Intelligent Data Visualization for Cross-Checking Spacecraft System Diagnoses

James C. Ong[1], Emilio Remolina[2] and David Breeden[3]
Stottler Henke Associates, Inc., San Mateo, CA, 94404

Brett A. Stroozas[4]
Stroozas FlightOps, Walnut Creek, CA, 94598

and

John L. Mohammed[5]
John Mohammed Consulting, Sacramento, CA, 95758

Any reasoning system is fallible, so crew members and flight controllers must be able to cross-check automated diagnoses of spacecraft or habitat problems by considering alternate diagnoses and analyzing related evidence. Cross-checking improves diagnostic accuracy because people can apply information processing heuristics, pattern recognition techniques, and reasoning methods that the automated diagnostic system may not possess. Over time, cross-checking also enables crew members to become comfortable with how the diagnostic reasoning system performs, so the system can earn the crew's trust. We developed intelligent data visualization software that helps users cross-check automated diagnoses of system faults more effectively. The user interface displays scrollable arrays of timelines and time-series graphs, which are tightly integrated with an interactive, color-coded system schematic to show important spatial-temporal data patterns. Signal processing and rule-based diagnostic reasoning automatically identify alternate hypotheses and data patterns that support or rebut the original and alternate diagnoses. A color-coded matrix display summarizes the supporting or rebutting evidence for each diagnosis, and a drill-down capability enables crew members to quickly view graphs and timelines of the underlying data. This system demonstrates that modest amounts of diagnostic reasoning, combined with interactive, information-dense data visualizations, can accelerate system diagnosis and cross-checking.

Nomenclature

A	= Amperes
DA	= diagnostic algorithm
V	= volts
W	= Watts

[1] Group Manager, 951 Mariners Island Blvd., Suite 360, San Mateo, CA 94404, AIAA Member.
[2] AI Project Manager, 951 Mariners Island Blvd., Suite 360, San Mateo, CA 94404.
[3] AI Software Engineer, 951 Mariners Island Blvd., Suite 360, San Mateo, CA 94404.
[4] President, 420 Augustus Ct., Walnut Creek, CA 94598, AIAA Member.
[5] Consultant, 7533 Shelby Street, Sacramento, CA 95758.

I. Motivation

Future space missions will carry crews far from Earth, so crew members will need to operate with greater autonomy and carry out some system management tasks currently performed by ground-based flight controllers. Intelligent diagnostic reasoning systems can increase crew autonomy by detecting and isolating component failures automatically, based on sensor data, previous commanding, and other data. However, any reasoning system is prone to error when it attempts to solve problems that lie outside of its knowledge base. For example, some diagnostic systems can detect and isolate components that fail abruptly and completely but have trouble isolating intermittent problems, gradual degradations, or simultaneous failures.

Because reasoning systems are fallible, crew members and flight controllers must be able to cross-check automated diagnoses by considering alternate diagnoses and analyzing supporting or rebutting evidence. Cross-checking improves diagnostic accuracy because crew members can apply information processing heuristics, pattern recognition techniques, and reasoning methods that the automated diagnostic system may not possess. Over time, cross-checking also enables crew members to become comfortable with how the diagnostic reasoning system performs, so the system can earn the crew's trust. Intelligent data visualization software can help crew members to cross-check automated diagnoses more quickly and accurately by identifying relevant data patterns automatically and presenting them in ways that are easy to discern and interpret.

II. Diagnosis Competition Testbed

In 2009, NASA hosted the first annual Diagnosis Competition[1]. Competing software systems, called diagnostic algorithms (DAs), analyzed prerecorded sensor data streams and system commands in real-time to detect and diagnose injected system faults. The diagnoses generated by each DA were collected and scored automatically.

The competition was divided into several tiers. During the "Tier 2" competition, DAs received sensor and command data from the Advanced Diagnostics and Prognostics Testbed (ADAPT) system[2], an experimental testbed at NASA Ames Research Center that supports automated diagnosis research. The ADAPT system, shown in Figure 1, is comprised of:

- Three batteries (named BAT1, BAT2, BAT3),
- Electrical loads such as fans, pumps, and lights, grouped in Load Bank 1 and Load Bank 2,
- Relays (e.g., EY141 in the upper left of Figure 1), which were commanded to open or close to control the connection of batteries to load banks and individual loads,
- Circuit breakers (e.g., CB136 in the upper left of Figure 1), and
- Sensors, shown as circular icons in Figure 1, which measure variables such as current (IT), voltage (E), temperature (TE), fan speed (ST), pump flow (FT), and the open/close positions of relays (ESH) and circuit breakers (ISH).

III. Strategies and Heuristics for Cross-Checking System Diagnoses

To gain a better understanding of the cross-checking process, we manually cross-checked the automatically-generated DA diagnoses for about twenty Diagnosis Competition scenarios. We then reflected upon and documented the sensor, command, and schematic data we reviewed, the data patterns we noticed, the diagnoses and other inferences we generated, and the conclusions we drew.

For example in Tier 2 experiment #824, one of the DAs hypothesized that fan FAN416 failed at t=138.5. The relevant portion of the ADAPT circuit and associated sensor data are shown in Figure 2. We identified two alternate diagnoses: the fan speed sensor might have failed or an upstream relay such as EY275 might have failed, cutting off power to the fan. We rejected the first alternate diagnosis because the reported fan speed decreased steadily to zero at a rate that was inconsistent with fan speed sensor failure modes. We rejected the second alternate hypothesis, failure of an upstream relay, because none of the relay position sensors such as ESH275 changed from closed to open near the time of the hypothesized fault. We accepted the original automated diagnosis, a fan failure, based on the following supporting evidence as shown in the graphs in Figure 2:

- Upstream AC current sensor IT267 decreased by 1 A, which yielded a power loss of 120 W (120 V * 1 A). This voltage drop equaled the fan's nominal power consumption and was consistent with the fan no longer operating.
- Upstream DC current sensors IT261 and IT340 decreased by 5 A, which yielded a power loss of 120 W (24 V * 5 A), equal to the fan's nominal power consumption.
- Fan speed sensor ST516 immediately began decreasing, reaching zero about 23 seconds later.
- Small DC voltage increases (0.2 V) were also seen in upstream indicators E340 and E261 at the time of the fault, consistent with an overall load reduction.

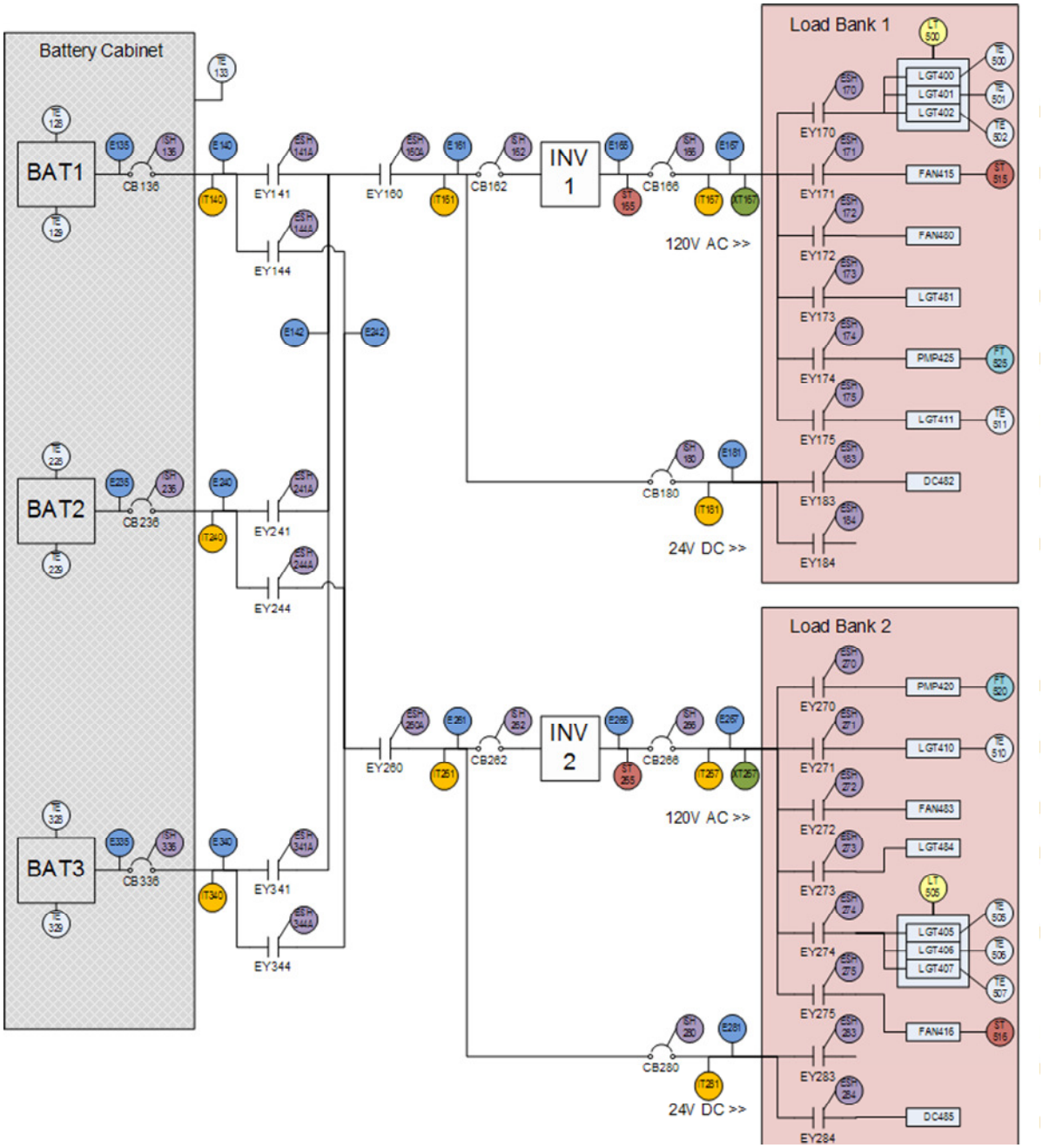

Figure 1. Schematic of the ADAPT electrical system used by the Diagnosis Competition, Tier 2.

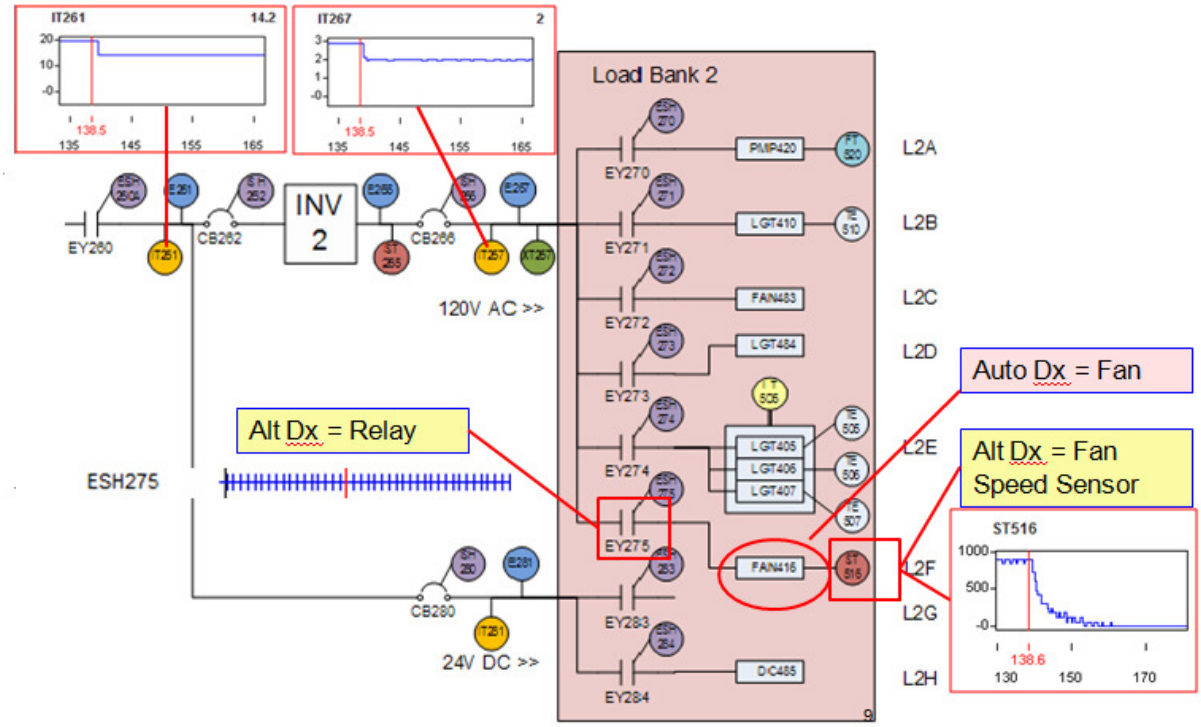

Figure 2. Portion of the ADAPT circuit near fan FAN416, suspected of failing, and graphs of sensor data that are relevant to cross-checking this suspected failure. Rectangles indicate alternate diagnoses.

Based on our analysis of the cross-checking process, we identified generic cross-checking strategies and domain-specific cross-checking heuristics, described in Table 1, which our data visualization system should support.

Table 1 – Generic strategies and domain-specific heuristics for cross-checking system diagnoses.

1	Prioritize diagnoses and cross-checking	By prioritizing possible problems, crew members can attend to the most critical and urgent situations first. For example, a crew member might ask: Do I believe the DA is reporting something worth investigating? If the diagnosis is true, what problems might it cause that affect safety, mission success, or system health?
2	Identify symptoms underlying diagnosis	Cross-checking a diagnosis includes generating and evaluating alternate diagnoses that explain the symptoms (sensor data) of the original diagnosis. For example, a functional component might be diagnosed as faulty if a sensor reports that the component's behavior is abnormal. The crew member can cross-check this diagnosis by searching for other possible explanations for the sensor readings, such as a faulty sensor or incorrect inputs to the suspect component.
3	Assess plausibility of symptoms	If the current and/or historical sensor data indicate a physically unlikely event or state, the sensor(s) may be faulty. For example, if a sensor reports a sudden change in temperature that is more rapid than could possibly occur, the temperature sensor might be exhibiting an offset error.
4	Recognize sensor failure signatures	Characteristic sensor readings (signatures) can provide evidence for particular types of sensor failures. For example, a current sensor operating normally reports readings that vary slightly over time. By contrast, a stuck current sensor reports the same exact value at all times.
5	Understand the reasoning behind the original diagnosis	The reasoning that supports the initial diagnosis may assert or presume various events and states. The crew member can cross-check the diagnoses by verifying these events and states. For example, if the diagnosis asserts a particular root cause that leads to a proximal cause, the cross-check could search for evidence for the root cause and for the proximal cause.

6	Hypothesize and evaluate alternate diagnoses	One can cross-check a diagnosis by hypothesizing alternate diagnoses and determining whether they are more plausible than the initial diagnosis, based on the presence or absence of data patterns that support or rebut each diagnosis.
7	Understand the overall pattern of problems and events	Analyze other data or commands that might be related to the current problem being diagnosed or cross-checked. The distribution of other problems across the system can provide suggestive evidence for the location of the root cause. If there are no other problems reported in the system, it is likely that the root cause is "near" the symptom. Conversely, if there are many *related* problems spread widely across the system, the root cause might be far from the symptom being investigated. For example, if voltage sensor readings have unexpectedly dropped to zero at many locations, analyze the circuit's topology to identify possible single failures that could explain these readings.
8	Look for abrupt changes	Search for abrupt changes that occurred shortly before the component failure or anomalous sensor data readings. Some components respond quickly to changes to their inputs or environment, so when their behaviors change abruptly, the causes might be abrupt and recent. For example, if a measured current or voltage decreases abruptly, look for an abrupt change in a power load, power distribution, or power supply.
9	Consider earlier events if necessary	If no abrupt causes occurred shortly before the symptom was detected, it may be necessary to consider earlier events. For example, the problem might have occurred before it was first noticed, or there might be a delay between the earlier event and the onset of the problem.
10	Search for components that might cause a component to misbehave	If component is not behaving normally, search for other components that might have caused this component's misbehavior. For example, suppose that an electrical load, such as a fan, is not running at the correct speed. This might be because the fan is not receiving appropriate power. Search for upstream stuck-open relays or circuit breakers that might prevent the fan from receiving power.
11	Search for possible causes that are near the symptoms	When seeking faulty component(s) that explain symptoms such as unexpected sensor readings, search first for faulty components that are close to the component associated with the sensor. Although a faulty component can cause both nearby and distant components to misbehave, it is easier to reason about shorter paths of connected components, so it is efficient to evaluate nearby causes first. If the cause was a distant component, it is often the case that many other components will be affected as well.
12	Check other sensor data for consistency with hypothesized fault	If sensor data indicate a faulty component, look for other evidence that corroborates this suspected fault. If the other data suggest that this component is operating normally, the sensor attached to the component may instead be faulty. For example, suppose that a relay is suspected of being stuck open because its associated relay position sensor reports that the relay is open and the relay was not commanded to be open. If the relay really is open, the current through all sensors downstream of the relay should be zero. If non-zero current is detected anywhere downstream of the relay, the relay must be passing current, so the root cause cannot be that the relay is stuck open
13	When explaining symptoms, consider specific failure modes	Many components have fault modes that can be detected in the sensor data. For example, a common failure mode for a relay is to remain stuck in an open position and unable to pass current, even when commanded to close. For example, if the symptom is an electrical load that operates in a way that suggests insufficient power, search for upstream stuck open relays or open circuit breakers that might have cut off power to the load.
14	Divide and Conquer	Sometimes there are states, events, proximal causes, or partial or less-specific diagnoses that, if confirmed or rejected, could prove or reject whole sets of candidate diagnoses. For example, if a voltage sensor indicates a non-zero voltage at a particular location, one can rule out the possibility that any relay or circuit breaker upstream of that location has failed in an open position.

15	Compare behavior of component with reference values and relationships	Sometimes, one can assess whether the behavior of a component is normal by comparing its behavior as reported by sensors with known reference values or relationships. Reference values can specify normal ranges or temporal patterns for individual variables. Multivariate relationships specify expected mathematical or logical relationships among two or more variables, possibly measured at different times. For example, the speed of a fan may be a time-delayed function of the power provided to the fan.
16	Compare the behavior of a component with a similar component	Sometimes, one can assess whether the behavior of a component is normal by comparing its apparent behavior as reported by sensors with the behavior of another component of the same type that is operated similarly. For example, this comparison might be appropriate if the system contains similar components or subsystems arranged in parallel. Or, the component's behavior could be compared with same component's behavior during a previous, similar situation.
17	Exploit physical constraints	Exploit reliable physical constraints on state variables to support diagnostic deduction. If the constraints are not satisfied, one or more assumptions that underlie the constraints must be false. Example constraints include Kirchoff's current law (the sum of all currents at any point must sum to zero) and conservation of mass (fluid flows at any point must sum to zero).

IV. Interactive Data Visualization

After we identified cross-checking strategies and heuristics, we developed Intelliviz, an interactive data visualization system to help users cross-check diagnoses. This system presented three integrated user interface windows. The *Temporal Data Display*, shown in Figure 3, contains timelines and time-series graphs that display time-stamped injected faults, continuous and discrete sensor data, commands, and diagnoses generated by Diagnostic Algorithms. The *Control Panel*, shown in Figure 3, enables users to specify criteria for selecting a subset of the sensor variables to display at the top of the Temporal Data Display. In addition, icons for sensors and components that satisfy the selection criteria are highlighted in the Circuit Diagram. The *Circuit Diagram*, shown in Figure 4, displays an icon for each functional component and sensor in the electrical distribution system. Icons are color-coded to show information about each component or sensor.

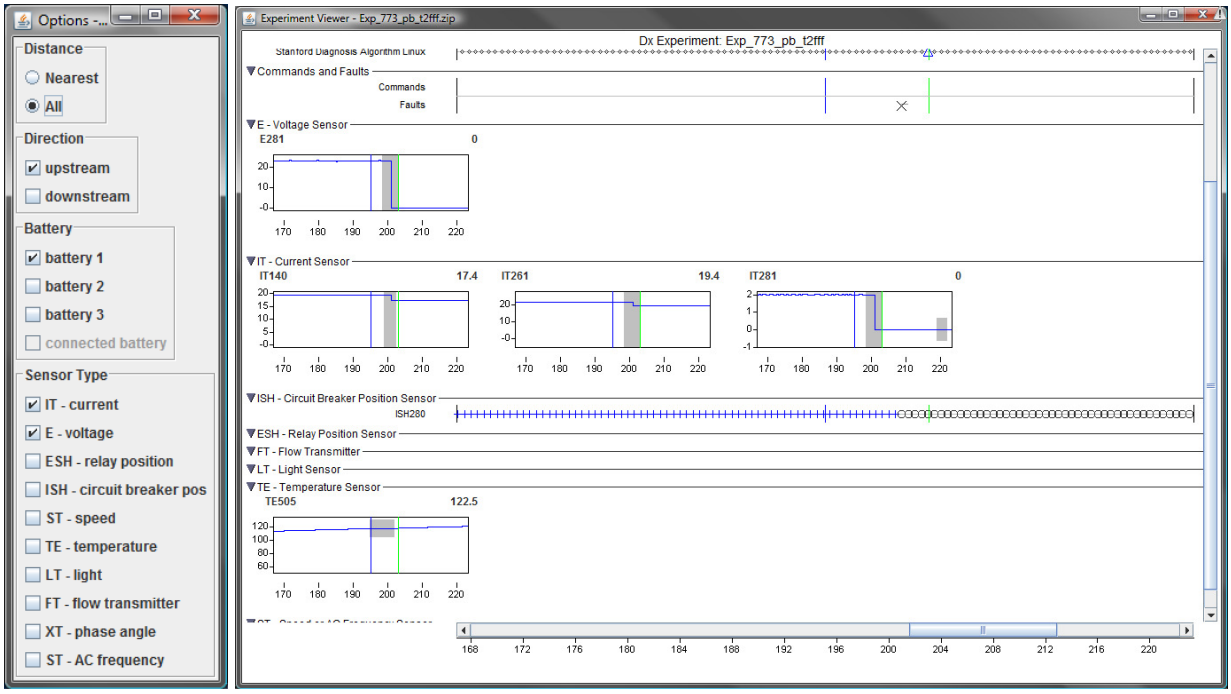

Figure 3. The control panel at left enables users to specify criteria for filtering or highlighting data in the schematic display and in the time-oriented data display. Scrollable timelines and time-series graphs at right help users see temporal data patterns such as coincident sensor data changes.

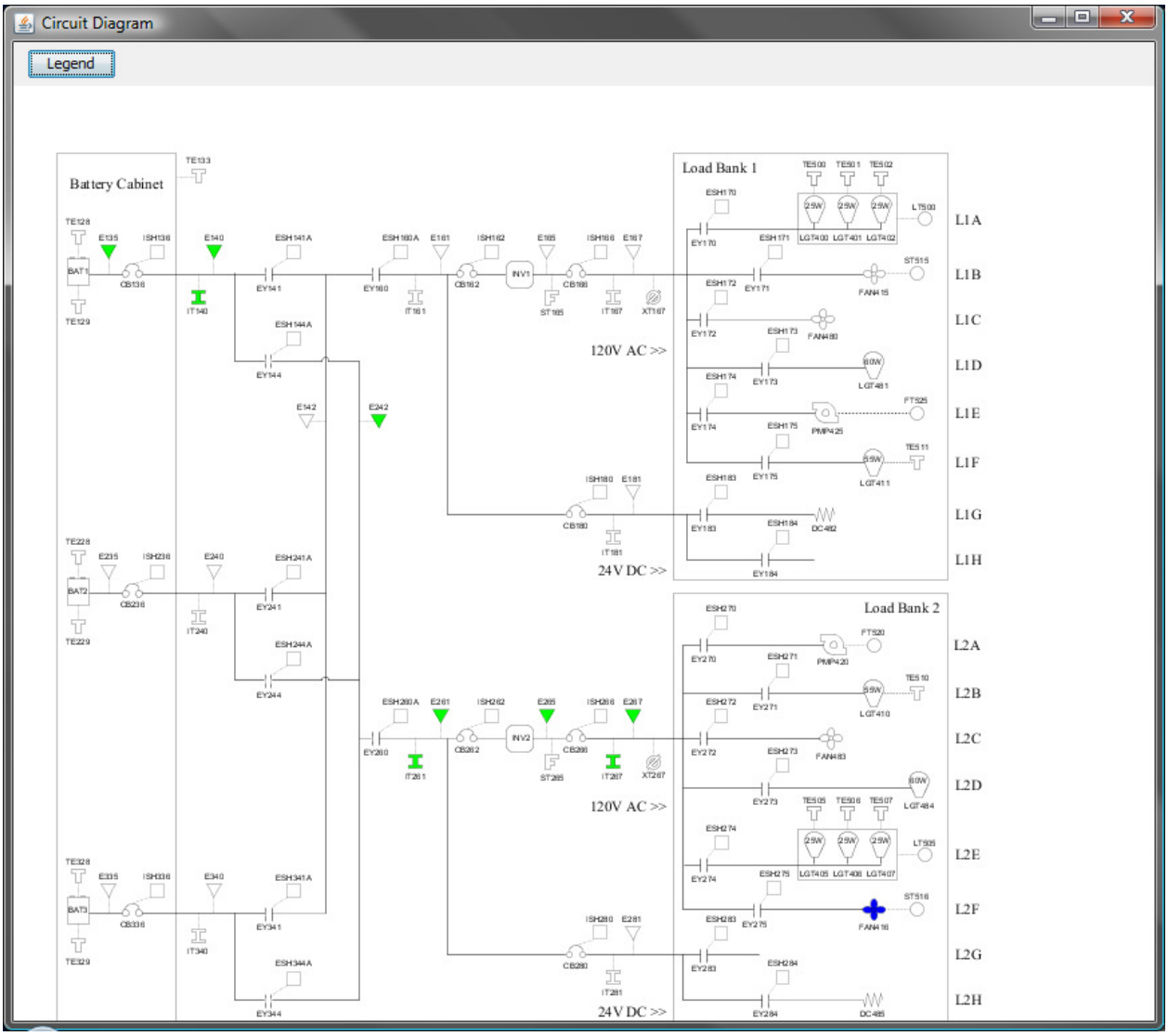

Figure 4. Green icons in the schematic show related sensors that satisfy user-specified criteria. Users can mouse-over icons to re-specify the reference component, highlighted in blue, and recolor the related sensors.

A. Automatic Detection and Highlighting of Changes in Sensor Data

Intelliviz automatically detects changes in each sensor's measurements and highlights the time- series graphs to show the type and timing of each change. Three types of data changes are detected: Changes in value, changes in slope, and "stuck" sensor values in which the variation drops to zero. Due to noise, each sensor's values constantly vary slightly over time, and determining when a *notable* change occurs is somewhat subjective. When developing algorithms for detecting changes, our goal was to identify changes at the same points in time that a human expert would identify as being both meaningful and likely to correspond to an identifiable physical event.

To detect abrupt changes in a sensor's time-series values, the software applies a low-pass filter to the signal and then searches for times when the 1st derivative of the smoothed signal peaks and exceeds a threshold specified for that sensor. To detect an abrupt change in the *slope* of the sensor's time-series data, the software searches for times when the second derivative of the smoothed signal peaks and exceeds a threshold. To detect a "stuck" sensor, the software searches for times when the first derivative of the smoothed signal is below a small threshold for n consecutive samples. In effect, this algorithm detects when the variation goes to zero.

B. Automatic Selection of Graphs and Timelines with Data Changes

We enhanced the Temporal Data Display to show the subset of the graphs and timelines that exhibited one or more changes in value, slope, or variation during a user-specified time period. This feature enables users to quickly select and focus on the subset of sensor data variables that changed during a time period of interest. For example, the user might want to search for sensor data changes that occurred just before an automated diagnosis was generated or just before the values of another sensor variable changed abruptly.

In Figure 3, the Temporal Data Display applies a filter to display only the graphs and timelines that show data changes during the time interval shortly before circuit breaker CB280 failed open at around t=200. This time interval is marked by green and blue vertical reference lines. During this time interval, all of the following automatically detected changes in sensor values are consistent with circuit breaker CB280 failing in an open position:

- Downstream voltage sensor (E281) dropped to zero
- Upstream current sensors (IT140 and IT261) decreased
- Downstream current sensor (IT281) dropped to zero
- Circuit breaker position sensor (ISH280) indicated a change from closed to open

C. Displaying Data Related to a Component or Sensor

The data visualization system enables users to quickly browse sensor data that are related to a particular component or sensor in some way. For example, using the Control Panel, users can specify that the Schematic and Temporal Data Display should show data from voltage sensors that are upstream or downstream of a reference component or sensor. Users can quickly select the reference component or sensor by mousing over its icon in the schematic. This feature accelerates analysis by displaying the sensor data that relate to the user's focus of attention.

D. Interactive Color-Coded Schematics

We implemented an interactive circuit diagram so that the display could be color-coded dynamically in response to user requests or Intelliviz's own reasoning. The Related Data feature highlights the sensor icons that satisfy user-specified selection criteria while displaying timelines and time series graphs for those sensors in the Temporal Data Display. By highlighting the icons of sensors that satisfy some criteria, the user can see significant spatial patterns in the data. For example, in Figure 4, green is used to highlight the icons of current and voltage sensors that are upstream of the user-selected reference fan component, highlighted in blue. The user can specify additional selection criteria such as including only sensor data that changed shortly before the anomaly.

V. Intelligent Data Visualization

We enhanced the interactive data visualization system described above with automated analyses that generate and display plausible alternate hypotheses and related data.

A. Automated Identification of Alternate Hypotheses and Related Data Patterns

We designed a simple algorithm for detecting data patterns that support cross-checking. The algorithm accepts as input the sensor data, commands, and user-selected automated diagnosis. It outputs:

- *Symptoms* – data patterns that might be explained by the automated diagnosis,
- *Alternate diagnoses* – that the user might want to consider and compare to the automated diagnosis,
- *Evidence* – data patterns that support or rebut the original diagnosis and alternate diagnoses. Evidence is not assumed to be conclusive, so the existence of supporting evidence does not mean that a hypothesis is necessarily true, and the existence of rebutting evidence does not mean that a hypothesis is necessarily false, and
- *Links* – that associate symptoms and evidential data patterns with the original and alternate diagnoses.

This algorithm relies on three types of rules:

- *Symptom Rules* – These rules are triggered by the presence of a type of automated diagnosis and certain data patterns. If a symptom rule fires, it creates (or finds) one or more data pattern objects that represent symptoms that the original diagnosis might explain. It also creates a *symptom link* between the original diagnosis and each symptom data pattern object.
- *Hypothesis Rules* – These rules are triggered by the presence of symptom data pattern objects and optional additional data patterns. If a hypothesis rule fires, it creates or finds a hypothesis that might explain the

symptom along with an *explanation link* that relates the hypothesis to the symptom. In our prototype implementation, all hypotheses are alternate diagnoses. However, a more sophisticated reasoning algorithm might generate hypotheses that describe beliefs about the system state or history that are not specific diagnoses. For example, a rule might hypothesize that a particular component is not receiving power. This hypothesis describes a state condition that is not a specific diagnosis.

- *Support Rules* – These rules trigger on the presence of a type of hypothesis object and optional additional conditions. The presence of the data pattern in the data can provide evidence that either supports or rebuts the hypothesis. If the inverse of the rule is true, the absence of the data pattern also provides evidence that rebuts or supports the hypothesis as summarized in Table 2 below:

Table 2. Types of support rule inferences.

	Inverse is true?	Inverse is not true?
Supports?	*Presence* of the data pattern *supports* the hypothesis. *Absence* of the pattern *rebuts* the hypothesis.	The *presence* of the data pattern *supports* the hypothesis.
Rebuts?	*Presence* of the data pattern *rebuts* the hypothesis. *Absence* of the pattern *supports* the hypothesis.	The *presence* of the data pattern *rebuts* the hypothesis.

Figure 5 summarizes the sequential application of symptom, hypothesis, and support rules:

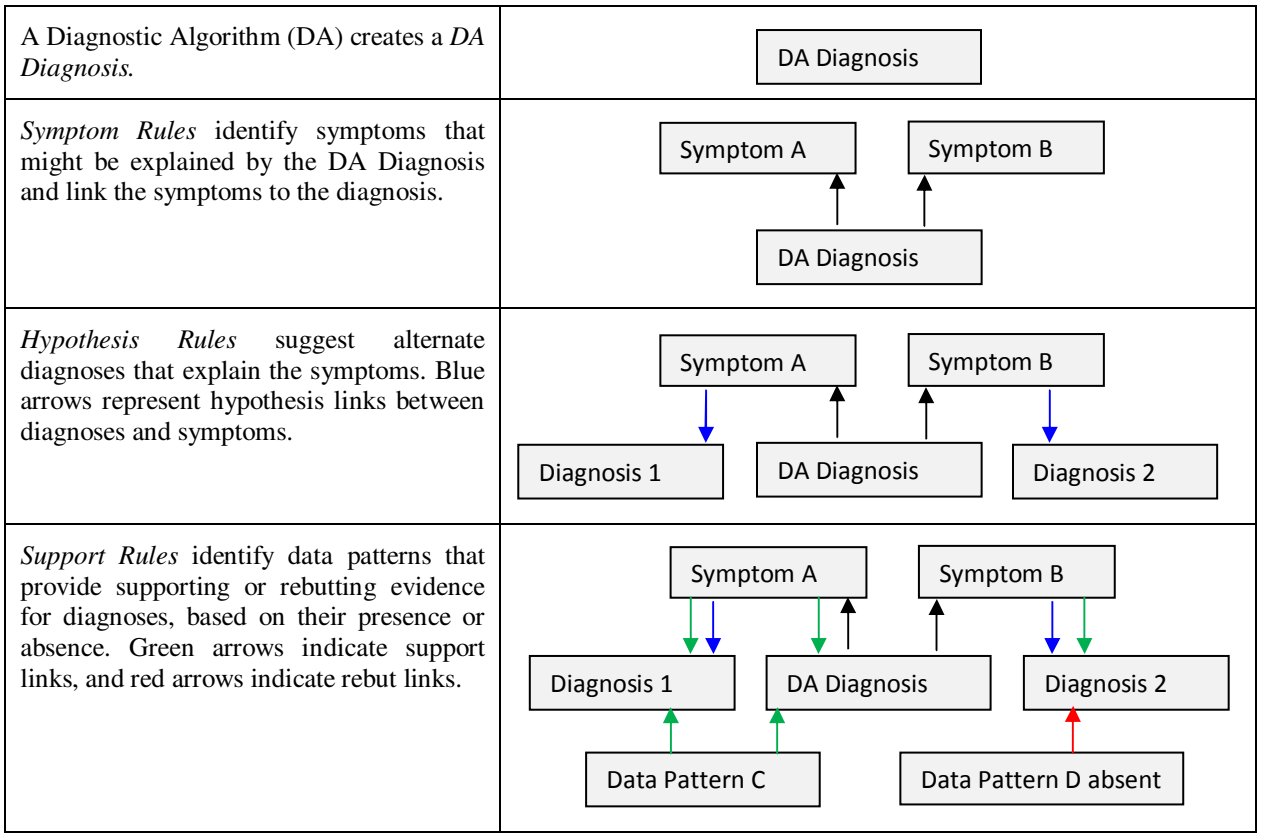

Figure 5. Sequential application of symptom, hypothesis, and support rules.

In our prototype, we implemented 13 sets of rules that relate a suspected faulty component to data reported by a sensor that measures the component. For example, one of the symptom rules in this set is:

IF:

1. The DA Diagnosis is: A CIRCUIT-BREAKER failed in mode STUCK-CLOSED, and
2. The following data pattern is present:
   - There is a sensor of type CB-POSITION-SENSOR that is linked to the CIRCUIT-BREAKER,
   - There is a data pattern for the sensor variable: EXISTS_VALUE CLOSED, and
   - The start time of the sensor data pattern precedes the hypothesis by less than 5 seconds

THEN assume that the DA diagnosis might have been generated to explain this data pattern (symptom).

We implemented 48 sets of rules that relate a suspected faulty component to data reported by a sensor that measures *another component* whose behavior might be affected by the suspected component. For example, one of the hypothesis rules in this set is:

IF the following symptom is present:

- There EXISTS a CURRENT-TRANSMITTER sensor variable that exhibits the data pattern: EXISTS ABRUPT_VALUE_DECREASE_TO_ZERO, and
- There EXISTS a CIRCUIT-BREAKER that is upstream of the CURRENT-TRANSMITTER

THEN hypothesize that a CIRCUIT-BREAKER might have failed open within 5 seconds of the CURRENT-TRANSMITTER ABRUPT_VALUE_DECREASE_TO_ZERO which would explain this symptom.

We implemented 18 sets of rules that relate a suspected faulty sensor to the sensor's data. For example, one of the support rules in this set is:

IF there exists a hypothesis: CB-POSITION-SENSOR failed in mode STUCK

THEN the following data pattern, if present, would provide supporting evidence for this hypothesis:

- There exists a data pattern for the CB-POSITION-SENSOR sensor variable: STUCK. and
- The start time of the sensor data pattern precedes the hypothesis by less than 5 seconds,

The inversion of this rule is true, so the absence of a STUCK CB-POSITION-SENSOR data pattern provides evidence that rebuts the hypothesis.

For each symptom rule, there is:

- A related hypothesis rule that hypothesizes a particular diagnosis if a symptom data pattern is present, and
- A related evidence rule that asserts that a data pattern, if present (or possibly absent), provides evidence for or against a hypothesized diagnosis.

B. Display of Significant Data Patterns

We augmented the interactive data visualization system with additional user interface windows that enable the user to request analyses and visualizations for cross-checking. The cross-checking process begins when the user clicks on a timeline symbol to select one of the DA diagnoses as shown in Figure 6.

If the diagnosis specifies more than one possible fault, the user is prompted to select the fault to cross-check. Then, the system applies symptom, hypothesis, and support rules to identify data patterns and alternate diagnoses, which are linked by symptom links, hypothesis links, and support/rebut links. This information is then summarized graphically in a Diagnostic Rationale Matrix as shown below in Figure 7.

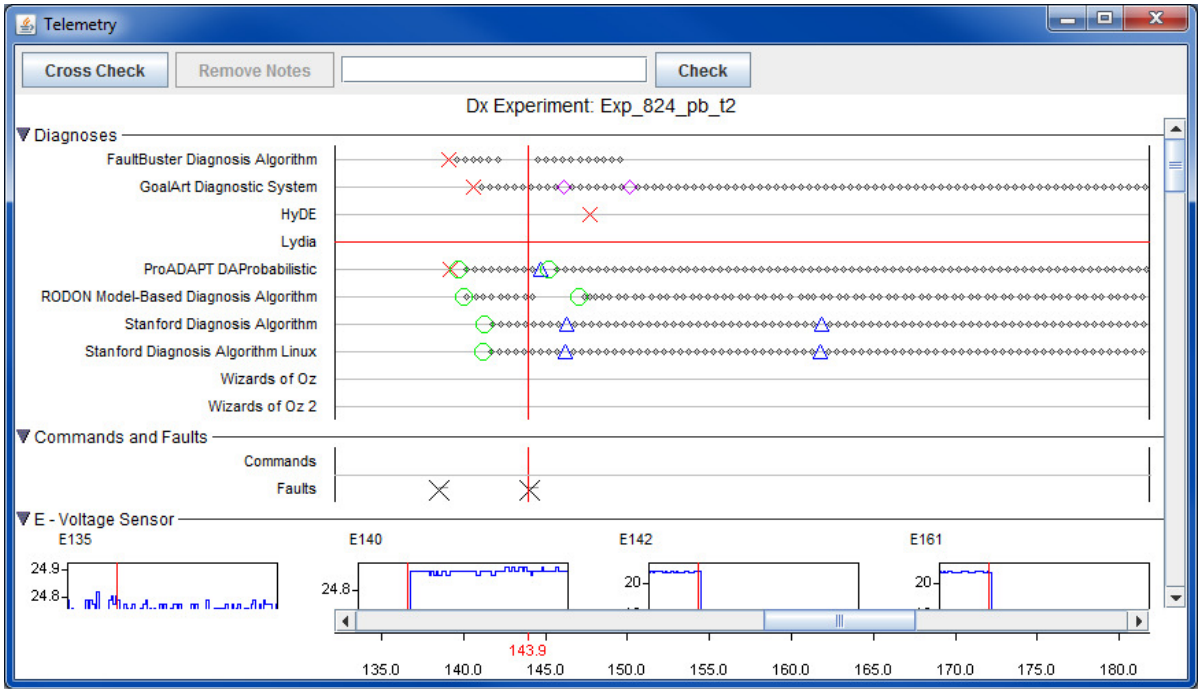

Figure 6. Users can click on a symbol in a timeline to select a DA diagnosis to cross-check

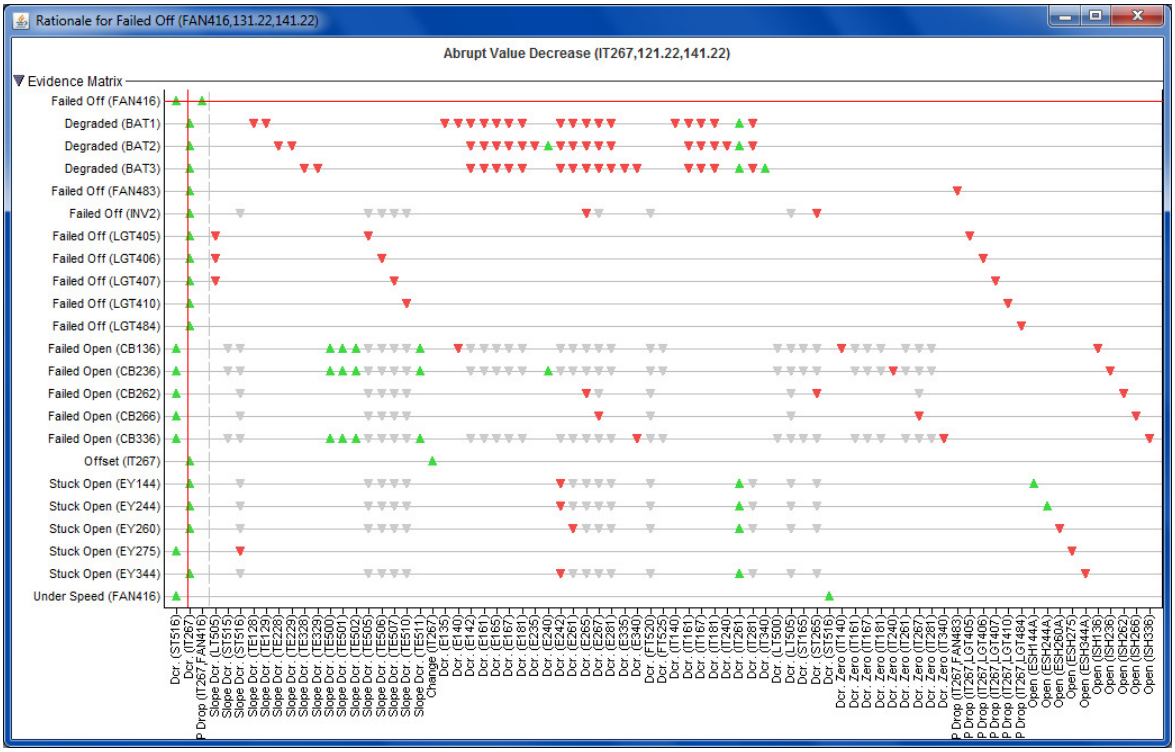

Figure 7. The interactive diagnostic rationale matrix summarizes the hypotheses and associated significant data patterns identified by Intelliviz. Users can click on a symbol to see each data pattern.

In this Diagnostic Rationale display:

- The top row, labeled Failed Off (FAN416), corresponds to the original diagnosis to be cross-checked.
- The other rows correspond to alternate diagnoses (other possibly faulty components) that each explain some or all of the symptoms that were explained by the original diagnosis.
- Each column corresponds to a pattern in the data associated with a sensor, such as an abrupt change.
- Each symbol represents the relationship between a data pattern and a diagnosis:
  - An upward triangle indicates that the data pattern is present.
  - A downward triangle indicates that the data pattern is absent.
  - Green indicates that the presence or absence of the data pattern provides evidence for the diagnosis.
  - Red indicates evidence against the diagnosis.
- Symbols to the left of the vertical gray line represent symptoms that are explained by the selected diagnosis.
- Symbols to the right represent data patterns whose presence or absence provides evidence for or against the diagnoses.

This graphical summary enables users to quickly review the data patterns and diagnoses. Specifically, the user can scan the labels along the horizontal and vertical axes to see the data patterns and diagnoses identified by Intelliviz. The user can scan each row of symbols to see the evidence for and against a diagnosis. For example, the user might focus his or her attention on the most likely diagnoses, which have supporting evidence and relatively little rebutting evidence. The user can also scan each column of symbols to see how each data pattern supports differential diagnosis among competing hypotheses.

Intelliviz provides drill-down capability. When users click on a symbol in the Diagnostic Rationale Matrix:

1. A time-series graph or timeline that shows the associated data pattern is displayed at the top of the array of timelines and time series graphs, and
2. The icons of the associated sensor and the suspected component are highlighted in the schematic.

Because the time-series graph is displayed with other graphs aligned by time in a Temporal Data Display, the user can easily compare these data with data for other variables. Our current implementation only displays timelines and time-series graphs. However, one could extend this idea to display any kind of graphical data display.

VI. Conclusions

Intelligent data visualization technology can significantly accelerate problem diagnosis and cross-checking. Unlike automated diagnostic reasoning systems that isolate specific faults, the diagnostic reasoning portion of our system merely seeks to identify the plausible diagnoses worth considering and the relevant data patterns that the user should review. Because this is a less difficult diagnostic task, the diagnostic reasoning embedded within our system is simpler and easier to understand. Thus, modest amounts of diagnostic reasoning, combined with interactive, information-dense data visualizations, provide a practical way of accelerating the cross-checking of system diagnoses. When automated diagnosis is unavailable, intelligent data visualization can also enable more rapid and effective diagnosis by crew members and flight controllers.

Acknowledgments

Funding for this research was provided by NASA contract NNX09CA07C**.**

References

[1]Kurtoglu, T., Narasimhan, S., Poll, S., Garcia, D., Kuhn, L., de Kleer, J., Gemund, A., Feldman, A., "First International Diagnosis Competition – DXC'09", *20th International Workshop on Principles of Diagnosis (DX-09),* Stockholm, Sweden, June 14-17, 2009.

[2]S. Poll, S., Patterson-Hine, A., Camisa, J., Garcia, D., Hall, D., Lee, C., Mengshoel, O. J., Neukom, C., Nishikawa, D., Ossenfort, J., Sweet, A. , Yentus, S., Roychoudhury, I., Daigle, M., Biswas, G., and Koutsoukos, X., "Advanced Diagnostics and Prognostics Testbed," *Proc. of the 18th International Workshop on Principles of Diagnosis (DX-07)*, Nashville, TN, May 2007.