Qualitative Event-based Diagnosis: Case Study on the Second International Diagnostic Competition

Matthew Daigle [1] and Indranil Roychoudhury [2]

[1] *University of California, Santa Cruz, NASA Ames Research Center, Moffett Field, CA, 94035, USA*
matthew.j.daigle@nasa.gov
[2] *SGT Inc., NASA Ames Research Center, Moffett Field, CA, 94035, USA*
indranil.roychoudhury@nasa.gov

ABSTRACT

We describe a diagnosis algorithm entered into the Second 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 and provide corresponding recovery recommendations. The diagnosis algorithm embodies a model-based approach, centered around qualitative event-based fault isolation. Faults produce deviations in measured values from model-predicted values. The sequence of these deviations is matched to those predicted by the model in order to isolate faults. We augment this approach with model-based fault identification, which determines fault parameters and helps to further isolate faults. We describe the diagnosis approach, provide diagnosis results from running the algorithm on provided example scenarios, and discuss the issues faced, and lessons learned, from implementing the approach.

1 INTRODUCTION

Timely and robust detection, isolation, and identification of faults in engineering systems lies at the core of systems health management technologies. This paper presents a model-based, qualitative, event-based fault diagnosis scheme that was entered into the Second International Diagnostic Competition (DXC'10) (Poll *et al.*, 2010). 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. ADAPT is an electrical power distribution system, representative of those found in spacecraft. Our entry into DXC'10 is called QED, for qualitative event-based diagnosis.

QED extends the TRANSCEND diagnosis scheme described in (Mosterman and Biswas, 1999; Daigle *et al.*, 2010). 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 incorporate additional diagnostic information, known as 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.*, 2007; 2009). DPI requires fault identification, and includes abrupt, incipient, and intermittent fault profiles. TRANSCEND deals only with abrupt profiles, so we incorporate extensions for incipient faults (Roychoudhury, 2009), and new work for identification of intermittent faults.

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

2 DIAGNOSIS APPROACH

We focus on Diagnostic Problem I (DPI) of the industrial track of DXC'10. The problem here is to decide whether the mission should be aborted or continued. In order to make this decision, we must determine if the system is faulty, and if the fault warrants an abort recommendation or not, given the system observations.

The diagnosis architecture is shown in Fig. 1. The system receives inputs $\mathbf{u}(t)$ and produces outputs $\mathbf{y}(t)$. Due to the simplicity of the monitored system, we use a predictive model instead of an observer. Our system model runs simultaneously, producing predicted outputs $\hat{\mathbf{y}}(t)$, given the inputs $\mathbf{u}(t)$. Using statistical methods, the fault detection module decides when a measurement has deviated from its nominal value, triggering fault isolation. 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.

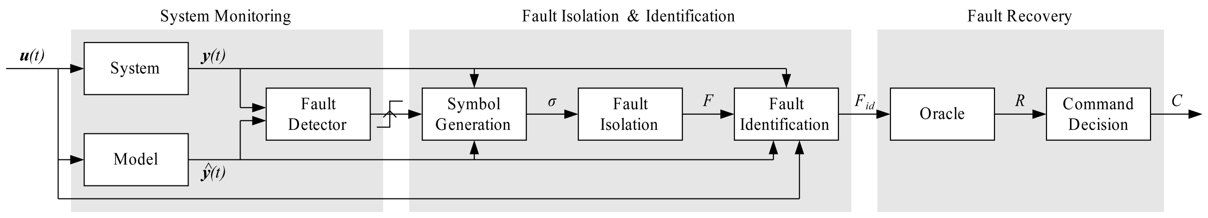


Figure 1: Diagnosis and recovery architecture.

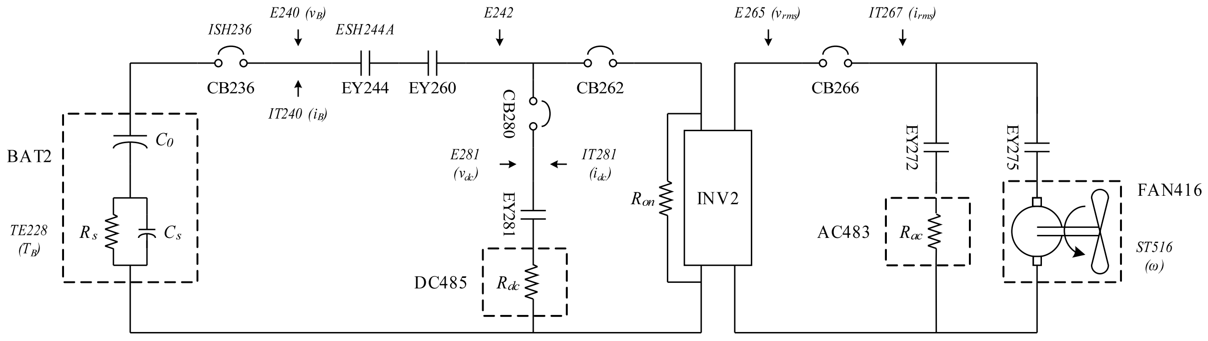


Figure 2: ADAPT-Lite schematic.

Each fault $f \in F$ is associated with a component, its fault mode, and its fault parameters. Fault identification computes, for each fault $f \in F$, the values of the fault parameters. An oracle is provided by DXC'10 that decides 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. It is used for prediction of nominal values and within the fault detection, isolation, and identification modules. In the following, we describe the models of nominal and faulty behavior of the ADAPT-Lite system.

3.1 Nominal Model

The schematic of ADAPT-Lite is given in Fig. 2. Sensors are denoted in italics. 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. *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. We describe models for each of the components in turn.

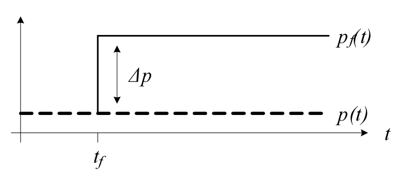
The battery consists of two 12 V lead-acid batteries in series. We lump these together into a single battery model. Battery models typically must include a set of complex nonlinear behaviors (Ceraolo, 2000; Daigle *et al.*, 2009). However, most of these characteristics are not evident within the short, four-minute time frame of the experimental scenarios. Therefore, we utilize a simplified electrical circuit equivalent model, consisting of a single large capacitance, C_0, in series with a capacitor-resistor pair, C_s and R_s, that subtracts from the voltage provided by C_0 (see Fig. 2). In reality, R_s is a function of state of charge, depth of charge, and temperature, but, for our purposes, we may assume it to be constant. C_s is much smaller than C_0. Since the battery voltage decreases faster at lower voltages, we express C_0 as a function that decreases with voltage. The battery also has a large parasitic resistance in parallel that accounts for the self-discharge of the battery due to various parasitic processes, which may be omitted here. The battery may then be described as

$$\begin{aligned}\dot{v}_0 &= \frac{1}{C_0} i_B \\ \dot{v}_s &= \frac{1}{C_s}\left(i_B - \frac{v_s}{R_s}\right) \\ v_B &= v_0 - v_s,\end{aligned}$$

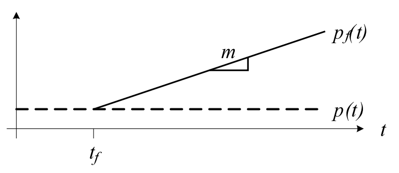
where v_B is the battery voltage, i_B is the battery current, v_0 is the voltage across C_0, and v_s is the voltage drop across the C_s-R_s pair. The battery temperature is unchanging over the scenario length so we express it as a constant T_B. Deviation from T_B implies a fault.

The inverter transforms DC power to AC power. When operating nominally, the voltage v_{rms} is controlled very close to 120 V AC. From a power balance of the AC and DC sides of the inverter, we have $v_{inv} i_{inv} = e \cdot v_{rms} i_{rms}$, where e is the inverter efficiency, v_{inv} is the input DC voltage to the inverter, and i_{inv} is the input DC current to the inverter. The inverter still draws a small amount of current even when $i_{rms} = 0$, and this is captured as a DC resistance parallel to the inverter, R_{on}. We have

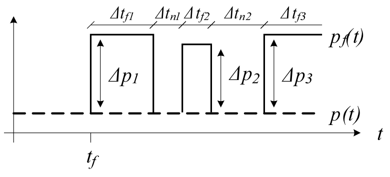
$$i_{inv} = \frac{v_{rms} i_{rms}}{e \cdot v_{inv}} + \frac{v_{inv}}{R_{on}}.$$



(a) Offset.



(b) Drift.



(c) Intermittent offset.

Figure 3: Fault profiles.

The DC and AC resistive loads are modeled as pure resistances, with R_{dc} and R_{ac}, respectively. The fan has both resistive and inductive properties, so introduces a phase difference in its current from the input voltage. We express its equivalent impedance as Z_{fan} and phase offset as ϕ. The inverter current i_{rms} is the vector sum of the AC load and fan currents. Nominally, the fan is always on, so we can express its speed as a constant ω. Any deviation from ω implies a fault.

Relays and circuit breakers are modeled as ideal switches. For DPI, there are no mode changes during nominal operation, so observed mode changes are directly attributed to faults.

Using the available data sets, we have identified the parameters of our model and obtained very accurate descriptions of nominal behavior.

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. These parameters may assume offset, drift, and intermittent offset profiles, as shown in Fig. 3 (t_f denotes the time of fault occurrence).

For an offset, the faulty value $p_f(t)$ is defined by

$$p_f(t) = p(t) + \Delta p,$$

where $p(t)$ is the nominal value, and Δp is the offset. A drift is defined by its slope m, i.e,

$$p_f(t) = p(t) + m(t - t_f).$$

For intermittent offsets, the offset alternates between zero and a nonzero value. The profile is defined by three parameters, 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$(\Delta t_{n1}, \Delta t_{n2}, \ldots)$.

Discrete faults include stuck faults of the relays and circuit breakers, inverter failure, load failure, fan overspeed and underspeed faults, the introduction of a battery parasitic resistance R_p, 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.

4 FAULT DETECTION AND SYMBOL GENERATION

Each sensor is assigned a fault detector and a symbol generator. 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. Statistically significant nonzero residual signals indicate faults. Following fault detection for a sensor, its symbol generator is initiated to calculate magnitude, slope, and discrete change symbols that are used for fault isolation.

4.1 Fault Detection

We use the Z-test for robust fault detection using a set of sliding windows (Daigle *et al.*, 2010). A small window, W_2, is used to estimate the current mean $\mu_r(t)$ of a residual signal:

$$\mu_r(t) = \frac{1}{W_2} \sum_{i=t-W_2+1}^{t} r(i).$$

The variance of the nominal residual signal, $\sigma_r^2(t)$, is computed using a large window W_1 preceding W_2, by a buffer W_{delay}, which ensures that W_1 does not contain any samples after fault occurrence. The variance is computed using:

$$\sigma_r^2(t) = \frac{1}{W_1} \sum_{i=t-W_2-W_{delay}-W_1+1}^{t-W_2-W_{delay}} (r(i) - \mu_r'(t))^2,$$

where

$$\mu_r'(t) = \frac{1}{W_1} \sum_{i=t-W_2-W_{delay}-W_1+1}^{t-W_2-W_{delay}} r(i).$$

A pre-specified confidence level determines the bounds $z^- < 0$ and $z^+ > 0$ for a two-sided Z-test. The fault detection thresholds, $\varepsilon_r^-(t)$ and $\varepsilon_r^+(t)$, are dynamically computed using:

$$\varepsilon_r^-(t) = z^- \frac{\sigma_r(t)}{\sqrt{W_2}} - E$$

$$\varepsilon_r^+(t) = z^+ \frac{\sigma_r(t)}{\sqrt{W_2}} + E,$$

where E is a modeling error term. A fault is detected if $\mu_r(t)$ lies outside of the thresholds at time t. The parameters W_1, W_2, W_{delay}, the z bounds, and E must be tuned to optimize performance.

Note that for stuck faults in sensors (recall that these faults eliminate noise from the signal $y(t)$), a sensor that is stuck within nominal ranges will not be detected by the above method. Hence, an additional detection test is required for these faults. For sensor y,

$$stuck_y(t) = \begin{cases} true, & \sum_i^{N_s} |y(t) - y(t-i)| = 0 \\ false, & \text{otherwise} \end{cases},$$

where N_s is a pre-defined limit. If the past N_s consecutive samples of $y(t)$ are all the same, then $stuck(t) = true$. The value of N_s depends on the particular sensor. For some sensors in ADAPT-Lite, N_s must be quite large (e.g., $N_s = 400$), because the sensors normally repeat the same value for long periods of time. For the discrete sensors ISH236 and ESH244A, we effectively set $N_s = \infty$, because these sensors are binary-valued and noiseless.

4.2 Symbol Generation

Robust methods based on the Z-test are also used for symbol generation (Daigle *et al.*, 2010). The first symbol is derived directly from the result of fault detection. If the measurement residual, $r(t)$, is greater than $\varepsilon_r^+(t)$ (or less than $\varepsilon_r^-(t)$), we obtain a + (or −).

The second symbol calculated is for the direction of the slope of the residual. We start with an estimate of the initial residual value, $\mu_{r_0}(t_d)$, at the time of fault detection, t_d, over a small window W_3:

$$\mu_{r_0}(t_d) = \frac{1}{W_3} \sum_{i=t_d}^{t_d+W_3-1} r(i).$$

The mean of the residual slope is computed over a window from t_d to t:

$$\mu_{r_d}(t) = \frac{1}{t - t_d + 1} \left(\sum_{i=t_d}^{t} r(i) - \mu_{r_0} \right)$$

Using bounds z^- and z^+, the thresholds are:

$$\varepsilon_{r_d}^-(t) = z^- \sigma_r \left(\frac{1}{\sqrt{W_3}} + \frac{1}{\sqrt{W_n}} \right) - E_s$$
$$\varepsilon_{r_d}^+(t) = z^+ \sigma_r \left(\frac{1}{\sqrt{W_3}} + \frac{1}{\sqrt{W_n}} \right) + E_s.$$

The − (+) symbol is generated when $\mu_{r_d} < \varepsilon_{r_d}^-(t)$ ($\mu_{r_d} > \varepsilon_{r_d}^+(t)$). The window used to calculate the slope is increased until the symbol is successfully generated, or $t - t_d$ becomes larger than a pre-specified limit, at which the slope is reported as 0, implying that the true slope is either zero, or unknown but very small. If the first and second symbols do not match, we interpret this as a discontinuity in the signal, otherwise, a smooth change is assumed.

We compute also a discrete change symbol, which is used to decide whether a signal has switched between a nonzero and zero value, which is useful for distinguishing between parametric and discrete faults (Daigle *et al.*, 2010). To compute the discrete change symbol, we do not use the residual, but use the observed and estimated values of the signal. We compute the mean of the measured signal, $y(t)$, and the mean of the estimate, $\hat{y}(t)$, over a small window, W_c:

$$\mu_y(t_d) = \frac{1}{W_c} \sum_{i=t_d}^{t_d+W_c-1} y(i)$$

$$\mu_{\hat{y}}(t_d) = \frac{1}{W_c} \sum_{i=t_d}^{t_d+W_c-1} \hat{y}(i).$$

We wish to determine whether each signal belongs to a population with zero mean, and choose the variance of the population to be the variance of the residual as a good approximation of the true variance of the zero-mean distribution. The thresholds are computed as:

$$\varepsilon_{y_d}^- = \varepsilon_{\hat{y}_d}^- = z^- \frac{\sigma_r(t_d)}{\sqrt{W_c}} - E_c$$
$$\varepsilon_{y_d}^+ = \varepsilon_{\hat{y}_d}^+ = z^+ \frac{\sigma_r(t_d)}{\sqrt{W_c}} + E_c,$$

where E_c is a modeling error term. If $\mu_y(t_d)$ is outside its bounds, we say it is nonzero, otherwise we say it is zero. Similarly, if $\mu_{\hat{y}}(t_d)$ is outside its bounds, we say it is nonzero, otherwise we say it is zero. If the estimate is nonzero and the measurement is zero, we report Z, and if the estimate is zero and the measurement is nonzero, we report N, else, we report X.

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 observed measurement values from nominal measurement values (Mosterman and Biswas, 1999). The transients 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 mode changes.

In addition to fault signatures, we also capture the temporal order of measurement deviations, termed *relative measurement orderings* (Daigle *et al.*, 2007), based on the intuition that fault effects will manifest in some parts of the system before others. Measurement orderings are based on analysis of the transfer functions from faults to measurements (Daigle *et al.*, 2007). The combination of fault signatures and measurement orderings forms qualitative event-based information for fault isolation (Daigle *et al.*, 2009).

Measurement orderings do not allow one to eliminate a fault hypothesis based on the lack of observing a measurement deviation. However, for ADAPT-Lite, there are a substantial number of cases where this is desirable. When a sensor fault occurs, it will cause a deviation in a single measurement only, so, if only one measurement deviation has been observed for a significant amount of time after fault detection, we may assume that it is a sensor fault and eliminate all candidates that are inconsistent with that assumption. For specific measurements, we also expect deviations to occur within a certain amount of time. For example, faults that affect the fan speed should cause deviations in *ST516* within 30 s of t_d. If *ST516* has not deviated by then, all such faults may be eliminated.

The fault signatures and measurement orderings can be computed manually or automatically from a system model. The temporal causal graph (TCG) representation, derived from the system model, can be used with a forward propagation algorithm to predict qualitative

Table 1: Selected Fault Signatures for ADAPT-Lite

Fault	*E240*	*E265*	*IT240*	*IT267*	*ST516*
AC483 $\Delta p > 0$	`0+X`	`+0X`	`-0X`	`-0X`	`00X`
DC485 $\Delta p > 0$	`0+X`	`00X`	`-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`
EY260 stuck open	`+0X`	`-0Z`	`-0Z`	`-0Z`	`0-X`
FAN416 underspeed	`0+X`	`+0X`	`-0X`	`-0X`	`0-X`

effects of faults on measurements and their possible sequences of deviations (Mosterman and Biswas, 1999; Daigle, 2008). A TCG captures system variables as nodes in a graph, and the mathematical relations between them as edges. Fault parameters appear on edges, allowing the propagation of parameter changes to be propagated over the system variables. Since there are no mode changes in the considered system, generation of signatures and orderings is performed offline, and provided as input to the diagnosis algorithm.

Selected fault signatures for ADAPT-Lite are shown in Table 1, 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`). An intermittent offset may also cause this initial transient, therefore, fault identification is necessary to distinguish these faults. The underspeed fault of the fan will cause a smooth increase in battery voltage (`0+X`), an abrupt increase in inverter voltage (`+0X`), abrupt decreases in battery and inverter currents (`-0X`), and a smooth decrease in fan speed (`0-X`). Many measurement orderings may be derived for a number of faults also. Because of the capacitive effect of the battery, faults cause changes in currents before changes in voltages, except for discrete failures which cause voltages to go directly to zero. For fan faults, the inverter current is affected before the change in fan speed. If we ever see the fan speed deviate first, then this allows us to immediately conclude that it is a sensor fault in *ST516*.

6 FAULT IDENTIFICATION

Fault identification is initiated immediately after the initial set of fault candidates is produced after fault detection. Each candidate has its own identification routine that updates its estimate at every time step. When the identification result is inconsistent with the fault mode, the fault candidate is eliminated; in this way identification helps in the isolation step.

Our fault identification procedure is related to (Roychoudhury *et al.*, 2008; Bregon *et al.*, 2009) in that it uses submodels for fault identification. However, we use simpler methods for estimating the fault parameters in our approach. For faults dealing with R_{ac}, R_{dc}, and R_p, we directly calculate the parameter value $\hat{p}(t)$ at each time step t as a function of sensor values at t, and (except for R_p, in which the goal is to calculate R_p directly), compute the offset at t using $\Delta p(t) = \hat{p}(t) - p(t)$, where $p(t)$ is the nominal value. For sensor faults, we compute the current offset at each time step using $\Delta p(t) = y(t) - \hat{y}(t)$, where $\hat{y}(t)$ is the model-predicted output at time t. In each case, we then analyze the history of Δp over $[t_d, t]$ to determine the offset, drift, or intermittent offset parameters at t.

The resistance value R_{dc} is given by

$$R_{dc}(t) = \frac{v_B(t)}{i_{dc}(t)},$$

where for $v_B(t)$ we use *E281*, and for $i_{dc}(t)$ we use *IT281*. 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*. Recall that 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}.

To identify the parasitic load R_p, we make two simplifying assumptions. First, since C_s is relatively small, the battery voltage reaches a new steady-state value soon after the fault is injected, and we may omit C_s from the model. Second, since C_0 is very large, the voltage v_0 will not change substantially over the duration of the scenario, and we may assume it is constant during that period. Given this, the equivalent circuit simplifies to that shown in Fig. 4. The value of the parasitic load may then be calculated directly as

$$R_p(t) = \frac{v_B(t)}{\frac{1}{R_s}(v_0 - v_B(t)) - i_B(t)}, t \geq t_d.$$

Here, $v_B(t)$ is provided by *E240*, and $i_B(t)$ is provided by the sensor *IT240*. The voltage v_0 is calculated over a small portion of data at the beginning of the scenario (where R_p is guaranteed to be absent) as

$$v_0 = v_B + i_B R_s,$$

i.e., when R_p is not attached, $i_0 = i_B$, and *IT240* may be used as i_B. We use the value of R_s estimated during model identification.

Given a history of Δp values over $[t_d, t]$, we compute the fault parameters for the given fault mode. For offset faults, we simply take the mean of $\Delta p(t)$, and this provides the offset. For drift faults, we compute the slope over three different intervals, as shown in

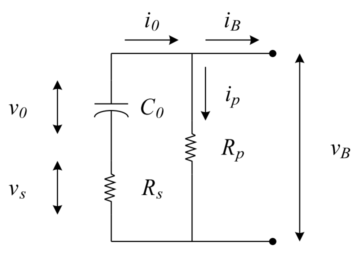


Figure 4: Battery equivalent circuit.

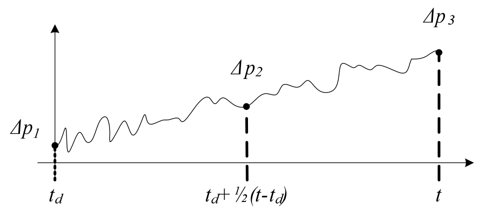


Figure 5: Identification of drift faults.

Fig. 5. We calculate

$$m_1(t) = \frac{\Delta p_2 - \Delta p_1}{\frac{1}{2}(t - t_d)}$$
$$m_2(t) = \frac{\Delta p_3 - \Delta p_2}{\frac{1}{2}(t - t_d)}$$
$$m_3(t) = \frac{\Delta p_3 - \Delta p_1}{t - t_d}$$
$$m(t) = \text{median}(m_1(t), m_2(t), m_3(t)).$$

For a large $t - t_d$, the effects of noise are diminished and accurate estimates may be achieved. Taking the median further decreases the sensitivity to noise. Other techniques may be used, such as taking the mean of a few samples around t_d and around t, and computing slope based on those, however, the method we adopt here has proven effective for the selected case study.

For intermittent offset faults, we utilize a limit l above which $\Delta p(t)$ is considered faulty, and below which is considered nominal. The limit l is typically chosen as something within 1-2% of the nominal value of $y(t)$ or $p(t)$. We step through the signal $\Delta p(t)$, and maintian two counters k_n and k_f. Each time we transition from a nominal value to a faulty value, we increment k_f, and when we transition from a faulty value to a nominal value, we increment k_n. In effect, these two counters keep track of the number of times the signal was faulty and nominal. For each new nominal value, we increment a second counter τ_n that keeps track of the total amount of time the signal is nominal. Similarly, for each new faulty value, we increment a counter τ_f that keeps track of the total amount of time it is faulty. Then, the fault parameters are

$$\mu_{\Delta p} = \text{mean}(v_f),\ \mu_f = \text{mean}\left(\frac{\tau_f}{k_f}\right)\ \mu_n = \text{mean}\left(\frac{\tau_n}{k_n}\right)$$

Fault identification is also used to help further isolate faults. For example, AC483 failing, EY272 becoming stuck open, and an increase in the AC483 resistance all have the same qualitative fault signatures and measurement orderings, so cannot be distinguished based on that information alone. But, for each of these, we can calculate an equivalent resistance offset. If the true fault is a failure in AC483 or EY272, then the R_{ac} equation yields a large negative value, but if a resistance offset is the true fault, then the R_{ac} equation will yield a reasonable value, allowing us to differentiate the faults.

Identification is also used to help differentiate between different fault profiles. Both persistent and intermittent offsets give the same signatures and orderings, but if the fault is truly persistent, then the fault parameters for the intermittent fault mode will have very small values for μ_n. If μ_n is less than 0.5 s by 60 s past t_d, we can eliminate the intermittent fault mode as a candidate. Identification may also help correct isolation mistakes. If a drift is small enough, then the corresponding slope symbol may be calculated as 0, wrongly identifying offset as the fault mode. But, we can compute the offset at t_d and at t, and if they are significantly different from each other (e.g., a 25% difference), then the fault mode is actually a drift.

7 FAULT RECOVERY

Towards the scenario end, a decision must be made as to whether the mission should be aborted or continued. The fault identification module computes a candidate set F, with each $f \in F$ 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 by the decision module. If the algorithm recommends $abort$ when the mission should be continued, the associated cost is that of the mission. If the algorithm recommends to continue when it should have been aborted, the associated cost is that of the mission and the vehicle. Therefore, we take the conservative approach where:

$$C = \begin{cases} \{abort\}, & \{abort\} \in \{O(f) : f \in F\} \\ \varnothing, & \text{otherwise,} \end{cases}$$

i.e., if an abort is recommended for at least one $f \in F$, we recommend abort. This is satisfactory because we have a high confidence in our diagnosis algorithm.

8 EXPERIMENTAL RESULTS

The results from running QED on the provided fault scenarios are as shown in Tables 2 and 3. The nominal scenarios are omitted, as no false positives were detected. The time of fault occurrence is denoted by t_f, of detection by t_d, and of isolation by t_i. All times are in seconds. The correct fault was always isolated, and the fault parameters, in most cases, are close to the actual values. Unique diagnoses were not obtained in four cases, where the faults are not actually distinguishable: (*i*) AC483 failing and its relay EY272 getting stuck open, (*ii*) the fan failing and its relay EY275 failing, (*iii*) DC485 failing and its relay EY284 becoming stuck open, and (*iv*) the inverter failing and the preceding circuit breaker CB262 failing. In each of these

Table 2: Mean Detection and Isolation Times

Fault Class	Size of Class	$t_d - t_f$	$t_i - t_f$
All Faults	34	6.21	42.91
Physical Faults	22	5.30	36.64
Sensor Faults	12	7.90	54.42
Abrupt Faults	27	2.95	29.33
Incipient Faults	7	18.81	95.30
Persistent Faults	25	8.38	50.24
Intermittent Faults	9	0.21	22.57

cases, the lack of a relay or circuit breaker sensor results in the ambiguity. In these cases, the recommendation is always the same, so the correct recommendation was made. For all other cases, the candidate was uniquely isolated and the fault parameters were identified with enough precision to also obtain the correct recommendation.

The mean detection and isolation times (in seconds) are shown in Table 2. Here, we divide the faults into different classes. On average, detection took under 10 s, and isolation took under 60 s. Physical faults could be detected and isolated faster than sensor faults, and abrupt faults could be detected and isolated significantly faster than incipient faults. Because intermittent faults were always abrupt, they could be detected and isolated faster than the persistent faults, which included drift faults.

As an illustrative example, we consider a resistance offset in AC483 that occurs at 180.4 s with $\Delta p = 15$, shown in Fig. 6. At 180.4 s, a decrease in *IT240* is detected. The initial candidate list contains resistance increases in AC483, AC483 failing, resistance increases in DC485, DC485 failing, each of the circuit breakers failing, each of the relays failing, the fan failing or in the underspeed mode, the inverter failing, or faults in *IT240*. At 180.4 s, a decrease in *IT267* is detected, eliminating faults in *IT240* and DC485. At 183.4 s, QED computes that *IT240* did not go to zero, so the fault in CB236 is eliminated. At 183.5 s, QED computes that *IT267* has not gone to zero, eliminating the remaining circuit breaker faults and the inverter failure as candidates. At 191.3 s, QED computes the slope of *IT240* as 0, eliminating the resistance drift fault. At 210.3 s faults in the fan and its relay are eliminated because deviations in *ST516* have not been observed. At 220 s, AC483 failed and EY272 stuck open are eliminated because the equivalent resistance offset for these faults does not agree with the calculated resistance offset. Also, the intermittent resistance offset is eliminated because μ_n was calculated as 0 s. This leaves a resistance offset of AC483 as the remaining candidate, and the offset is calculated as 13.9, which differs from the true value by 7.3%. The corresponding recommendation is to continue the mission.

9 CONCLUSIONS

We described our entry into DXC'10, called QED, which incorporates principles of qualitative event-based fault isolation. We extended our approach with fault identification and several heuristics to further improve fault isolation and identification, based on

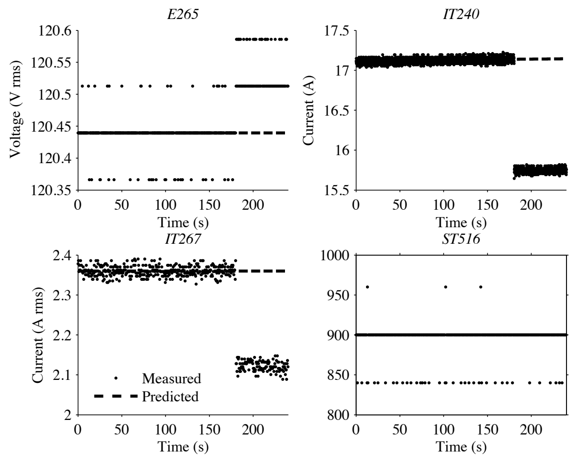


Figure 6: Selected measurements for AC483 offset with $\Delta p = 15$.

knowledge of the system. We found it crucial to also utilize the results of fault identification to help resolve further ambiguities in fault isolation. The performance of the algorithm hinges on correct symbol generation, but it can be difficult to tune the slope calculation because it is also used for discontinuity detection. We believe that a separate reliable method of discontinuity detection is necessary to alleviate this problem.

Although successful on the provided diagnosis scenarios, there are potential problems that could arise when applied to the competition data set. Our fault detectors and symbol generators were tuned for optimal performance with the provided scenarios, but may be too sensitive for some of the competition data, resulting in false alarms or incorrect symbol generation, which may result in incorrect diagnoses, and, consequently, incorrect recovery recommendations. In many places, we make final fault isolation decisions based on manually selected quantitative thresholds, so incorrect diagnoses cannot be later corrected using new evidence (e.g., finding out a stuck sensor is not really stuck). This issue may be overcome using nonmonotonic or probabilistic reasoning. The approach is limited to single faults, so the capability to handle multiple faults, of which initial progress has been described in (Daigle, 2008), would be needed to apply the approach to the full ADAPT system. In order to help manage the complexity of the much larger system, distributed diagnosis approaches such as those explored in (Roychoudhury *et al.*, 2009; Roychoudhury, 2009; Daigle *et al.*, 2010) may be useful as well.

REFERENCES

(Bregon *et al.*, 2009) A. Bregon, B. Pulido, and G. Biswas. Efficient on-line fault isolation and identification in Transcend for nonlinear systems. In *Proceedings of the 20th International Workshop on Principles of Diagnosis*, pages 291–298, June 2009.

(Ceraolo, 2000) Massimo Ceraolo. New dynamical models of lead-acid batteries. *IEEE Trans. on Power Systems*, 15(4):1184–1190, November 2000.

Table 3: Diagnosis Results

True Candidate	t_f	t_d	t_i	F	C
AC483 $\Delta p = -21$	90.2	90.2	150.2	AC483 $\Delta p = -21.82$	$\{abort\}$
AC483 $\Delta p = 15$	180.4	180.4	220	AC483 $\Delta p = 13.91$	$\{abort\}$
AC483 $m = -0.1$	32	41.3	101.3	AC483 $m = -0.09$	$\{abort\}$
AC483 $m = 0.071$	30	42.1	102.1	AC483 $m = 0.069$	$\{abort\}$
AC483 $\mu_{\Delta p} = -21, \mu_f = 3.6, \mu_n = 19.6$	29.9	30	33.2	AC483 $\mu_{\Delta p} = -20.4, \mu_f = 3.61, \mu_n = 17.16$	$\varnothing$
AC483 $\mu_{\Delta p} = -148, \mu_f = 3.8, \mu_n = 5.7$	30.5	30.6	33.6	AC483 $\mu_{\Delta p} = -148.7, \mu_f = 3.73, \mu_n = 5.12$	$\{abort\}$
AC483 failed	50.1	50.1	110.1	AC483 failed, EY272 stuck open	$\{abort\}$
BAT2 $R_p = 6$	120.8	157.4	160.7	BAT2 $R_p = 6.0$	$\varnothing$
CB266 failed	61.5	61.5	64.7	CB266 failed	$\{abort\}$
DC485 $\Delta p = -2.5$	59.5	59.7	119.7	DC485 $\Delta p = -2.56$	$\{abort\}$
DC485 $\Delta p = 4.5$	150.5	150.5	210.5	DC485 $\Delta p = 3.85$	$\{abort\}$
DC485 $m = -0.005$	35	77.6	137.6	DC485 $m = -0.0044$	$\{abort\}$
DC485 $m = 0.021$	30	44.6	220	DC485 $m = 0.021$	$\{abort\}$
DC485 $\mu_{\Delta p} = -3, \mu_f = 3.9, \mu_n = 7.7$	30.5	30.7	34.2	DC485 $\mu_{\Delta p} = -2.94, \mu_f = 3.73, \mu_n = 7.42$	$\{abort\}$
DC485 $\mu_{\Delta p} = -2.8, \mu_f = 4.1, \mu_n = 6.1$	30.6	30.8	35.1	DC485 $\mu_{\Delta p} = -2.96, \mu_f = 4.17, \mu_n = 5.53$	$\{abort\}$
DC485 $\mu_{\Delta p} = -3.2, \mu_f = 3.9, \mu_n = 13.5$	30.5	30.8	34.3	DC485 $\mu_{\Delta p} = -3.07, \mu_f = 3.82, \mu_n = 12.1$	$\varnothing$
DC485 failed	70.8	70.8	75.2	DC485 failed, EY284 stuck open	$\{abort\}$
E240 $\Delta p = -1$	110	110	170	*E240* $\Delta p = -0.99$	$\varnothing$
E242 $m = 0.005$	75	108.3	168.3	*E242* $m = 0.0055$	$\varnothing$
E265 $c = 0$	150	150	164.6	*E265* $c = 0$	$\varnothing$
E281 $\mu_{\Delta p} = 0.9, \mu_f = 2.7, \mu_n = 17.8$	35	35.3	39	*E281* $\mu_{\Delta p} = 0.99, \mu_f = 3.14, \mu_n = 18.56$	$\varnothing$
EY244 stuck open	131.6	131.6	131.7	EY244 stuck open	$\{abort\}$
FAN416 failed	80.8	80.8	84.2	EY275 stuck open, FAN416 failed	$\{abort\}$
FAN416 overspeed	91	91.1	96.7	FAN416 overspeed	$\{abort\}$
FAN416 underspeed	101	101.1	115.7	FAN416 underspeed	$\varnothing$
INV2 failed	111	111	113.4	CB262 failed, INV2 failed	$\{abort\}$
IT240 $m = 0.005$	90	106.6	166.6	*IT240* $m = 0.0047$	$\{abort\}$
IT240 $\mu_{\Delta p} = 7.8, \mu_f = 3.1, \mu_n = 6.2$	35	35	95	*IT240* $\mu_{\Delta p} = 7.77, \mu_f = 3.08, \mu_n = 5.98$	$\{abort\}$
IT267 $\Delta p = -1$	40	40.2	100.2	*IT267* $\Delta p = -1$	$\{abort\}$
IT267 $m = 0.015$	50	53.2	113.2	*IT267* $m = 0.015$	$\{abort\}$
IT267 $\mu_{\Delta p} = -0.3, \mu_f = 3.1, \mu_n = 15$	40	40.4	100.4	*IT267* $\mu_{\Delta p} = -0.3, \mu_f = 3.11, \mu_n = 14.77$	$\varnothing$
IT281 $\Delta p = 0.2$	120	120.9	180.9	*IT281* $\Delta p = 0.2$	$\varnothing$
IT281 $\mu_{\Delta p} = -1, \mu_f = 3, \mu_n = 5.2$	50	50.3	110.3	*IT281* $\mu_{\Delta p} = -1, \mu_f = 3, \mu_n = 5.2$	$\{abort\}$
ST516 $c = 840$	170	209.6	209.6	*ST516* $c = 840$	$\{abort\}$

(Daigle *et al.*, 2007) M. J. Daigle, X. D. Koutsoukos, and G. Biswas. Distributed diagnosis in formations of mobile robots. *IEEE Trans. on Robotics*, 23(2):353–369, April 2007.

(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 *et al.*, 2010) M. Daigle, I. Roychoudhury, G. Biswas, X. Koutsoukos, A. Patterson-Hine, , and S. Poll. A comprehensive diagnosis methodology for complex hybrid systems: A case study on spacecraft power distribution systems. *IEEE Transactions of Systems, Man, and Cybernetics, Part A*, 4(5):917–931, September 2010.

(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.*, 2010) S. Poll, A. Feldman, D. Garcia, J. de Kleer, T. Kurtoglu, and S. Narasimhan. Second international diagnostics competition – DXC'10. In *Proceedings of the 21st International Workshop on Principles of Diagnosis*, October 2010.

(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.

(Roychoudhury *et al.*, 2008) I. Roychoudhury, G. Biswas, and X. Koutsoukos. Comprehensive diagnosis of continuous systems using dynamic bayes nets. In *Proceedings of the 19th International Workshop on Principles of Diagnosis*, pages 151–158, September 2008.

(Roychoudhury *et al.*, 2009) I. Roychoudhury, G. Biswas, and X. Koutsoukos. Designing distributed diagnosers for complex continuous systems. *IEEE Transactions on Automation Science and Engineering*, 6(2):277–290, April 2009.

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