Qualitative Event-based Diagnosis with Possible Conflicts: Case Study on the Third International Diagnostic Competition

Matthew Daigle [1], **Anibal Bregon** [2], **and Indranil Roychoudhury** [3]

[1] *University of California, Santa Cruz, NASA Ames Research Center, Moffett Field, CA, 94035, USA*
matthew.j.daigle@nasa.gov
[2] *Department of Computer Science, University of Valladolid, Valladolid, 47011, Spain*
anibal@infor.uva.es
[3] *SGT Inc., NASA Ames Research Center, Moffett Field, CA, 94035, USA*
indranil.roychoudhury@nasa.gov

ABSTRACT

We describe two model-based diagnosis algorithms entered into the Third International Diagnostic Competition. We focus on the first diagnostic problem of the industrial track of the competition in which a diagnosis algorithm must detect, isolate, and identify faults in an electrical power distribution testbed in order to provide correct abort recommendations. Both diagnosis algorithms are based on a qualitative event-based fault isolation framework augmented with model-based fault identification. Although based on a common framework, the fundamental difference between the two algorithms is that one is based on a global model for residual generation, fault isolation, and fault identification, whereas the other uses a set of minimal submodels computed using Possible Conflicts. We describe, compare, and contrast the two algorithms in terms of practical implementation and their diagnosis results.

1 INTRODUCTION

This paper presents a model-based, qualitative, event-based fault diagnosis scheme that was entered into the Third International Diagnostic Competition (DXC'11). The competition allows for a comparative study of different diagnostic approaches, and includes multiple diagnostic problems. We focus on diagnostic problem I (DPI) of the industrial track of the competition, which consists of fault diagnosis and recovery for a subset of the Advanced Diagnosis and Prognosis Testbed (ADAPT) (Poll *et al*., 2007), called ADAPT-Lite, which is an electrical power distribution system. Our diagnosis scheme has two instantiations, QED (Qualitative Event-based Diagnosis), which is a revised version of the algorithm submitted to DXC'10 (Daigle and Roychoudhury, 2010), and QED-PC (QED with Possible Conflicts), which is a new entry that utilizes Possible Conflicts (Pulido and Alonso-González, 2004) as a basis for residual generation and fault identification.

QED extends the TRANSCEND diagnosis scheme described in (Mosterman and Biswas, 1999). In this scheme, fault isolation is achieved through analysis of the transients produced by faults, manifesting as deviations in observed behavior from predicted nominal behavior. We extend TRANSCEND by including relative measurement orderings, which provide a partial ordering of measurement deviations for different faults, leading to an enhanced event-based fault isolation scheme (Daigle *et al.*, 2009). DPI requires fault identification, and includes abrupt, incipient, and intermittent faults. TRANSCEND deals only with abrupt faults, so we incorporate methods for incipient faults (Roychoudhury, 2009) and intermittent faults (Daigle and Roychoudhury, 2010).

The second algorithm, QED-PC, uses the Possible Conflicts (PCs) diagnosis approach (Pulido and Alonso-González, 2004). This approach decomposes the global system model into minimal submodels containing sufficient analytical redundancy to generate fault hypotheses from observed measurement deviations. In this work, we combine the PCs together with the qualitative fault isolation ideas from TRANSCEND. Residuals are computed using the PCs (instead of the global system model) and measurement deviations are analyzed following the TRANSCEND ideas as in the previous scheme. Then, for fault identification, we use minimal parameter estimators computed from PCs for each faulty parameter as described in (Bregon *et al.*, to appear).

The paper is organized as follows. First, Section 2 overviews the diagnosis approaches. Section 3 provides the system model. Section 4 describes fault detection. Section 5 discusses fault isolation, and Section 6 describes fault identification. Section 7 covers fault recovery. Section 8 presents diagnosis results, and Section 9 concludes the paper.

2 DIAGNOSIS APPROACH

The problem for DPI is to decide whether a mission should be aborted or continued. To make this decision, the diagnostic algorithm must determine if the system is faulty, which fault has occurred, and the parameters that define the fault behavior.

The implemented diagnosis architecture is shown in Fig. 1, and reflects the implementation of both al-

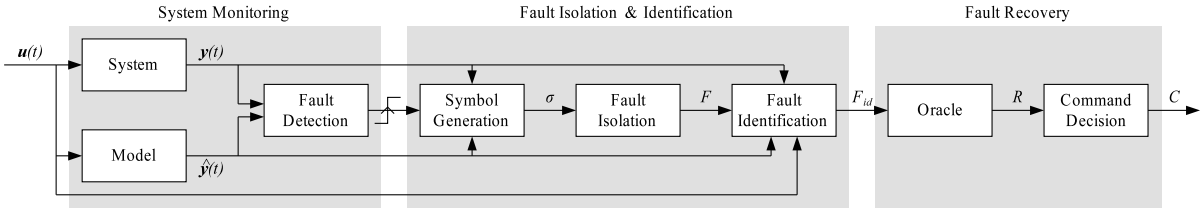


Figure 1: Diagnosis and recovery architecture.

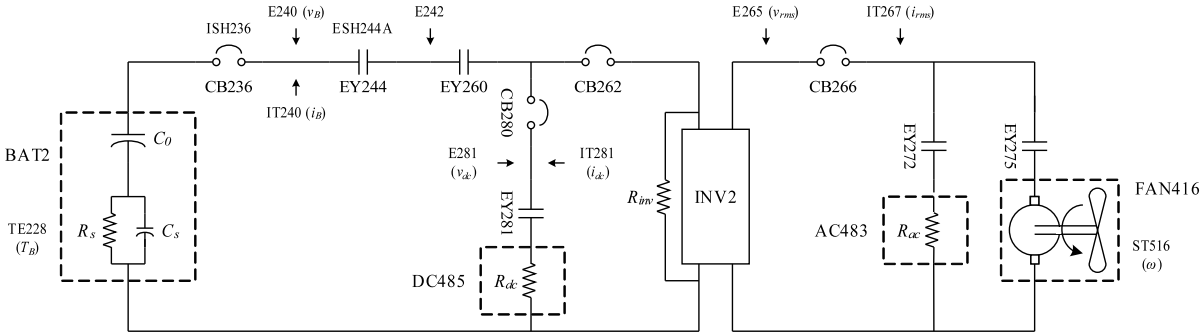


Figure 2: ADAPT-Lite schematic.

gorithms. The system receives inputs $\mathbf{u}(t)$ and produces outputs $\mathbf{y}(t)$. The system model, given inputs $\mathbf{u}(t)$, computes predicted values $\hat{\mathbf{y}}(t)$. The fault detection module decides whether a measurement has deviated from its nominal value in a statistically significant manner, triggering the fault isolation and identification modules. Measurement deviations, viewed as events, are abstracted into a symbolic representation using the symbol generator. The sequence of these symbols, where a symbol is denoted by σ, is used to isolate faults F. Fault isolation consists of candidate generation at the point of fault detection, and hypothesis refinement as new symbols are provided. Each fault $f \in F$ is associated with a component, a fault mode, and a set of fault parameters. Fault identification computes, for each fault $f \in F$, the values of the fault parameters. An oracle (provided by DXC'11) is called upon to decide for each fault f whether an abort is recommended, producing a set of recommendations R. The decision module selects a recommendation from R and outputs the associated control actions C.

3 SYSTEM MODELING

Our diagnosis approach is model-based, requiring a model of both nominal and faulty behavior for use throughout the diagnosis process. The two algorithms implement the nominal model in a different way. For QED, the nominal model is a global model of the system $\mathcal{M}$, and its inputs are those of the global system. For QED-PC, the nominal model is composed of a set of 11 minimal submodels, with each submodel $\mathcal{M}_i$ estimating the value of sensor i using a subset of the system measurements as input variables. In the following, we describe the models of nominal and faulty behavior of the ADAPT-Lite system for QED and QED-PC, indicating their similarities and differences.

3.1 Nominal Model

A schematic of ADAPT-Lite is given in Fig. 2. Sensors prefixed with an "E" are voltage sensors, those with an "IT" are current sensors, and those with "ISH" or "ESH" are for states of circuit breakers and relays, respectively. TE228 is the battery temperature sensor, and ST516 is the fan speed sensor. Note that the inverter converts DC power to AC, and E265 and IT267 provide rms values of the AC waveforms.

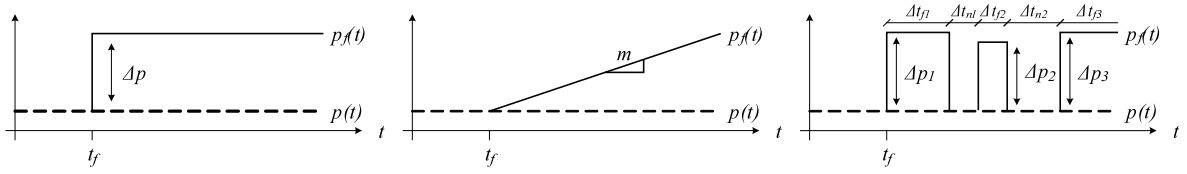
Most of the global model and the corresponding PCs are summarized in Table 1. Details may be found in (Daigle and Roychoudhury, 2010). Here, v_B and i_B are the battery voltage and current, v_0 is the voltage across C_0, v_s is the voltage across C_s, e is the inverter efficiency, v_{inv} is the inverter voltage on the DC side, R_{inv} is the DC resistance of the inverter, R_{dc} is the DC load resistance, J_{fan} is the fan inertia, and B_{fan} is a damping parameter. Both QED and QED-PC assume TE228, ISH236, and ESH244A are constant. The PCs for E242 and E281 are simply other measured voltages with a bias term added.

Most of the PCs are derived directly from the global model, but in some cases, the PCs had to account for additional dynamics. For example, the fan speed (ω) has no dynamics during nominal operation because it is always operated at the same speed. So, QED models the fan speed as a constant. QED-PC, on the other hand, must model the dynamics, because some faults independent of the fan submodel will cause the fan speed to decrease through a decrease in E265, which is an input to the PC.

A key difference, then, compared to the global model, is that the behavior of each PC has to be nominal not only for the nominal situation, but also for those faulty situations where the faulty parameters are independent of a PC. This decoupling requires a more

Table 1: Models for the ADAPT-Lite System.

Global Model (QED)				PCs (QED-PC)			
$\dot{v}_0$	$= \frac{1}{C_0}(-i_B)$	$\dot{v}_s$	$= \frac{1}{C_s}(i_B R_s - v_s)$	$\dot{v}_0$	$= \frac{1}{C_0}(-\text{IT240})$	$\dot{v}_s$	$= \frac{1}{C_s}(\text{IT240} R_s - v_s)$
v_B	$= v_0 - v_s$	i_{dc}	$= \frac{v_B}{R_{dc}}$	$\widehat{\text{E240}}$	$= v_0 - v_s$	$\widehat{\text{IT281}}$	$= \frac{\text{E281}}{R_{dc}}$
i_{inv}	$= \frac{v_{rms} i_{rms}}{e \cdot v_{inv}} + \frac{v_{inv}}{R_{inv}}$	i_B	$= i_{inv} + i_{dc}$	i_{inv}	$= \frac{\text{E265} \cdot \text{IT267}}{e \cdot \text{E242}} + \frac{\text{E242}}{R_{inv}}$	$\widehat{\text{IT240}}$	$= i_{inv} + \text{IT281}$
i_{rms}	$= \frac{v_{rms}}{R_{fan}} + \frac{v_{rms}}{R_{ac}}$	v_{rms}	$= v_{inv,0}$	$\widehat{\text{IT267}}$	$= \frac{\text{E265}}{R_{fan}} + \frac{\text{E265}}{R_{ac}}$	$\widehat{\text{E265}}$	$= v_{inv,0} - \frac{\text{IT267}}{R_{rms}}$
$\dot{w}$	$= 0$			$\dot{\omega}$	$= \frac{1}{J_{fan}}\left(\frac{\text{E265}}{B_{fan}} - \omega\right)$	$\widehat{\text{ST516}}$	$= \omega$



(a) Offset. (b) Drift. (c) Intermittent offset.

Figure 3: Fault profiles.

detailed modeling of the system for the QED-PC algorithm. This introduced some modeling difficulties, especially concerning IT240. In nominal operation, the measured value averages around 16 ± 2 A. When faults occur, however, the value takes on a much wider range, and the IT240 PC must accurately predict values in the entire range due to faults that are decoupled from the PC. This made the system identification task more complex. System identification was also more complex for QED-PC because sensor biases had to also be considered for the inputs to the PCs.

3.2 Fault Modeling

We consider both *parametric* faults, defined as unexpected changes in system parameter values, and *discrete* faults, defined as unexpected changes in the operating mode of a component. Parametric faults include changes in the AC and DC resistances, R_{ac} and R_{dc}, and additive terms to sensor equations. Discrete faults include stuck faults of the relays and circuit breakers, inverter failure, load failure, fan overspeed and underspeed faults, and sensor stuck faults. Note that sensor stuck faults are defined as $y(t) = c$, where c is a constant, and sensor noise is absent.

Parametric faults may assume offset, drift, and intermittent offset profiles, as defined in Fig. 3 (t_f denotes the time of fault occurrence). For an offset fault, we identify the offset Δp; for a drift fault, we identify the slope m; and for an intermittent fault, we identify the mean offset $\mu_{\Delta p}$, i.e, mean$(\Delta p_1, \Delta p_2, \ldots)$, the mean faulty time μ_f, i.e, mean$(\Delta t_{f1}, \Delta t_{f2}, \ldots)$, and the mean time it is nominal μ_n, i.e, mean$(\Delta t_{n1}, \Delta t_{n2}, \ldots)$.

4 FAULT DETECTION

QED and QED-PC use the same approach for fault detection and symbol generation, and is described in (Daigle and Roychoudhury, 2010). Each sensor is assigned a fault detector. For each sensor output $y(t)$, we define the residual as $r(t) = y(t) - \hat{y}(t)$, where $\hat{y}(t)$ is the model-predicted output signal. As described in the previous section, for QED, $\hat{y}(t)$ is computed using the global model, whereas for QED-PC, it is computed using the PC for $\hat{y}(t)$. Statistically significant nonzero residual signals indicate faults.

Fault detection works by applying a Z-test to the residual values. A threshold based on the Z-test is computed, and to account for modeling error, an additional error term E is added to the threshold. When the absolute value of the mean residual value over a small window (e.g., 5 samples) is over this combined threshold, a fault is detected.

The error terms E for QED and QED-PC are listed in Table 2. Overall, the thresholds are smaller for QED than for QED-PC. For example, for E240, QED uses a detection threshold of 0.09 compared to that of 0.1 used by QED-PC. Similarly, for E281, QED uses 0.15 as the detection threshold, while QED-PC uses 0.3, and so on. Because the PC models had to capture larger operational ranges for the sensors, the overall accuracy was not as good as with the simpler global model used by QED that had to capture only a small range of values, so E tended to be larger. Also, PCs use sensors as inputs, which are noisy, and this corrupts the predictions with noise, which also leads to an increase in E. Since the detectors were tuned to avoid false alarms, the detectors for QED-PC could not be tuned to be as sensitive as for QED.

5 FAULT ISOLATION

We utilize a qualitative diagnosis methodology that isolates faults based on the transients they cause in system behavior, manifesting as deviations in residual values (Mosterman and Biswas, 1999). In both QED and

Table 2: E Values for Fault Detection.

Sensor	QED	QED-PC
E240	0.090	0.100
E242	0.080	0.100
E265	0.070	0.110
E281	0.150	0.300
ESH244A	0.000	0.000
ISH236	0.000	0.000
IT240	0.110	0.114
IT267	0.050	0.060
IT281	0.060	0.090
ST516	20.000	40.000
TE228	0.200	0.200

QED-PC, we define residuals, as described in the previous section, with respect to a particular sensor. The main difference is the model that produces the estimate $\hat{y}$ for the residual.

The transients produced by faults are abstracted using qualitative + (increase), − (decrease), and 0 (no change) values and N (zero to nonzero), Z (nonzero to zero), and X (no discrete change) values to form *fault signatures*. Fault signatures represent these measurement deviations from nominal behavior as the immediate (discontinuous) change in magnitude, the first nonzero derivative change, and discrete zero/nonzero value changes in the measurement from the estimate caused by discrete faults. These symbols are computed from the residuals using symbol generation, as described in (Daigle and Roychoudhury, 2010).

In addition to signatures, for QED we also capture the temporal order of measurement deviations, termed *relative measurement orderings* (Daigle *et al.*, 2009). Within the QED-PC algorithm, since measurement orderings may be defined only within a given submodel, we cannot define any orderings between residuals of two different PCs, because they are decoupled. The predicted fault signatures and measurement orderings can be computed manually or automatically from a system model (Mosterman and Biswas, 1999; Daigle, 2008). The predicted signatures and orderings are compared with observed signatures and orderings in order to isolate faults.

Selected fault signatures for ADAPT-Lite are shown in Table 3 for QED and Table 4 for QED-PC, where the first symbol is the immediate change in magnitude, the second is the slope, and the third is the discrete change. For example, a positive offset in E240 will cause an abrupt increase in the E240 residual with no change in slope, and no discrete change behavior (+0X). No other sensors are affected (00X) by this fault in the QED approach, but for the QED-PC algorithm, an abrupt decrease in the E242 residual with no change in slope, and no discrete change behavior (−0X) is also caused, because sensor E240 is used as input for the PC that estimates E242. A resistance offset in AC483 causes multiple deviations for QED but only one in IT267 for QED-PC, because the corresponding parameter, R_{ac}, is present only in the PC for IT267.

Table 3: Selected Fault Signatures for the QED algorithm for ADAPT-Lite

Fault	E240	E242	IT281	IT267	ST516
AC483 $\Delta p > 0$	+0X	+0X	+0X	−0X	00X
DC485 $\Delta p > 0$	+0X	+0X	−0X	00X	00X
E240 $\Delta p > 0$	+0X	00X	00X	00X	00X
E240 $m > 0$	0+X	00X	00X	00X	00X
E240 $\mu_{\Delta p} > 0$	+0X	00X	00X	00X	00X
EY244 stuck open	+0X	−0Z	−0Z	−0Z	0−X
FAN416 underspeed	+0X	+0X	00X	−0X	−0X

Table 4: Selected Fault Signatures for the QED-PC algorithm for ADAPT-Lite

Fault	E240	E242	IT281	IT267	ST516
AC483 $\Delta p > 0$	00X	00X	00X	−0X	00X
DC485 $\Delta p > 0$	00X	00X	−0X	00X	00X
E240 $\Delta p > 0$	+0X	−0X	00X	00X	00X
E240 $m > 0$	0+X	0−X	00X	00X	00X
E240 $\mu_{\Delta p} > 0$	+0X	−∗X	00X	00X	00X
EY244 stuck open	00X	−0Z	00X	00X	00X
FAN416 underspeed	00X	00X	00X	−0X	−0X

The system is actually not diagnosable based only on signatures and orderings. First, there are four pairs of faults that produce exactly the same quantitative behavior on the given measurements: failures in CB262 and INV2, failures in EY281 and DC485, failures in EY272 and AC483, and failures in EY275 and FAN416. This problem cannot be remedied, but the recovery action is the same for both faults in each pair, so the ambiguity does not ultimately matter. Second, offset and intermittent faults produce the same initial transients, therefore they can only be distinguished by their quantitative effects. The fault identification module can handle that issue. Third, for QED, a sensor fault affects only a single residual, so when a sensor fault occurs, we, in theory, have to wait infinitely long to confirm that no other residuals deviate. For QED-PC, on the other hand, sensor faults affect many residuals, but due to the decoupling introduced by PCs, some nonsensor faults, e.g., a fault in AC483 (see Table 4) affect only a single measurement, and we run into the same issue.

To resolve this third issue, we introduce heuristic isolation rules based on timing information. We expect that residuals, if they will deviate, will do so within a certain time since fault detection. If not, we assume a 00X signature for that residual, and this allows us to isolate sensor faults for QED and nonsensor faults for QED-PC in finite time. For example, for QED we expect that nonsensor faults will affect multiple residuals within 60 s of fault detection. Note that due to these diagnosability properties, in general, QED will be faster at isolating nonsensor faults and QED-PC will be faster at isolating sensor faults, because these classes of faults produce many deviations in residuals, allowing the faults to be quickly isolated.

We also introduce several other heuristic isolation

rules, most also based on timing information, in order to improve fault isolation times and generally enhance diagnosability. For example, for both QED and QED-PC, fan faults should affect ST516 within 30 s, and relay/circuit breaker faults should affect their position sensors within 2 s. Some of these rules are based on fault identification results, and will be described in Section 6.

6 FAULT IDENTIFICATION

Fault identification takes the set of faults F produced by the fault isolation module, and computes the fault parameters for each fault, producing a new fault set F_{id} that includes this information. In some cases, fault identification may change the fault mode for a fault based on identification results, so F and F_{id} do not always have an exact correspondence. Identification is initiated immediately after the initial set of fault candidates is produced after fault detection. An identification routine is run for each fault candidate, which updates at each time step. Identification terminates for a given fault candidate when the fault isolation module determines the candidate is no longer consistent.

The faults that require identification are faults in the resistances R_{ac} and R_{dc}, and sensor faults. In addition, equivalent resistance values are computed for relays EY272 and EY281. For sensor faults of the stuck profile, the stuck value is simply computed as the most recent sensor measurement, and the candidate is eliminated as soon as two consecutive measurement values are different. Each of the remaining faults is directly associated in the model with a parameter change Δp.

In each case, we have a submodel that computes the value of $\Delta p(t)$ at a given time t based on the sensor measurements $\mathbf{y}(t)$. The resistance value R_{dc} is computed using the PC for IT281, solved for R_{dc}. The resistance value R_{ac} is given by

$$R_{ac}(t) = \frac{v_{ac}(t)}{\sqrt{i_{ac}(t)^2 - \left(\frac{v_{ac}(t)}{Z_{fan}} \sin\phi\right)^2} - \frac{v_{ac}(t)}{Z_{fan}} \cos\phi},$$

where for $v_{ac}(t)$ we use $\sqrt{2}$E265, and for $i_{ac}(t)$ we use $\sqrt{2}$IT267. Here, ϕ is the phase offset introduced by the fan load, and Z_{fan} is its equivalent impedance. The nominal values of Z_{fan} and ϕ were calculated by solving the following expression at steady state using two different values R_{ac} and measured values of i_{ac} and v_{ac}:

$$|i_{ac}| = \left| \frac{v_{ac}}{Z_{fan}} (\cos\phi + j \sin\phi) + \frac{v_{ac}}{R_{ac}} \right|.$$

This equation is derived from the complex impedance expressions for the fan and R_{ac}.

For sensor faults, $\Delta p(t)$ is computed as the residual $y(t) - \hat{y}(t)$ for output y using the predictive model. In the case of QED, $\hat{y}(t)$ is computed using the global model, but for QED-PC, it is computed using only the PC that is associated with y.

The time profile of the parameter change Δp over $[t_d, t]$, where t is the current time, may be an offset, a drift, or an intermittent offset. For the offset profile, we must identify the magnitude of the offset; for the drift profile, we must identify the slope; and for the intermittent profile, we must identify the average time the parameter value is faulty, the average offset value during this time period, and the average time the parameter value is nominal.

The identification procedure keeps the history of both the predicted nominal values p and the computed faulty values p_f. We compute the fault parameters as follows:

- For offset faults, we compute the offset simply as the mean of $\Delta p = p - p_f$.
- For drift faults, we compute the slope of Δp as $(\Delta p(t_2) - \Delta p(t_1))/(t_2 - t_1)$ over three different time intervals with (t_1, t_2) as (t_d, t), $(t_d, (t + t_d)/2)$, and $((t + t_d)/2, t)$ where t is the current time, and take the median of these values as the slope, as described in (Daigle and Roychoudhury, 2010). We improve on the robustness of this computation by computing the average value over a small window of $\Delta p(t_1)$ and $\Delta p(t_2)$.
- For intermittent offset faults, we define a limit l above which $\Delta p(t)$ is considered faulty, and below which is considered nominal. The limit l is typically chosen within 1-2% of the nominal value of $y(t)$ or $p(t)$. We step through the signal $\Delta p(t)$ to determine the average nominal time μ_n, the average faulty time μ_f, and the average faulty value $\mu_{\Delta p}$. The full procedure is described in (Daigle and Roychoudhury, 2010).

Fault identification is also used to improve fault isolation, either by eliminating candidates whose identification results are inconsistent with the supposed candidate, e.g., a stuck fault candidate is not actually stuck, or by changing the fault mode of the candidate to be consistent with the identification results, e.g., if an offset fault looks like a drift fault, then the fault mode of the candidate will be changed to drift. This type of change is meant to correct errors in fault isolation caused by incorrect symbol generation. For example, a 0+ or 0− signature must be generated in order to hypothesize a drift fault as a candidate, but, for very small drift faults, the slope may not be able to be confidently calculated so instead an offset fault is generated. The identification results are used to correct that error. Another example is distinguishing between offset and intermittent faults, because they produce the same fault signatures at the time of fault occurrence.

To generally account for this type of error, for each fault we identify the parameters for each possible fault profile, and define tests that determine which fault profile is valid. For example, if μ_n or μ_f are less than 0.5, we conclude that the fault is not intermittent. For slowly developing drift faults on noisy sensors, the computed value of Δp can fluctuate above and below the intermittent threshold l, so the fault may look intermittent. So we select three points, as in the computation of drift, and check that the relative change between Δp at these three points is greater than some threshold (e.g., 15%), and that the absolute differeces get larger over these three points. If so, the fault is indeed a drift, and not intermittent.

To determine whether a drift fault is in fact a drift, we use the same test for checking whether an inter-

mittent fault is actually a drift fault, only with lower thresholds. We use lower thresholds because if fault isolation provided the drift fault candidate, then it is because a smooth change was detected, and a positive result on the slope test has high confidence. To determine whether an offset fault is in fact an offset fault, we check that both the intermittent and drift tests fail.

7 FAULT RECOVERY

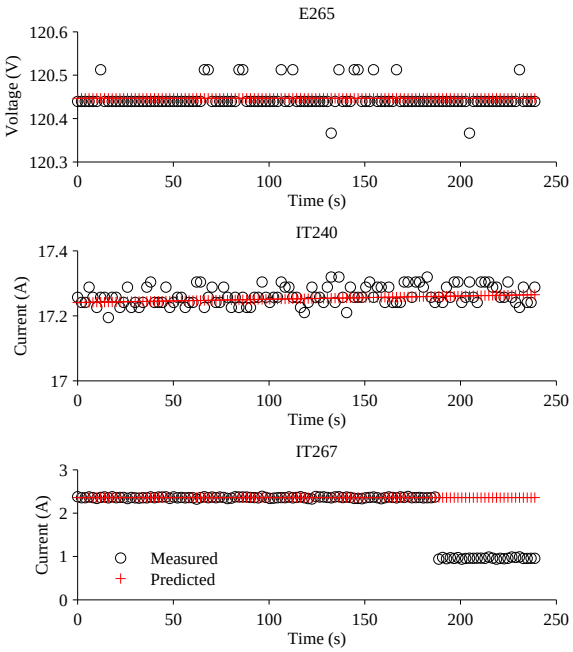
At the end of the scenario, the decision whether to abort or continue the mission must be made. The fault identification module computes a candidate set F_{id}, with each $f \in F_{id}$ being defined by the component, its fault mode, and the associated fault parameters. The oracle is viewed as a function $O(f)$ which, for a given fault, computes a recommended set of commands C. For DPI, either $C = \{abort\}$ or $C = \varnothing$.

Each command set has an associated cost. The cost is zero when the correct command is chosen. If the algorithm recommends $abort$ when the mission should be continued, the associated cost is that of the mission (25). If the algorithm recommends to continue when it should have been aborted, the associated cost is that of the mission and the vehicle (125). Therefore, we take the conservative approach and recommend $abort$ if $O(f) = \{abort\}$ for at least one $f \in F_{id}$. In the case that a fault was detected but all candidates were eliminated, then one may assume either a false positive, or a true positive with incorrect fault isolation. We assume the latter, and in this case, we again take the conservative route, and recommend an abort.

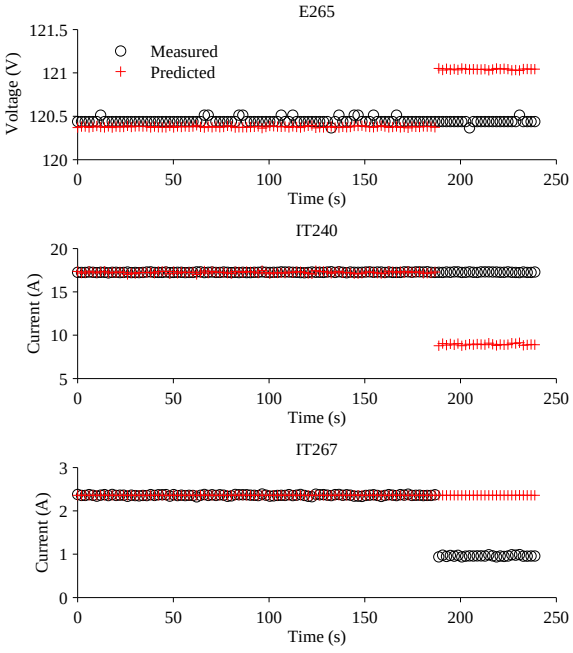
8 EXPERIMENTAL RESULTS

As an illustrative example, we consider an offset fault in IT267, with the offset being -1.4 (see Fig. 4). The fault is injected at 187 s, and QED detects the decrease in the measured value immediately. The candidate set is reduced to a failure, resistance offset, resistance drift, and intermittent resistance offset in AC483; failures in CB236, CB262, and CB266; failures in EY244, EY260, EY272, and EY275; failure or underspeed of FAN416; failure of INV2; and offset, drift, intermittent offset, and stuck faults of IT267. At the next new measured value of IT267, the IT267 stuck fault is eliminated. A fault in CB236 can also be eliminated since a corresponding change in ISH236 was not observed. At 192 s, symbol generation determines that IT267 did not undergo a zero/nonzero transition, eliminating the remaining circuit breaker faults, a fault in EY260, and the failure in INV2. At 196.7 s symbol generation determines that the slope on IT267 is 0, eliminating the resistance drift fault of AC483 and an IT267 drift fault. At 216.7 s, faults in EY275 and FAN416 are eliminated since deviations in ST516 were not observed. At 220 s, the intermittent faults are eliminated since the identified parameters are inconsistent, and faults in AC483 and EY272 are eliminated because a deviation was not observed in E265, leaving only the offset fault in IT267. The offset is computed as -1.405, and an $abort$ is correctly recommended.

QED-PC detects the fault a little later at 187.5 s with deviations in the PCs for IT267 (decrease) and IT240 (increase). A deviation is detected in IT240 because the PC computes its output based on the measured



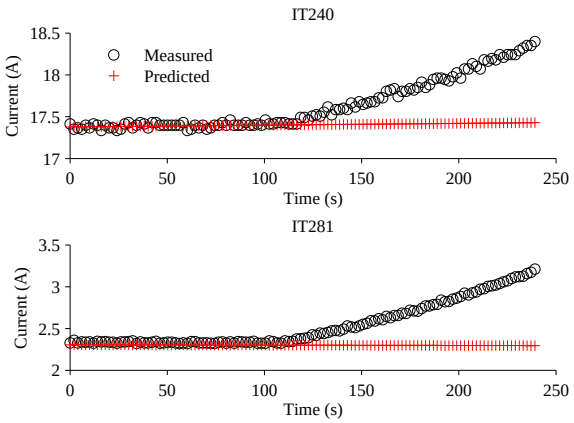
(a) QED.



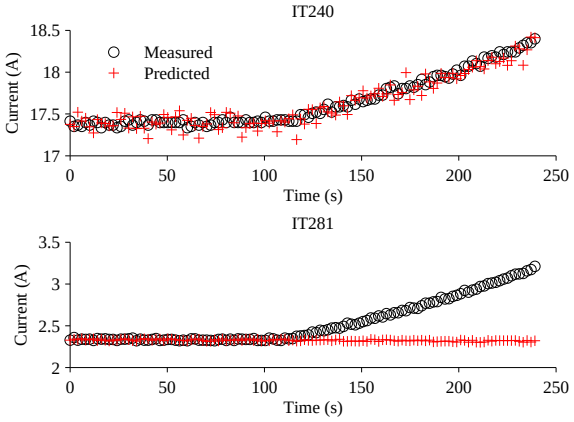
(b) QED-PC.

Figure 4: Selected measured and predicted values for IT267 offset fault with $\Delta p = -1.4$.

value of IT267, and since the sensor is faulty, the PC predicts a different value for IT240 than the measured value. The initial candidate list consists only of faults in E265, IT267, which is a much smaller candidate set than that initially generated by QED. At 188.1 s, a decrease in E265 is detected, leaving only a fault in IT267 as a candidate. At 230 s, the offset fault mode is



(a) QED.



(b) QED-PC.

Figure 5: Selected measured and predicted values for DC485 drift fault with $m = -0.022$.

confirmed with the offset computed to be -1.398, and an *abort* recommendation is sent.

As a second example, we consider a drift fault in DC485 (see Fig. 5). The fault is injected at 108.51 s, with the drift being -0.022. For QED, an increase in IT281 is detected at 122.8 s, and the initial candidate list includes faults in AC483, CB262, CB266, DC485, EY272, EY275, INV2, and IT281. At 132.1 s, an increase in IT240 is detected, eliminating all faults except those of DC485. At 132.9 s, the slope for IT281 is computed to be 0, eliminating the resistance drift fault. At 182.8 s, the intermittent resistance offset fault is eliminated because the identified parameters are not consistent with that fault mode, leaving only the resistance offset fault. However, the fault is more consistent with a drift fault, and therefore the fault mode of the candidate is changed to a drift. At 230 s the drift fault is confirmed with the drift computed to be -0.019, and an *abort* recommendation is correctly sent.

QED-PC detects the fault at 129.1 s with an increase in IT281. The initial candidate list consists of faults in DC485, E281, EY260, and IT281. At 139.1 s, the slope for IT281 is computed to be a +, eliminating all fault modes but drift. Also at this time, the fault in EY260 is eliminated since a deviation in E265 was not observed. At 189.1 s, the E281 drift fault is eliminated because a deviation in the PC for E281 was not observed. At 199.1 s, the IT281 drift fault is eliminated since no deviation was detected in any of the PCs which use it as an input, leaving only the DC485 resistance drift fault (much later than for QED). At 230 s, the drift is computed as -0.019 and an *abort* is recommended.

Table 5 summarizes the performance of QED and QED-PC in DPI using the evaluation metrics on the competition data set from DXC'10. The metrics consist of the mean time to detect faults M_{fd}, the mean false negative rate M_{fn}, the mean false positive rate M_{fp}, detection accuracy M_{da}, the mean time to isolate faults M_{fi}, the number of classification errors M_{err}, the mean CPU time M_{cpu}, the mean peak memory usage M_{mem}, and the overall recovery cost M_{rc}.

From Table 5, it is clear that both algorithms perform better than QED from DXC'10. The previous version of QED had overly sensitive fault detectors, which were fine for the training data, but resulted in many false alarms for the competition data. Further, the window used for detecting stuck faults was too small for ST516, and this resulted in some misdiagnoses that also contributed to the poor performance. A few errors in the computed signatures were also present. All these issues were corrected in the new version of QED, and the model was also improved to better match the data and further reduce false alarms.

Both QED and QED-PC were tuned such that no false alarms were detected. This, however, did result in some missed detections. QED missed a small offset fault in IT281 and a small drift fault in ST516. QED-PC missed these faults as well, in addition to a small offset fault in ST516, because its detector for ST516 had to be less sensitive due to the PC having to model the transients in fan speed. Because of the decreased sensitivity, the mean time to detect faults has increased, but this resulted in overall better isolation performance. QED-PC has a larger average detection time due to the decreased sensitivity as described in Section 4.

Of the experiments, there were 8 expected classification errors due to faults that could not be distinguished, e.g., DC485 failing and its relay failing produce the same observations. QED had the two missed detections which resulted in 10 total errors, and QED-PC had the 3 missed detections in addition to another case where a fault in E265 could not be eliminated from consideration from a fan fault. This is because the relationship of the fan speed on inverter voltage was not properly modeled, and therefore we could not drop E265 in that case. This resulted in 12 total classification errors.

Due to the excellent fault isolation results and satisfactory fault identification, the recommendation was always correct except in one case. The case where it was not correct was an intermittent resistance offset fault in DC485 in which the identified fault parameters were slightly off, causing an *abort* to be recommended when the correct action was to continue. For the missed detection cases, the faults were small enough that the correct action was to continue the mission, so the correct action was still recommended in

Table 5: QED Diagnosis Results

DA	M_{fd} (s)	M_{fn}	M_{fp}	M_{da}	M_{fi} (s)	M_{err}	M_{cpu} (ms)	M_{mem} (kb)	M_{rc}
QED (for DXC'10)	7.307	0.015	0.105	0.882	115.499	71.752	239.0	5364	2350
QED (for DXC'11)	12.632	0.015	0.0	0.987	125.996	10.0	116.192	5587	25
QED-PC	17.528	0.023	0.0	0.980	129.334	12.0	120.291	5541	25

those cases.

The CPU times and memory usages did not change considerably between algorithms or from the previous version. CPU times are improved over the previous version of QED, but this is not an entirely fair comparison since the algorithms were run on different computers.

9 CONCLUSIONS

We described our entries into DXC'11, called QED and QED-PC, which incorporate principles of qualitative event-based fault isolation. We improved over our approach from DXC'10 by improving the models, decreasing the sensitivity of fault detection, and making fault identification more robust. Further, the QED-PC algorithm utilized Possible Conflicts, and we compared the implementation and performance with the QED algorithm that uses only a global model.

Overall, diagnosis results were much improved from the previous competition entry. We found that with the PC approach, additional modeling was required and fault detection sensitivity had to be decreased. Relative measurement orderings cannot be used with a purely PC-based approach, however the decoupling introduced by the PCs enhances diagnosability in its own way. Overall, isolation times were fairly well-matched, because these were computed based on the time of the last change to the candidate set, and the final decision on what the correct fault mode is (e.g., intermittent vs. drift) was often made very late when a large amount of data was available to declare the correct fault mode with high confidence. Generally, QED-PC was faster at isolating sensor faults, whereas QED was faster at isolating non-sensor faults, for the reasons explained in Section 5. Therefore, a new approach that uses residuals from both the global model and PCs would increase robustness, improve overall diagnosability, and enable many of the heuristic fault isolation rules to be eliminated.

In the future, we would like to apply these algorithms to Diagnostic Problem II (DPII), which increases the complexity of DPI in several ways. First, it encompasses the entire ADAPT system. Generally, this is not a problem for QED or QED-PC, it simply requires modeling the additional components, and both algorithms scale well with model size. Second, DPII includes mode changes as part of nominal behavior. Extensions for QED to hybrid systems are described in (Daigle, 2008), and recent work applying PCs to hybrid systems is described in (Bregon *et al.*, 2011). Third, DPII includes multiple faults. Extensions to QED for multiple faults are also described in (Daigle, 2008), although fault identification is not addressed. Handling multiple faults with QED-PC is part of ongoing work. Performing multiple fault diagnosis within hybrid systems adds another level of complexity that QED and QED-PC are not yet equipped to handle, but DPII presents a good case study for further developing such extensions.

ACKNOWLEDGMENTS

Anibal Bregon's work has been funded by the Spanish MCI DPI2008-01996 and TIN2009-11326 grants.

REFERENCES

(Bregon *et al.*, 2011) A. Bregon, C. Alonso, G. Biswas, B. Pulido, and N. Moya. Hybrid systems fault diagnosis with possible conflicts. In *Proceedings of the 22nd International Workshop on Principles of Diagnosis*, October 2011.

(Bregon *et al.*, to appear) A. Bregon, G. Biswas, and B. Pulido. A decomposition method for nonlinear parameter estimation in TRANSCEND. *IEEE Trans. on Systems, Man, and Cybernetics, Part A*, to appear.

(Daigle and Roychoudhury, 2010) M. Daigle and I. Roychoudhury. Qualitative event-based diagnosis: Case study on the second international diagnostic competition. In *Proceedings of the 21st International Workshop on Principles of Diagnosis*, pages 371–378, October 2010.

(Daigle *et al.*, 2009) M. J. Daigle, X. Koutsoukos, and G. Biswas. A qualitative event-based approach to continuous systems diagnosis. *IEEE Trans. on Control Systems Technology*, 17(4):780–793, July 2009.

(Daigle, 2008) M. Daigle. *A Qualitative Event-based Approach to Fault Diagnosis of Hybrid Systems*. PhD thesis, Vanderbilt University, 2008.

(Mosterman and Biswas, 1999) P. J. Mosterman and G. Biswas. Diagnosis of continuous valued systems in transient operating regions. *IEEE Trans. on Systems, Man and Cybernetics, Part A*, 29(6):554–565, 1999.

(Poll *et al*., 2007) S. Poll *et al*. Evaluation, selection, and application of model-based diagnosis tools and approaches. In *AIAA Infotech@Aerospace 2007 Conference and Exhibit*, May 2007.

(Pulido and Alonso-González, 2004) B. Pulido and C. Alonso-González. Possible Conflicts: a compilation technique for consistency-based diagnosis. *IEEE Trans. on Systems, Man, and Cybernetics, Part B*, 34(5):2192–2206, 2004.

(Roychoudhury, 2009) I. Roychoudhury. *Distributed Diagnosis of Continuous Systems: Global Diagnosis Through Local Analysis*. PhD thesis, Vanderbilt University, August 2009.