A Framework to Debug Diagnostic Matrices

Anuradha Kodali[1], Peter Robinson[2], and Ann Patterson-Hine[2]

[1]*SGT Inc., NASA Ames Research Center, Moffett Field, CA, 94035, USA*
anuradha.kodali@nasa.gov

[2]*NASA Ames Research Center, Moffett Field, CA, 94035, USA*
peter.i.robinson@nasa.gov
ann.patterson-hine@nasa.gov

ABSTRACT

Diagnostics is an important concept in system health and monitoring of space operations. Many of the existing diagnostic algorithms utilize system knowledge in the form of diagnostic matrix (D-matrix, also popularly known as diagnostic dictionary, fault signature matrix or reachability matrix). The D-matrix maps tests on observed conditions to failures. This matrix is mostly gleaned from physical models during system development. But, sometimes, this may not be enough to obtain high diagnostic performance during operation due to system modifications and lag and noise in sensor measurements. In such a case, it is important to modify this D-matrix based on knowledge obtained from sources such as time-series data stream (simulated or maintenance data) within a framework that includes the diagnostic/inference algorithm. A systematic and sequential update procedure, diagnostic modeling evaluator (DME) is proposed to modify D-matrix and wrapper/test logic considering the least expensive update first. The user sets the diagnostic performance criteria. This iterative procedure includes conditions ranging from modifying 0's and 1's in the matrix, adding/removing the rows (failure sources)/columns (tests), or modifying test/wrapper logic used to determine test results. We will experiment this framework on ADAPT datasets from DX challenge 2009.

1. INTRODUCTION

Traditionally, diagnostics is performed in the following way: System modeling → List failure causes (faults) → Design tests → Generate D-matrix → diagnosis via inference algorithm (Luo & Pattipati, 2007). Here, the process from system modeling to generate D-matrix is independent of the diagnoser. But, when the diagnostic algorithm based on D-matrix (Singh, Kodali, Choi, Pattipati, Namburu, Chigusa, Prokhorov, & Qiao, 2009) is applied during operations, and the performance is not robust, it is important to reexamine the system model (D-matrix) in terms of its correctness and utility towards diagnosability. Thus, we propose a debugging architecture, termed diagnostic modeling evaluator (DME) that includes the diagnoser and repairs the system model (D-matrix) to suit better diagnostic performance based on new/updated information. This updated information is mostly available after system development or during operation.

D-matrix can be developed from physical models, historical field failure data, service documents, engineering schematics, and Failure Modes, Effects and Criticality Analysis (FMECA) data (Singh, Holland, & Bandyopadhyay, 2011) by establishing causal relationship between faults and tests (Luo, Tu, Pattipati, Qiao, & Chigusa, 2006). Initially, D-matrix is generated from any of these sources (e.g., physical model). The initial model, when developed during system development, ignores lag and noise in sensor measurements during operation and other system advancements during deployment. Then other sources (e.g., operations data (time-series)) that contain these critical changes can be used as reference material in DME framework to repair the initial D-matrix. This provides a debugging environment to the initial model. This also provides an effective platform to represent information from different sources (model-based, data-driven, or knowledge-based) in a unified D-matrix concept.

Diagnostic modeling evaluator (DME) is developed as an automated debugging process to update/repair D-matrix that best suits user-defined performance requirements. This includes assessing the level of fault definitions (component or failure mode level), number of tests required, test logic by considering the thresholds for faulty behavior, and most importantly the fault-test relationships. Conditions (repairs) ranging from modifying 0's and 1's in the matrix, or modifying the rows to accommodate lower-level fault modeling with failure modes, or adding or removing tests, or changing their test logic are identified to experiment for

Anuradha Kodali et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 United States License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

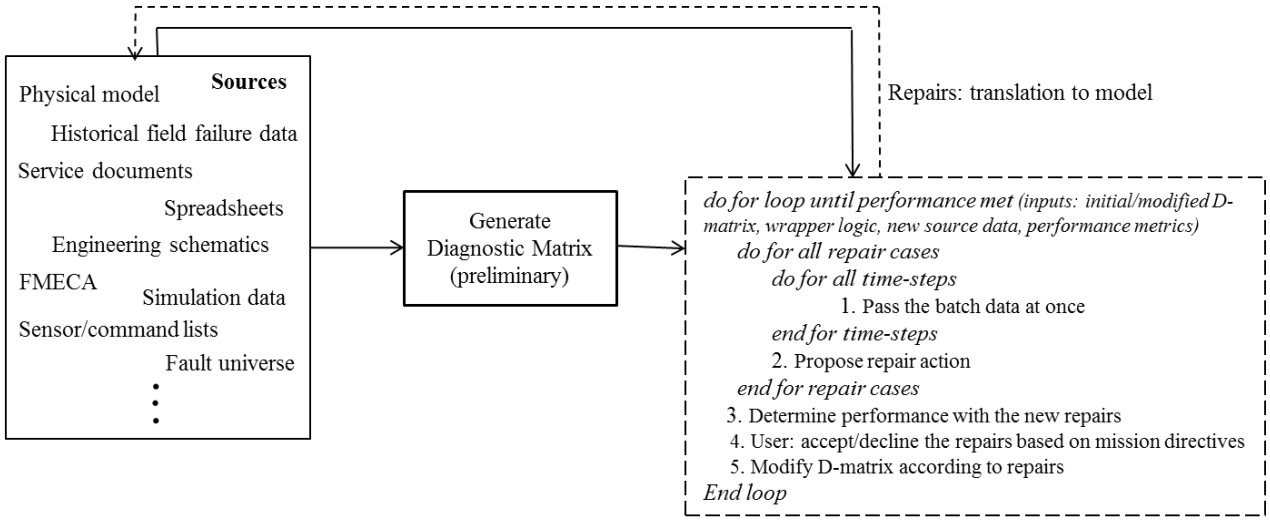

Figure 1. Framework to debug D-matrix

better performance in terms of diagnostics (detection + isolation). This is implemented as an iterative feedback process by tuning D-matrix at every step with repair conditions. Sometimes, more than one repair is applicable on a given iteration. Those repairs are accepted/declined by the user/decision maker based upon their diagnostic performance and also, most importantly, mission directives. The user-defined performance criteria are quantified based on the following metrics: diagnostic efficiency, false positive/false negative rate, diagnostic time and cost. This is again communicated to update procedure and the iterations go on until there are no further changes (shown in Figure 1).

In this paper, section 2 focuses on explaining DME procedure as a debugger and the conditions required to update D-matrix. In section 3, the process of updating D matrix is shown with examples. Two example systems, rover and ADAPT (from DX challenge 2009) are included in this paper.

2. DEBUGGING FRAMEWORK

Conventionally, system model doesn't consider diagnostic utility when developing D-matrix. While developing a robust diagnostic system, it is important for both system modeling and diagnostic process to interact coherently resulting in high detection and isolability performance during operation. To make this idea possible, DME acts as a debugger to the initial D-matrix using the available operational or simulations data. It plays the data in batch mode in order to determine which repairs to make. No single time step decisions are made, though this would be required to utilize these techniques during runtime operation (other data sources can also be used to repair D-matrix). These repairs to D-matrix can be translated back to the initial system model. This is pursued as future work. Here, we will explain the modules and process of DME framework as shown in Figure 1:

2.1. Information Sources

The diagnostic modeling, firstly, starts with building the model from the system information, viz., physical model, historical field failure data, service documents, spreadsheets, engineering schematics, FMECA, sensor/commands list, and simulations data (Singh et al., 2011). D-matrix, in DME context, is built from one source initially and then updated/repaired with the other available sources. These sources can be model-based, data-driven, or knowledge-based informative sources. This repair process can be performed during system development or operational phase. The data collected during real-time operations accommodates lags and noise ignored during system development, thus providing operational information about the relationship between faults and tests. But, sometimes, the information from different sources can be counter-explanative and this needs to be dealt carefully during corrective actions to the D-matrix.

2.2. Generating D-matrix (preliminary)

Here, the initial D-matrix is generated from any of the available sources listed above. For this purpose, it is important to determine the level at which fault modeling is performed for diagnosis. This can be done at sub-system, or component, or failure mode level depending on the system properties and requirements. Also, the testing criterion for each test is formulated based on sensor measurements. These tests can either be threshold, trending, or statistical tests. Subsequently, D-matrix is generated either by hand or by automated software methods like TEAMS Designer (Qualtech Systems Inc.). The D-matrix generated in this step is preliminary and is modified in the next step.

Loop until performance_criteria_met (based on cost effective repairs)

- *Step 1:* Address Row and Column redundancy
  - Remove redundant rows (address row redundancy)
  - Remove redundant columns (address column redundancy)
- *Step 2:* Change rows of Dmatrix
  - Add rows (refine component into its failure modes)
  - Remove rows (abstract failure modes to component level)
- *Step 3:* Change wrapper logic (Fault detection test criteria)
  - Change the threshold-based tests
    - Narrow criteria (address false positive detection)
    - expand criteria (address false negative detection)
  - Change the wrapper logic: threshold to trending or statistical tests (e.g., z-test)
- *Step 4:* Change elements of Dmatrix (Fault isolation relationship)
  - Change Dmatrix element value: 1->0 (address false positive diagnosis)
  - Change Dmatrix element value: 0->1 (address false negative diagnosis)
- *Step 5:* Change columns of Dmatrix
  - Add columns (add test/failure mode relationships)
  - Remove columns (remove test/failure mode relationships)

Figure 2. Top-level iterative loop identifying progression of repairs for DME framework

2.3. Diagnostic Modeling Evaluator (DME)

In the process of repairing D-matrix for better performance, an iterative loop (DME) consists of a sequential procedure with low-cost repairs considered first (as shown in Figure 2). The updates in the D-matrix are made in accordance to the new information from another source. After each iteration, the performance of the updated matrix is determined and the changes are accepted/declined accordingly based on mission directives. Here, the performance can be defined as combination of the required metrics, viz., false-positive, or false-negative rate, accuracy, time to diagnose, or cost involved for test and diagnoser implementation. Sometimes, improvement in one metric can affect others. Thus, balance should be maintained given the mission directives. This process is continued until there can be no further improvement. The necessary steps involved in the iterative updating DME procedure of the D-matrix are listed below:

2.3.1. Repair Cases for Updating D-matrix

1. Address row/column redundancy

Faults/test corresponding to rows and columns of D-matrix are assessed for redundancy in terms of two or more rows or columns having exactly the same signature. Duplicate rows or columns can result in ambiguous/masking faults and bad/duplicate tests, respectively (Simpson & Sheppard, 1992). In such a case, to decrease computational complexity and simplify representation, those faults/tests are grouped in to one. It is better to keep track of this change to avoid ignoring the subsequent repairs and also when they are required for mission critical functions or in other system mode.

2. Modify Fault Modeling (Change Rows of D-matrix)

In general, faults are modeled at component level. But, sometimes, components can be faulty with different severity levels based on their root-cause resulting in different fault signatures in D-matrix. In this case, new rows are added to the D-matrix when failure modes need to be refined. We add a row for each addition of a new component. Similarly, when the diagnosis is required at higher level (e.g Line Replaceable Unit (LRU) or Orbital Replaceable Unit (ORU)) or when components are removed, we remove the corresponding rows in D-matrix.

Table 1. List of tests[1]

Tests	Symbols
Voltage sensors	V1, V2, V3, V4
Battery temp. sensor	TB1, TB2, TB3, TB4
Battery current sensor	i
Position sensors	xFL, xFR, xBL, xBR
Velocity sensors	wFL, wFR, wBL, wBR
Current sensors	iFL, iFR, iBL, iBR
Temperature sensors	TFL, TFR, TBL, TBR

[1] FL, FR, BL, BR represents front left, front right, back left, back right wheels, respectively.

Table 2. D-matrix and the corresponding delays (in seconds) of each test for the rover system

	V1	TB1	V2	TB2	V3	TB3	V4	TB4	I	xBL	wBL	iBL	TBL
Motor Friction Fault BL	1/0.55	1/25.75	1/0.55	1/22.6	1/0.55	1/19.55	1/0.55	1/23.6	1/0.4	1/0.9	1/0.2	1/0.4	1/30.95
Parasitic Load	1/0.2	1/28.8	1/0.2	1/26.85	1/0.2	1/26.1	1/0.2	1/31.9	1/0.15	0	0	0	0
Voltage Sensor 1 Bias	1/0.15	0	0	0	0	0	0	0	0	0	0	0	0

3. Change Test Logic/Wrapper Code

We modify the test logic to attain better detection for each test. The test criteria, especially when defined by thresholds, may not hold during operation or with degradation of component's performance over time. This necessitates changing the logic subsequently either to trending or statistical tests. Additionally, refinement of abstraction of failure modes may in turn require test logic/wrapper code to be refined as well. Changes in test logic should be properly monitored, sometimes, for increased false positive or false negative detection rate with respect to the user defined expected performance measures for the D-matrix.

4. Repair False positives/negatives (Change the Entries of D-matrix)

The most important correction to the D-matrix is updating its entries, i.e., changing 1 to 0 or vice-versa, thus decreasing false positive or false negative isolation rate, correspondingly. This can be reflected as system (physical model) change. This means that a 1 in the D-matrix means two conditions are true: that a path exists between the fault and the test and that a set of signals propagate from the failure are detected at the test. A change in fault test relationships means that change for the paths and signals are applied. Note that both false positives and false negative isolation rates cannot be improved concurrently. This is because, to improve false negative isolation rate, we need to expand the threshold logic (red lines) which will increase false positives. Thus, the required acceptable metric is obtained from the mission directives.

5. Add/remove Tests (Change Columns of D-matrix)

Adding tests incurs additional cost, so we restrict this repair strategy to be done at last. We have to design new tests if some faults are not adequately detected or if they are not isolatable. Tests also can be broken into finer levels, similar to the component to failure mode representation, to be able to detect different fault modes with different severities. On the other hand, sometimes, low reliable and delayed tests hinder the overall diagnostic performance efficiency. Such tests when not detecting critical faults in any other system mode can be removed.

2.3.2. Diagnostic Algorithm

Here, any standard diagnostic procedure based on D-matrix can be applied as a diagnoser. We have applied diagnostic algorithms Dynamic Multiple Fault Diagnosis (DMFD) algorithms based on primal-dual optimization framework that can detect multiple, delay, and intermittent faults over time. Our problem is to determine the time evolution of fault states based on imperfect/perfect test outcomes observed over time and is formulated as one of finding the maximum *a posteriori* configuration to evaluate fault state evolution over time (which is why the time series is process in batch mode) that best explains the observed test outcome sequence. More details of these algorithms can be found in (Singh et al., 2009), (Kodali, Singh, & Pattipati, 2013), (Kodali, Pattipati, & Singh, 2013).

2.3.3. Performance Evaluation

Performance metrics can be overall diagnostic efficiency, false-positive rate, false-negative rate, diagnostic time and cost. The choice of metrics is dependent on the user-set criteria based on mission directives. As an example, user can determine to have less false positives (this may increase false negatives).

3. SIMULATIONS AND RESULTS

We demonstrate DME framework on two example systems, viz., rover and ADAPT systems. We do not provide the description for these systems. More details can be found in (Narasimhan, Balaban, Daigle, Roychoudhury, Sweet, Celaya, & Goebel, 2012) and (Poll, Patterson-Hine, Camisa, Garcia, Hall, Lee, Mengshoel, Neukom, Nishikawa, Ossenfort, Sweet, Yentus, Roychoudhury, Daigle, Biswas & Koutsoukos, 2007) for rover and ADAPT systems, respectively.

3.1. Example System 1: Rover

The initial D-matrix of the rover system is generated via simulations with three faults and thirteen tests (see Table 1 and Table 2). Three fault scenarios viz., battery parasitic load, motor friction fault, and voltage sensor fault are simulated (Narasimhan et al., 2012) by injecting them in the rover test bed and altering the corresponding measurements

Table 4. List of faults in ADAPT-Lite

No.	Fault ID	Fault modes	Test ID	Sensor description
1	ISH266	Stuck	E235	DC voltage
2	TE228	Stuck, offset	E240	DC voltage
3	IT267	Stuck, offset	E242	DC voltage
4	E267	Stuck, offset	E261	DC voltage
5	IT240	Stuck, offset	E265	AC voltage
6	ESH260A	Stuck	E267	AC voltage
7	E242	Stuck, offset	ESH244A	Actuator position
8	ESH275	Stuck	ESH260A	Actuator position
9	IT261	Stuck, offset	ESH275	Actuator position
10	E261	Stuck, offset	ISH236	Actuator position
11	E240	Stuck, offset	ISH262	Actuator position
12	E235	Stuck, offset	ISH266	Actuator position
13	E265	Stuck, offset	IT240	DC current transmitter (50A Max)
14	TE229	Stuck, offset	IT261	DC current transmitter (50A Max)
15	ISH262	Stuck	IT267	AC current transmitter (12A Max)
16	ST516	Stuck, Offset	ST265	AC frequency transmitter
17, 18, 19	FAN416	FailedOff, under speed, over speed	ST516	Speed (RPM) transmitter
20	EY275	Stuckopen, stuckclosed	TE228	Temperature
21	CB266	Tripped, failedopen	TE229	Temperature
22	INV2	FailedOff	XT267	Phase angle transducer
23	CB262	Tripped, failedopen		
24	EY260	Stuckopen, stuckclosed		
25	EY244	Stuckopen, stuckclosed		
26	CB236	Tripped, failedopen, stuckclosed		
27	ISH236	Stuck		
28	XT267	Stuck, offset		
29	ST265	Stuck, offset		
30	ESH244A	Stuck		

Table 3. Diagnosis time for each fault (fault injected at 50s)

	HyDe	QED	D-matrix diagnoser
Motor Friction Fault BL	50.2s	50.25s	50.2s
Parasitic Load	50.05s	51.2s	50.05s
Voltage Sensor 1 Bias	50.1s	50.3s	50.15s

with erroneous values. All these faults are injected at 50s. Temperature sensors have high detection delays (see Table 2), therefore, the corresponding diagnostic delays will also be longer. The DMFD algorithm with delays (Kodali et al., 2013) delivers intermediate diagnosis at each time-step with partial test information and updates it as the test information becomes available.

The inputs to DME framework are initial D-matrix and delay metrics in Table 2. Then, at iteration 1, the time to diagnosis is high due to detection delays of temperature sensors. There can be 2 repair actions for this case. Either, the test logic can be changed for the temperature sensors or they can be removed. But, changing test logic cannot avoid detection delays. Thus, the tests relating to temperature sensors are removed. This improves the time to diagnosis with out compromising the isolability for the listed fault universe[2]. The results with updated D-matrix are comparable with other diagnostic algorithms (HyDe, QED (Narasimhan et al., 2012) in Table 3 (the faults are injected at 50s).

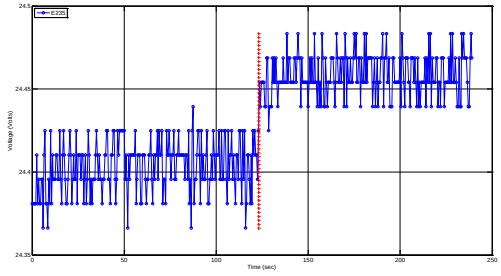

Figure 3. E235 sensor measurement when fault is injected

3.2. Example System 2: ADAPT-Lite

Here, we used the dataset generated from ADAPT-Lite system for DX workshop tier 1 competition 2009 (Kurtoglu, Narasimhan, Poll, Garcia, Kuhn, de Kleer, van Gemund, &

[2] This strategy may not hold if we expand the fault universe. Here, this is demonstrated as an example.

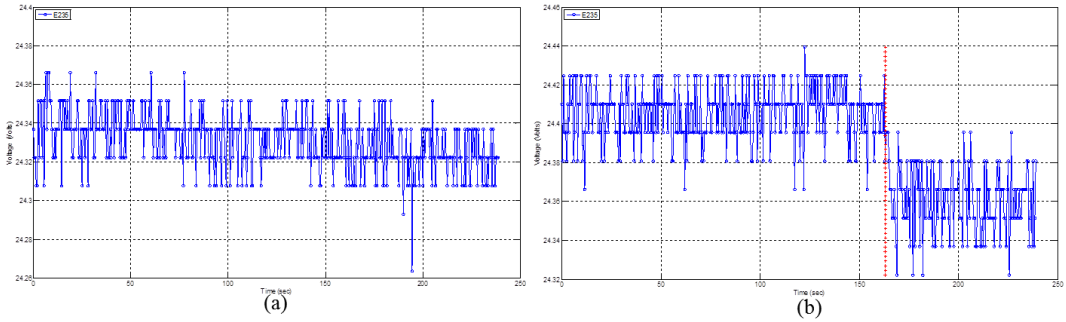

Figure 4. E235 sensor measurement: (a) nominal case (b) when fault 19 is injected (red line indicates the fault injection time)

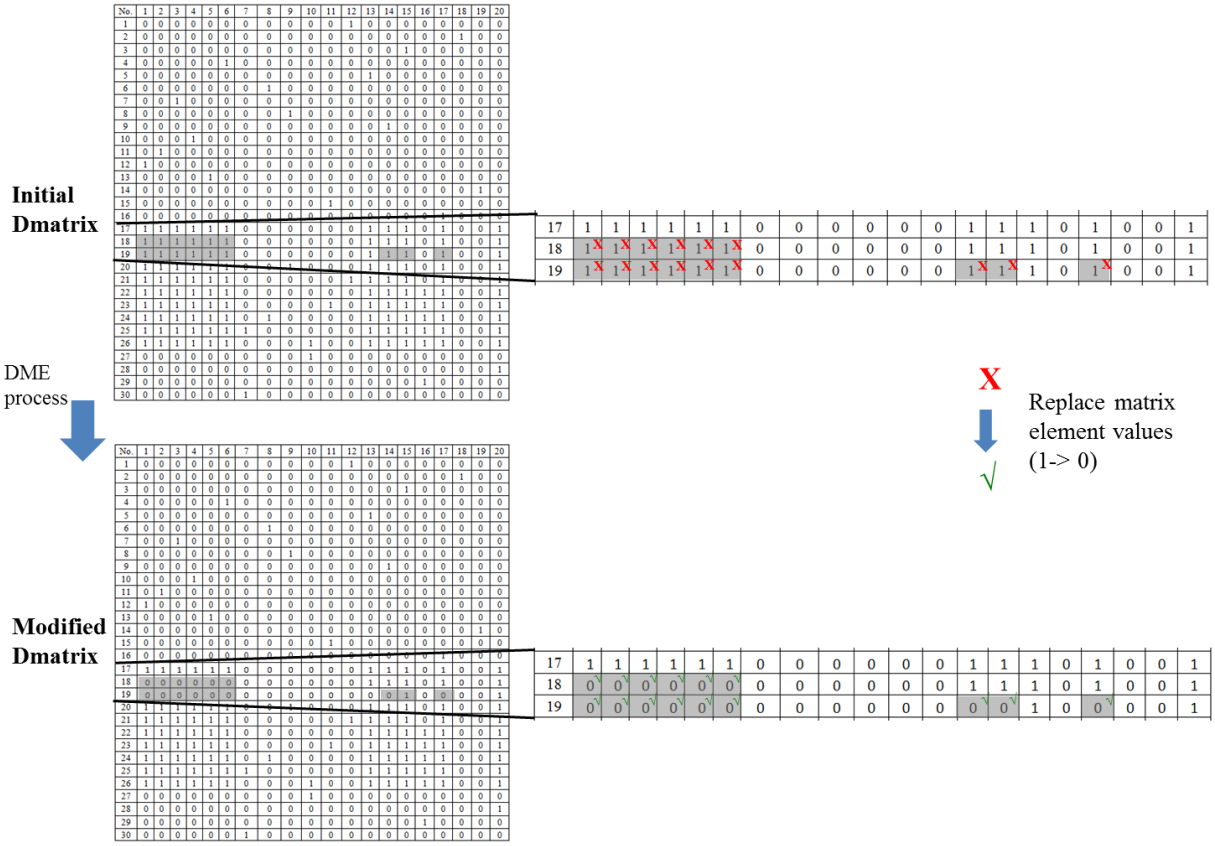

Figure 5. DME process to generate modified D-matrix from initial D-matrix (ADAPT system)

Feldman, 2009). This dataset has 2 parts: sample and competition data. Data (sensor measurements) is collected for 238.5s with a sampling rate of 0.5s, thus the data contains a total of 478 time steps for both nominal case and when faults are injected. Each fault is injected and the corresponding continuous sensor measurements over time are noted (see Table 4 for list of faults and tests). Initially, D-matrix is generated by visualizing the sensor measurements in sample data. As seen in Figure 3, there is a clear change in the mean value of the sensor E235 when fault E235 is injected. Then, the corresponding row-column entry in D-matrix is depicted as 1 in the D-matrix. The corresponding test logic in terms of threshold logic is also generated to suit fault detection.

In DME framework, at iteration 1 using sample data, rows 17-19 corresponding to different fault modes, viz., failed off, under speed, and over speed for the component FAN have similar fault signature. These are identified as ambiguous rows. Also, when faults 18 and 19 occur, they are misdiagnosed as fault 3 (IT 267 sensor fault). This is identified as either incorrect test logic or D-matrix entries in DME framework. But, most of the tests connected to faults

Table 5. Competition data results

(in %)	Initial D-matrix	Modified entries of D-matrix
False positive rate	4.85	6.94
False negative rate	2.39	2.41
Classification accuracy	94.57	96.15

18 and 19 didn't fail. Thus, the former repair is rejected. Therefore, the D-matrix entries are changed to generate new fault signatures for faults 18 and 19 (shown in Figure 5). The entries corresponding to voltage and current sensors are changed to 0. This is because, the effect of these faults on the corresponding sensor values is very low, i.e., the change in the measurement values is minimal. This is evident in Figure 4. Even though there is a clear indication of shift in the measurement value when fault is injected in Figure 4(b), those faulty values are overlapping with the values in the nominal case (24.3-24.36V). The repaired D-matrix is shown in Figure 5. These D-matrix repairs are verified on competition data using DMFD algorithm (Singh et al., 2009) (see Table 5). Classification accuracy is the percentage number of events that are correctly diagnosed (both nominal and faulty cases over 478 time-steps). Evidently, the diagnostic performance with modified D-matrix is better; thus, the corresponding repair strategy is accepted. Note that, the false positive and false negative rates are increased with modified D-matrix. But, here, the classification accuracy is considered as the decisive performance metric. There are no further changes in subsequent iterations.

4. Summary and Future Directions

Traditionally, diagnostics is viewed as an open-loop cascading process with the D-matrix as the input to the inference algorithm. In this context, DME is proposed to allow feedback from the diagnoser to the initial model via D-matrix repairs. Here, most importantly, the D-matrix can be updated to account for noise, lag and other effects by validating it through a time-series data stream or any other information that comes along during or after system development and operational deployment. Thus, in this iterative process, DME updates D-matrix and the corresponding test logic through a sequential procedure in the order of cost-effective repairs using the time-series data stream.

In our future research, DME framework will be updated and implemented as an automatic process. We will also experiment with systems that can accommodate the other repair strategies. We will also define user's role to validate the repair recommendation based on performance and mission directives and experiment with different metrics as the performance criteria. Most importantly, we will provide user-computer interface to communicate repair actions and provide necessary feedback. A systematic process to transverse the repairs back to the system model will also be investigated.

References

Luo, J., & Pattipati, K. (2007). An integrated diagnostic development process for automotive engine control systems. *IEEE Trans. Syst., Man, Cybern. C*, vol. 37, no. 6, pp. 1163–1173, Nov. 2007.

Singh, S., Kodali, A., Choi, K., Pattipati, K., Namburu, S., Chigusa, S., Prokhorov, D.V., & Qiao, L. (2009). Dynamic multiple fault diagnosis: Mathematical formulations and solution techniques. *IEEE Trans. Syst., Man, Cybern*. A, vol. 39, no. 1, pp. 160–176.

Singh, S., Holland, S., & Bandyopadhyay, P. (2011). Trends in the development of system-level fault dependency matrices. *IEEE Aerospace Conference*, Big Sky, Montana.

Luo, J., Tu, H., Pattipati, K., Qiao, L., & Chigusa, S. (2006). Graphical models for diagnostic knowledge representation and inference. *IEEE Instrum. Meas. Mag.*, vol. 9, no. 4, pp. 45–52.

Qualtech Systems Inc., www.teamqsi.com.

Simpson, W., & Sheppard, J. (1992). System Testability Assessment for Integrated Diagnostics. *IEEE Des. Test. Comput.*, vol. 9, no. 1, pp. 40-54.

Kodali, A., Singh, S., & Pattipati, K. (2013). Dynamic set-covering for real-time multiple fault diagnosis with delayed test outcomes. *IEEE Trans. Syst., Man, Cyben. A*, vol. 43, no. 3, pp. 547-562.

Kodali, A., Pattipati, K., & Singh, S. (2013). Coupled factorial hidden Markov models (CFHMM) for diagnosing multiple and coupled faults. *IEEE Trans. Syst., Man, Cyben. A*, vol. 43, no. 3, pp. 522-534.

Narasimhan, S., Balaban, E., Daigle, M., Roychoudhury, I., Sweet, A., Celaya, J., & Goebel, K. (2012) Autonomous decision making for planetary rovers using diagnostic and prognostic information. *Proceedings of the 8th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes (SAFEPROCESS 2012)*, Mexico City, Mexico.

Poll, S., Patterson-Hine, A., Camisa, J., Garcia, D., Hall, D., Lee, C., Mengshoel, O., Neukom, C., Nishikawa, D., Ossenfort, J., Sweet, A., Yentus, S., Roychoudhury, I., Daigle, M., Biswas, G., & Koutsoukos, X. (2007). Advanced diagnostics and prognostics testbed. *In Proc. DX'07*, pp. 178–185.

Kurtoglu, T., Narasimhan, S., Poll, S., A., Garcia, D., Kuhn, L., de Kleer, J., van Gemund, A., & Feldman, A. (2009). First international diagnostics competition – DXC'09. *In Proc. DX'09*.