type.

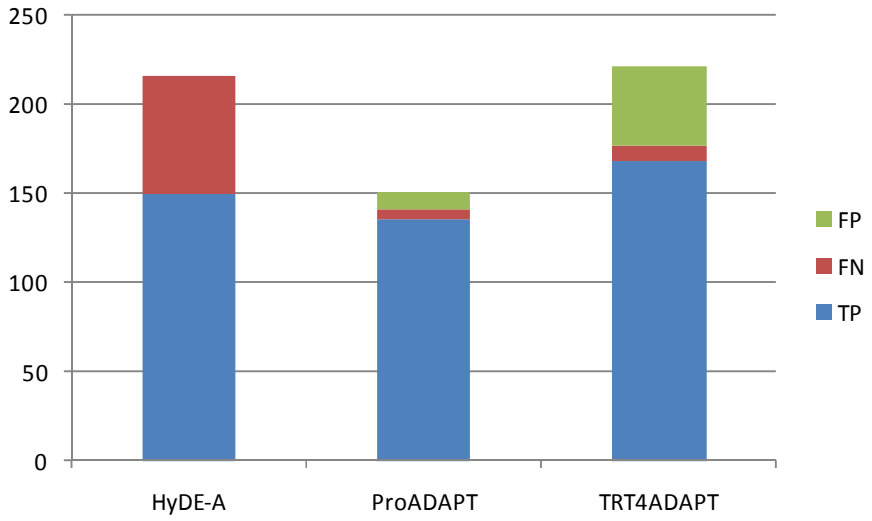


Figure 10: DP-II classification error breakdown by scenario detection type.