Model-Based Assurance of Diagnostic Procedures for Complex Systems

Tolga Kurtoglu[1], Robyn Lutz[2], and Martin S. Feather[3]

[1] *Mission Critical Technologies @ NASA Ames Research Center, Moffett Field, CA, 94035, USA*
tolga.kurtoglu@nasa.gov

[2] *Jet Propulsion Laboratory/Caltech, Pasadena, CA, 91109, and Iowa State University, USA*
robyn.r.lutz@jpl.nasa.gov

[3] *Jet Propulsion Laboratory, California Institute of Technology, CA, 91109, USA*
martin.s.feather@jpl.nasa.gov

ABSTRACT

Verifying diagnostic procedures for complex systems is hard and labor-intensive. Usually this verification is accomplished primarily through extensive review of the procedures by experts. We aim to augment this review process by using insights from comparing the diagnostic steps described in the procedural definitions with diagnostics information derived from existing models of the system. These comparisons offer various conformance checks between the manually developed diagnostic procedures and the diagnostic trees auto-generated from the diagnostic system models. We previously described our DTV (Diagnostic Tree for Verification) technique based on these comparisons. This paper describes an extension to DTV, and reports results of an application of DTV to a representative system's diagnostic procedures. Specifically, it outlines four analyses (branch analysis, root cause coverage, path verification, and efficiency) that can be performed using DTV; illustrates the process for applying DTV; and reports results from our application of DTV to assure fifteen of the procedures developed for diagnosing problems in an electrical power system testbed for spacecraft.

1 INTRODUCTION

The operation of complex engineered systems requires the development of diagnostic procedures. These provide a detailed set of instructions to help operators monitor the system's parameters and respond to potential problems by detecting and identifying faults, and guides them in performing system reconfiguration or restoration.

These procedural definitions include a complicated mix of software checks and calibrations, conditional commands, manual inputs, checks of console data, and inspection of physical equipment. The crew of the Space Shuttle, for example, relies on a collection of procedural definitions and checklists in order to interpret any potential anomalies, figure out the root cause of problems, and work on mitigating the root-cause failure (Hayashi et al., 2008). As a result, crew safety and mission success become highly dependent on the correctness of the diagnostic procedures.

It is therefore imperative that these procedures are verified before being used. However, verifying diagnostic procedures for complex systems is hard and labor-intensive. Usually this verification is heavily dependent on extensive review of the procedures by experts.

In this research, we aim to augment this review process by using insights from comparing the diagnostic steps described in the procedural definitions with diagnostics information derived from existing models of the system. These comparisons offer various conformance checks between the manually developed diagnostic procedures and the diagnostic system models.

Checking conformance in this way has two advantages. First, the models offer an independent perspective distinct from expert review. Second, there exists computer software that is able to systematically explore the diagnostic implications of a model, a task that can be quite intricate as systems grow in size and complexity. Both these advantages increase the likelihood of revealing errors (if present) in the diagnostic procedures. This approach can therefore contribute to assuring the correctness of diagnostic procedures for complex systems. Furthermore, this approach can also identify opportunities for improving the efficiency of diagnostic procedures by revealing the presence of redundant steps in the existing procedures, and/or by offering alternately structured procedures that are capable of arriving at the same diagnostic conclusions but in fewer steps.

In previous work we introduced the DTV (Diagnostic Tree for Verification) technique (Kurtoglu et al., 2009). The key idea of DTV is to compare the text-based procedures for diagnosing faults during a system's operations with the diagnostic trees auto-generated from a model of the system. This paper describes our investigation of, and an extension to, the DTV technique. The extension addresses the challenge of assuring the correctness of a text-based procedure in the (frequent) case when the auto-generated diagnostic tree uses different sequences of tests to arrive at its diagnostic conclusions. In such cases, if both trees are able to diagnose to the same sets of root causes, this offers some assurance, but does not guarantee, that the paths followed in the text-based procedure in fact imply their diagnostic conclusions. In our extension, we use the steps in a path of the text-based procedure to "drive" the system model so that once the end of the path is reached we can compare the path's diagnosis with the model's conclusion.

The specific contributions of this paper over our previous work are: (1) describes four analyses (branch analysis, root cause coverage, path verification, and efficiency) that can be performed using DTV; (2) describes and illustrates the process for applying DTV; and (3) reports results from our application of DTV to assure fifteen of the procedures developed for diagnosing problems in an electrical power system testbed for spacecraft.

In what follows, we first present a review of related work in verification of operational procedures. Section 3 describes the Diagnostic Tree for Verification method. Section 4 introduces the Electrical Power System Testbed in the ADAPT Lab at NASA Ames Research Center that is used as a case example in this study. Section 5 discusses the application of the DTV method to the verification of a specific set of procedures developed for the electrical power system. Section 6 describes the four analyses and presents the results. Finally, Section 7 presents concluding remarks and an outlook for future work.

2 RELATED WORK

Model-based reasoning methods utilize a wide variety of engineering models as the foundation for representing diagnostic knowledge and developing algorithms that use this knowledge for fault detection and isolation. In parallel developments, different communities have found value in analytic state-based models, input-output transfer function models, fault propagation models, and quantitative physics-based models to develop online automated diagnostic software for monitoring and diagnosis of dynamical systems (Patterson-Hine et al., 2005). In this work, we seek to leverage information used in building diagnostic models in order to check and verify diagnostic procedures.

There are various techniques developed to help verify and validate procedures. Most current verification techniques are largely manual (inspection and reviews) and focus primarily on the conformance of command programs – scripts written for execution of procedures - to procedure definitions. Some advanced procedure authoring tools such as PRIDE (Kortenkamp et al., 2008) support syntax checking and can enforce syntax constraints.

Automated approaches to verification enable the verification of syntactic and semantic well-formedness properties of procedure scripts, and simulation capabilities enable the validation of equivalence and correctness relations between different system representations (Brat et al., 2008). Among these, static analyzers verify that procedures are syntactically and semantically well written and structured. They are used to check for missing variable declarations, divide-by-zero, null pointer dereferences, out-of-bounds array references, incorrect ordering of procedure calls, etc. An example of a static analysis tool is the commercially available Polyspace C-verifier (Polyspace, 2008). Static analysis, however, cannot catch all possible problems with procedures. An alternative verification method, model checking, allows for the systematic exploration of the entire state space of a system, checking that user-specified properties hold of all possible behaviors of a system. As a result, model checkers can find errors in models like deadlocks or race conditions, and can verify whether a procedure is guaranteed to terminate with the system left in a desired state. Examples of model checkers used in space applications include LTSA (LTSA 2008), and Java Path Finder (Visser et al., 2003). (Verma et al., 2008) discusses use of Java Path Finder to validate a simulated lunar robotic site survey operations plan.

3 DIAGNOSTIC TREE FOR VERIFICATION (DTV) METHOD

The DTV method seeks to exploit knowledge and automated analysis techniques applied for the diagnostic process by model-based diagnosis systems. These tools utilize information from the design phase, such as safety and mission assurance analysis, failure modes and effects analysis (FMEA), fault propagation models and testability analysis, and employ topological and analytical models of the nominal and faulty operations of a system for fault diagnosis, isolation, and recovery (Patterson-Hine et al., 2005). This information

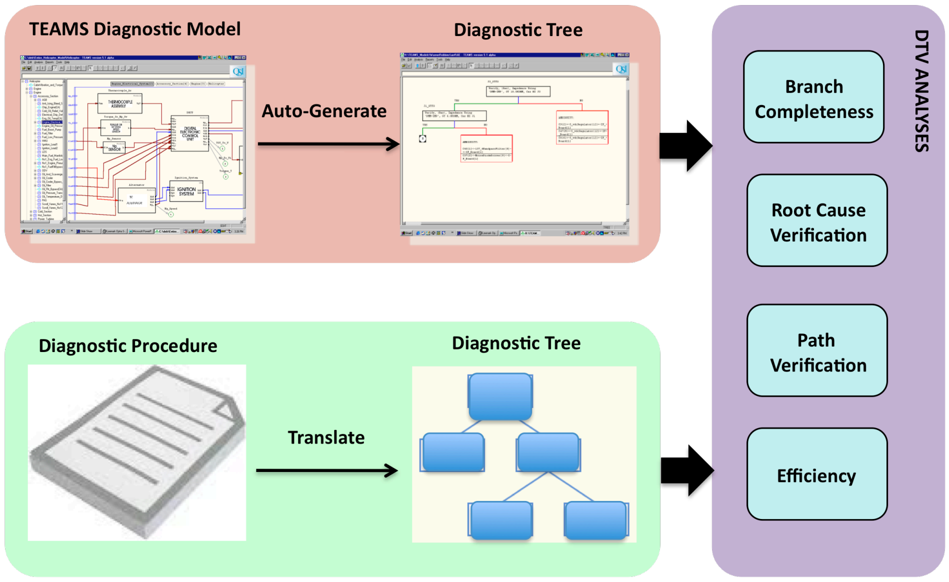

Figure 1. Overview of the developed DTV Method

provides an independent perspective that the DTV method uses both to verify the correctness and consistency of diagnostic procedures, and also to identify areas where the procedures themselves can be improved upon.

The primary goal of the DTV method is to enable analysts and procedure developers to statically and dynamically explore the system model using a set of auto-generated diagnostic trees in order to gain insight into the efficacy and efficiency of alternative fault diagnosis and isolation paths. By comparing the sequence of steps described in text-based procedures with the paths of auto-generated diagnostic trees, they can find relationships between the two representations (diagnostic model and text-based procedures) as illustrated in Figure 1.

The basis for this comparison is a five-step process. The steps to apply the DTV analysis for a selected system and a set of operational procedures are as follows:

(1) The system's failure effects (how failures will propagate through the system, and the observable conditions those failures will manifest) are formally modeled.

(2) A set of procedures is selected for DTV analysis.

(3) Conditions (or observations) that trigger the selected procedures are documented and added to the model as "symptoms".

(4) A set of diagnostic trees is auto-generated from the model for each "symptom" point in the model.

(5) Analyses are conducted by comparing the elements and steps in the text-based procedural definitions with those in the diagnosis trees generated from the model.

This comparison is currently performed manually. However, we are working towards developing a representation that would capture the information contained within the text-based procedures and diagnostic trees using a common language, such that this comparative analysis can be done in an automated fashion. In what follows, we describe each step of the DTV method in detail.

3.1 Step 1: Build a System Model

The Testability Engineering and Maintenance System (TEAMS) tool suite (QSI, 2009) is the primary platform used for modeling by the DTV method. TEAMS is built upon the multi-signal modeling formalism (Deb et al., 1995), which is a hierarchical modeling methodology where the propagation paths of the effects of a failure are captured using directed graphs. The model is based on structural connectivity or a conceptual block diagram of a physical system connected by links or paths. Software modules interfacing with the system are treated like any other hardware component, and can be included in the model. Functions describe attributes of system variables to be traced. The TEAMS modeling elements called test points are then added to the model. Test points represent the physical or computational locations of checks using sensors or sensor data as well as other means for observing a system. Tests are checks that look at the data from the sensors and make decisions about system attributes associated with those measurements. This graph topology is then converted into a matrix representation describing the relationship between faults and test points for a given mode of the system. This representation contains the basic information needed to interpret test results and diagnose failures during operations.

3.2 Step 2: Select/Develop a Set of Procedures

A procedure is a detailed set of instructions specifying how a piece of equipment is operated, or a task is to be performed (Frank, 2008). In this research we are focusing on a specific class of procedures, namely those developed for troubleshooting anomalies in a system. Their purpose is to guide operators on how to diagnose potential faults and identify their root causes. The operational steps in diagnostic procedures include checks/tests/verification of the physical and software parameters of the system, commands to and from the system, and manual and automated actions for recovery.

3.3 Step 3: Add "Symptoms" to the Model

A "symptom" in the TEAMS model is a condition, observation, or an observable state that indicates whether or not a system is performing its intended function. For example, for an electrical power system the observation of "low voltage level at battery output" constitutes a symptom. For the purposes of the DTV method, symptoms are used as starting points to perform subsequent troubleshooting and diagnosis analyses. Therefore, symptom points corresponding to the selected set of procedures in Step 2 are added to the TEAMS model to facilitate comparative analysis of the procedures.

3.4 Step 4: Auto-generate Diagnostic Trees for the Selected Set of "Symptoms"

A diagnostic tree describes a branching sequence of checks/tests used for troubleshooting an anomaly in the system. Different operational modes or configurations have different diagnostic trees since some faults are only possible, and some checks are only appropriate or available, in certain configurations or modes of operation. All the branch points of the diagnostic tree are tests in the TEAMS model. Each of the leaf nodes of the diagnostic tree is a candidate root cause explaining the symptom, while sets of faults form so-called "ambiguity groups". Each element of an ambiguity group is a possible root cause of the symptom, but isolation to which specific element of that set is the actual cause is not determinable in the model.

3.5 Step 5: Compare Procedure Definitions and Diagnostic Trees

In this step, the sequence of steps described in the text-based diagnostic procedure definitions is compared to the diagnostic paths of the auto-generated diagnostic trees. In particular, four different analyses (branch completeness, root cause coverage, path verification, and efficiency) are performed to assure and verify the diagnostic procedures. These analyses are further detailed in Section 6.

4 CASE EXAMPLE: ADAPT ELECTRICAL POWER SYSTEM TESTBED

In this section, we provide a brief overview of the electrical power system used for DTV analysis and describe its major elements.

The Advanced Diagnostics and Prognostics Testbed (ADAPT) at the NASA Ames Research Center is a unique facility designed to test, measure, evaluate, and mature diagnostic and prognostic health management technologies. Reflecting the importance of electrical power systems (EPS) in aerospace (Button and Chicatelli, 2005; Poll et al., 2007), ADAPT provides a representative aerospace vehicle EPS that enables automated diagnosis in a complex domain. A simplified version of main functions and layout of the ADAPT electrical power system is shown in Figure 2. The EPS can deliver power to various loads, which in an aerospace vehicle would include subsystems such as the avionics, propulsion, life support, and thermal management systems.

ADAPT EPS contains elements common to many aerospace applications: power storage and power distribution. In the simplified version used in this study, the power storage consists of two battery modules. Either of the two batteries can be used to power either of the two load banks in the power distribution element. This design gives the ADAPT EPS basic redundancy and reconfiguration capability. Electromechanical relays are used to route the power from the sources to the batteries and from the batteries to the loads. An inverter converts the DC battery input to AC output. Circuit breakers are located at various points in the distribution network to prevent overcurrents from causing unintended damage to the system components.

A data acquisition and control system sends commands to, and receives data from the EPS. Testbed operator stations are integrated into a software architecture that allows for nominal and faulty operations of the EPS, and includes a system for logging all relevant data. The instrumentation allows for monitoring of voltages, currents, temperatures, switch positions, light intensities, and AC frequencies, and includes over 100 sensors. (More information on

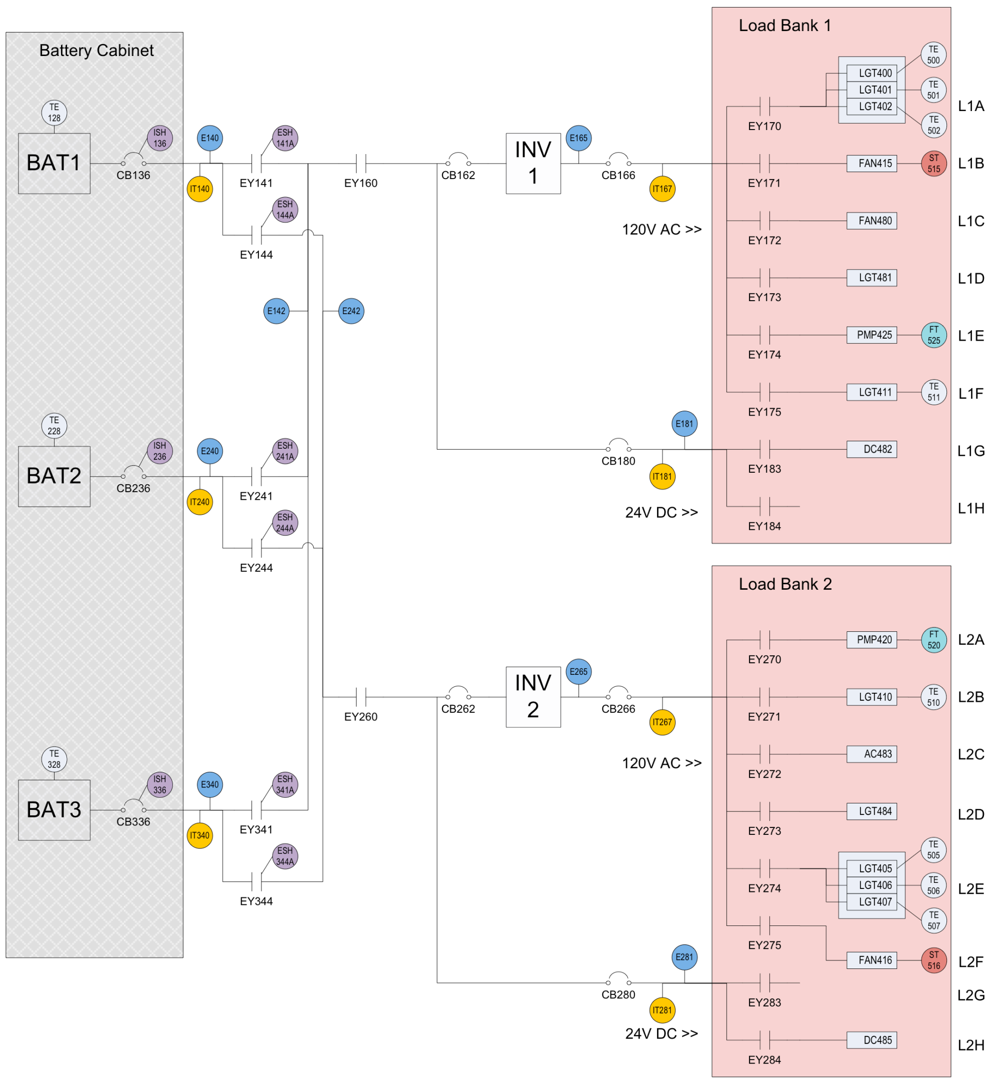

Figure 2. The schematic of the Electrical Power System (EPS)
in the Advanced Diagnostics and Prognostics Testbed (ADAPT) Laboratory

the ADAPT EPS testbed can be found in (Poll et al., 2007)).

5 APPLICATION OF THE DTV METHOD TO THE ADAPT EPS SYSTEM

This section describes our work to date to apply the diagnostic tree for verification (DTV) method and its modeling, analysis and implementation steps outlined in Section 3 to the electrical power system testbed.

5.1 Step 1: Building the ADAPT EPS System Model

A detailed model of the ADAPT EPS Testbed has been previously developed at NASA Ames. Figure 3 shows the high-level structure of the system model in TEAMS. The model incorporated the hierarchical structure of the EPS system, as well as all the relevant components and sensors. The model also captured failure modes of individual components and tests that are used for fault detection and isolation.

5.2 Step 2: Selecting ADAPT EPS Procedures

The operational procedures for the EPS system were modified from an advanced caution and warning system developed as an interface concept for a crewed vehicle (McCann et al., 2006). Table 1 shows the list of 15 unique procedures that were used in our study of the application of the DTV method to the EPS system.

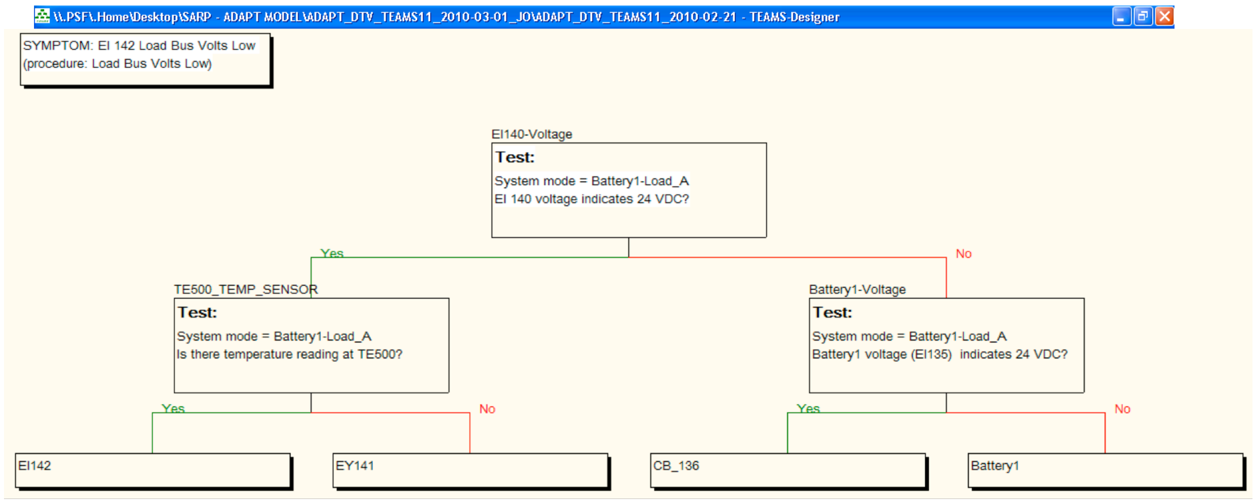
\\.PSF\.Home\Desktop\SARP - ADAPT MODEL\ADAPT_DTV_TEAMS11_2010-03-01_JO\ADAPT_DTV_TEAMS11_2010-02-21 - TEAMS-Designer
SYMPTOM: EI 142 Load Bus Volts Low (procedure: Load Bus Volts Low)
EI140-Voltage
Test:
System mode = Battery1-Load_A
EI 140 voltage indicates 24 VDC?
Yes
No
TE500_TEMP_SENSOR
Test:
System mode = Battery1-Load_A
Is there temperature reading at TE500?
Battery1-Voltage
Test:
System mode = Battery1-Load_A
Battery1 voltage (EI135) indicates 24 VDC?
Yes
No
Yes
No
EI142
EY141
CB_136
Battery1

Figure 4. An auto-generated diagnostic tree for a voltage anomaly indicated at sensor EI142 (E142 in Fig. 2)

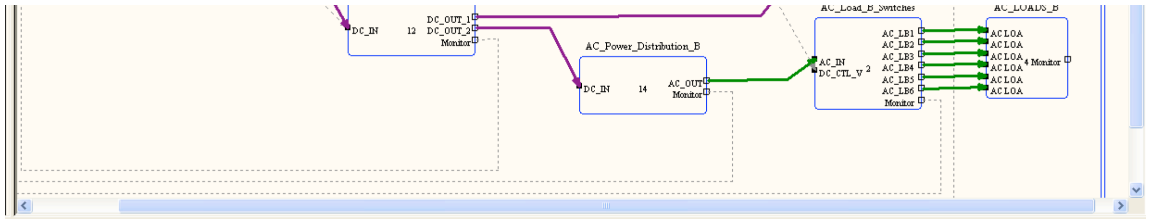

Figure 3. High-level screenshot of the ADAPT model developed

Table 1. The list of the 15 procedure definitions (and corresponding symptom points in the ADAPT model)

Procedure
SYMPTOM: ST 515 Fan (L1B) Low Rpm ((procedure: Fan Load Low RPM)
SYMPTOM: FT 525 Pump (L1E) Low Flow (procedure: Pump Load Low Flow)
SYMPTOM: LT 500 Light (L1A) Low Light Level (procedure: Light Load Low Light Level)
SYMPTOM: DC 482 Load (L1G) Off (procedure: DC Load Off)
SYMPTOM: EI 167 AC Bus V2 Volts Low (procedure: AC Bus V2 Volts Low)
SYMPTOM: EI 181 DC Bus Volts Low (procedure: DC Bus Volts Low)
SYMPTOM: EI 142 Load Bus Volts Low (procedure: Load Bus Volts Low)
SYMPTOM: EY 183 DC Bus Load Relay Open (procedure: DC Bus Load Relay Open)
SYMPTOM: EI 165 AC Bus V1 Volts Low (procedure: AC Bus V1 Volts Low)
SYMPTOM: EI 135 Battery Volts Low (procedure: Battery Volts Low)
SYMPTOM: EI 140 Battery Out Volts Low (procedure: Battery Out Volts Low)
SYMPTOM: EY 160 Load Bus Relay Open (procedure: Load Bus Relay Open)
SYMPTOM: EY 141 Distribution Relay Open (procedure: Distibution Relay Open)
SYMPTOM: EY 170 AC Bus Load Relay Open (procedure: AC Bus Load Relay Open)
SYMPTOM: EI 161 Load Volts Low (procedure: Load Volts Low)

5.3 Step 3: Adding Symptoms to the ADAPT EPS TEAMS Model

In running the DTV analyses, we have identified the symptoms (i.e., observable states) of the EPS system that correspond to anomalous conditions. We then added 15 symptoms to the TEAMS model for the procedures listed in Table 1. The table also shows where each symptom is implemented in the model. For example, the “Battery Out Volts Low” symptom is implemented in component EI140.

5.4 Step 4: Auto-Generating Diagnostic Trees

In this step, we set the previously defined 15 symptoms to be active in the TEAMS model one at a time. This automatically generates a diagnostic tree in TEAMS by forcing the symptom to be the root node of the tree. As described before, the diagnostic tree captures a sequence of tests/checks that, if performed, will diagnose/isolate the fault that is causing the analyzed symptom (such as the one shown in Figure 4).

5.5 Step 5: Comparing Procedural Definitions and Diagnostic Trees

As a final step, we conducted four analyses by comparing different characteristics of the 15 manual procedures listed in Table 1 with those 15 diagnostic trees auto-generated from the TEAMS EPS model. The next section describes the application of each of the four analyses in detail.

6 ANALYSES USING THE DTV METHOD

The four analyses conducted to assure and verify the diagnostic procedures were: branch analysis, root cause coverage, path verification, and efficiency. Each is described below, together with the results of the analysis and the possible implications of these results for application of the DTV technique on other systems.

The analyses used the diagnostic tree auto-generated by TEAMS to compare with the steps described in the text-based procedure. To facilitate the comparison, we first manually converted the steps in each procedure to a Visio tree representation similar in style to the look of the trees that TEAMS auto-generates. This conversion was straightforward for our fifteen procedures, but we are also working toward an automated translation to use with systems containing a large number of procedures. The next step was to do a tree-to-tree comparison of the Visio drawings to the diagnostic trees generated from the TEAMS model. As described below, the DTV analyses gave an independent perspective on the procedures' branch completeness, root cause coverage, path correctness, and efficiency.

6.1 Branch Completeness Analysis

Diagnostic, operational procedures lead one through a sequence of checks where each check's result is either passed or failed. In the decision tree representation of the procedure, this means that each check should have two outgoing branches, each leading to a successor node (either another check, or a diagnosis). The branches indicate the next action to be taken based on the result of the check.

The branch completeness analysis of the procedures applies three criteria:

- that every non-leaf node in the textual procedure's diagnostic tree has two outgoing branches, representing both a passed and a failed check, each branch leading to another node which if it is a leaf node must be a diagnosis, or if not, another check,
- that any check in a path from root node to leaf node is not duplicated in that path, nor its negation is also included in that path, and
- that both outgoing branches from a non-leaf node in the procedure's diagnostic tree have the same successor node in the TEAMS generated diagnostic tree.

The first two criteria do not require the equivalent diagnostic tree to have yet been generated by TEAMS. Applying the third criteria does require the equivalent diagnostic tree to have been generated by TEAMS.

Branch completeness. Figure 5 shows an example of the first criteria. The first diagnostic check in the procedure, whether the voltage sensor EI 167 returns a value, has only one branch, regardless of the result of the diagnostic check. This could indicate that either this check or the subsequent check (whether there is voltage at EI 181) is superfluous. It could also be that both checks are needed but that the procedure is missing a branch, or that our translation of the procedure into a diagnostic tree was flawed. In this case, the procedure is used to diagnose the likely cause when the symptom is that Load Bus Voltage is Low. The diagnostic procedure redundantly checks both the downstream AC and DC voltage sensors (although the TEAMS generated diagnostic tree does not). By examination of the diagnostic procedures' trees, we found a total of four diagnostic procedures with two sequential checks tied to a single yes-no outcome.

Test uniqueness. An example from applying the second criteria is a procedure which begins with the check "Is CB 166 closed?" and then has the negation of this, "Is CB 166 open?" farther down in the path. Given the assumption that the state of the system does not change during the diagnosis, this check is redundant.

Branch successors. With regard to the third criteria, an example is two cases in which the successor branches to a check in the diagnostic procedures were different from the successor branches to that same check in the TEAMS generated diagnostic trees. These turned out to be caused by a naming inconsistency between the procedures and the TEAMS model. In both cases the diagnostic procedure described a specific load ("Fan 415", "Pump 425"), while the model described the load in terms of its location ("AC_Load_L1B", "AC_Load_L1E"). Such naming inconsistencies have been time-consuming across the project and have led the project to improved standardization in naming model components and to rigorous maintenance of name-mapping tables. More generally, to our surprise, the TEAMS-generated diagnostic tree was not the same as the textual procedure's diagnostic tree for any of the fifteen symptoms we investigated. This is because there are many test points along the path by which current flows from sources to the loads in EPS, and the TEAMS generated diagnostic trees typically choose a different test point as the first step compared to the text-based procedures.

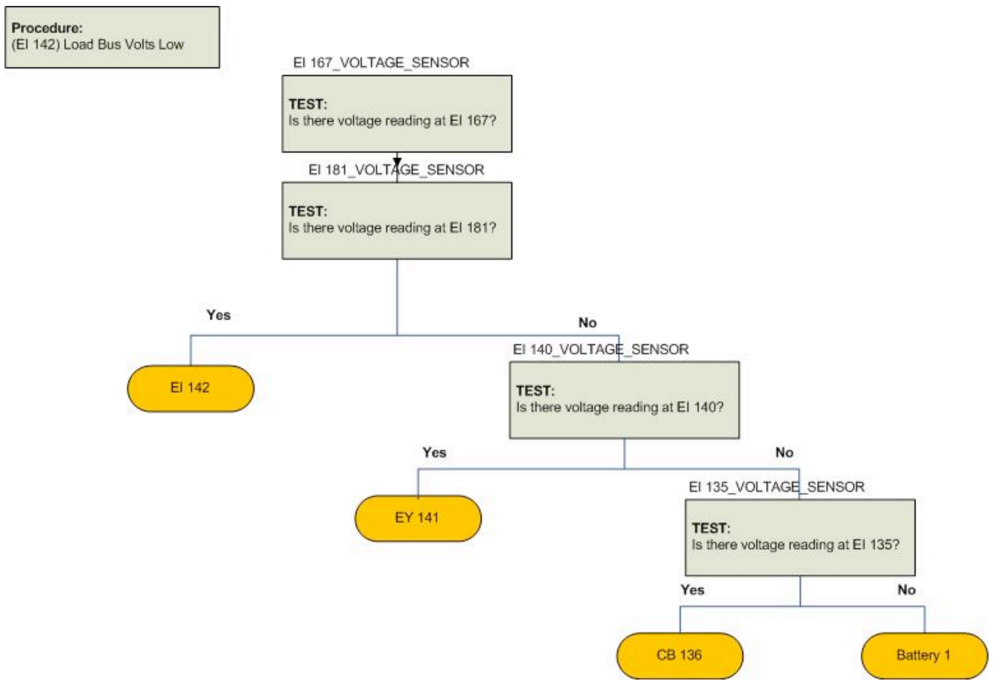
Procedure:
(EI 142) Load Bus Volts Low
EI 167_VOLTAGE_SENSOR
TEST:
Is there voltage reading at EI 167?
EI 181_VOLTAGE_SENSOR
TEST:
Is there voltage reading at EI 181?
Yes
No
EI 142
EI 140_VOLTAGE_SENSOR
TEST:
Is there voltage reading at EI 140?
Yes
No
EY 141
EI 135_VOLTAGE_SENSOR
TEST:
Is there voltage reading at EI 135?
Yes
No
CB 136
Battery 1

Figure 5. A translated tree representation of a diagnostic procedure for a voltage anomaly indicated at sensor EI142

6.2 Root Cause Coverage Analysis

A root cause of a symptom is a component whose failure would cause that symptom to be exhibited. Root Cause Coverage Analysis compares two diagnostic trees to see whether they are able to isolate failures to the same sets of possible root causes.

Each of the leaf nodes of a symptom's diagnostic tree should identify either a single component whose failure would cause that symptom, or a set of several components (an "ambiguity group") the failure of any single one of which would cause that symptom. Thus:

- Every component identified in a leaf node of a symptom's diagnostic tree should be a root cause of that symptom.
- The union of all of a symptom's diagnostic tree's leaf node components should include all the possible root causes of the symptom.
- A leaf node should consist of a set of several components (an "ambiguity group") only if it is not possible to disambiguate further among the components in that set.

In the DTV approach, these criteria are assessed by comparing the leaf nodes of the text-based procedure's diagnostic tree for a symptom against the leaf nodes of a diagnostic tree generated by TEAMS to diagnose that same symptom. Note that only the leaf nodes of the two trees are compared, not the structure of the trees themselves – in the later subsections we will consider the tree structures.

First, we form the union of the symptom's procedural diagnostic tree's leaf node components; we form the union of the TEAMS generated diagnostic tree's leaf node components; we compare the two sets so formed.

- If the two sets are equal, then the comparison has not revealed any problems (this does not prove there are no problems – it could be that both the procedure and the TEAMS model have the same flaw(s)).
- If a component in the set formed from the procedure's diagnostic tree is not in the set formed from the TEAMS generated diagnostic tree, then either the TEAMS model is incorrect, or that component is *not* a root cause of the symptom – i.e., we have identified an instance of *incorrectness* in the procedure.
- If a component in the set formed from the TEAMS generated diagnostic tree is not in the set formed from the procedure's diagnostic tree, then either the TEAMS model is incorrect, or that component is a root cause of the symptom missing from the procedure's diagnostic tree – i.e., we have identified an instance of *incompleteness* in the procedure.

We can also compare the two diagnostic trees' ability to isolate failures by examining their leaf nodes consisting of sets of components – the "ambiguity groups." If one tree contains a leaf node comprising a set of components, but that set is not a leaf node in the other tree, instead some of those components are individual leaf nodes, or are in smaller ambiguity groups, then the second tree is demonstrating a more refined isolation than the first.

As illustration, consider the symptom Load Bus Volts Low (EI 142). The procedure's diagnostic tree is shown in Figure 5, and the TEAMS-generated diagnostic tree is shown in Figure 4. As can be seen, both diagnostic trees have the same set of leaf nodes, namely {EI 142, EY 141, CB 136 and Battery 1}, despite the two trees having different overall structures and containing different checks.

Overall, we found that for each symptom, the set of leaf nodes of the procedure's diagnostic tree for that symptom was equal to the set of leaf nodes of the TEAMS-generated diagnostic tree for that same symptom. However, for several symptoms the TEAMS-generated diagnostic tree contained a leaf node consisting of several components (i.e., an "ambiguity group"), while the procedure's diagnostic tree for that same symptom contained only leaf nodes consisting of single components. These discrepancies suggested that the procedure's diagnostic trees were capable of isolating each symptom to unique root causes, yet the same was not true in the TEAMS model. It turned out that the TEAMS model was lacking some of the tests equivalent to checks utilized by the procedures in isolating faults.

We also observed that for a few symptoms, the procedure's diagnostic tree contained a check with one of its two outcome branches terminating without indicating any leaf node whatsoever! When we examined these cases we were able to determine that for the check result to take the abruptly terminating outcome branch, it would require more than one component to have failed. Thus given our assumption of diagnosis in the context of only single component failures, that abruptly terminating branch of the check could never be taken.

6.3 Path Verification Analysis

A path within a symptom's diagnostic tree is a sequence that starts at the root node of the tree, descends through the tree taking one of the outcome branches of each check it encounters on the way, and terminates at a leaf node. It indicates that given the symptom and that sequence of check outcomes, the diagnosis is the root cause(s) listed in the leaf node. Path verification analysis aims to verify whether the path's diagnosis is correct. Note that it is possible for a diagnostic tree to satisfy root cause coverage analysis when compared to another (correct) diagnosis tree, and yet contain incorrect path diagnoses if the "pass" branch of a check was assigned the outcome of the "fail" branch, and vice-versa.

In general, path verification analysis cannot be achieved by comparing the procedure's diagnostic tree with the TEAMS-generated diagnostic tree for the same symptom. This is because the respective paths in the two trees leading to the same diagnosis may make use of different sequences of checks. This was the case for all 15 of the procedures we examined. The DTV approach instead uses another capability of the TEAMS tool, to indicate what can be deduced about the state of components given a set of check outcomes. To do this, we take a path from the procedure's diagnostic tree for a symptom, and input to TEAMS the symptom itself (as a "failed" test in TEAMS parlance), and each of the check outcomes along the path (each as a "passed" or "failed" test as appropriate). We then compare the procedure's path's diagnosis with the TEAMS-calculated status of components. The two agree if they indicate the same set of components as possible root causes.

Using this approach we were able to verify six of the procedure diagnostic trees completely, i.e., every path in the diagnostic tree. For several more of the diagnostic trees we could only perform verification on *some* of their paths. The paths we could not perform verification on were those that made use of checks for which the equivalent test had not been modeled in the TEAMS model. Finally, for a few of the diagnostic trees the very first check in the tree did not have an equivalent in the TEAMS model, so verification could not be performed on *any* of their paths.

For all the paths on which we were able to perform verification, the TEAMS result confirmed the procedure's diagnosis. Again, this does not prove correctness, since it could be that both the procedure and the TEAMS model have the same flaw(s).

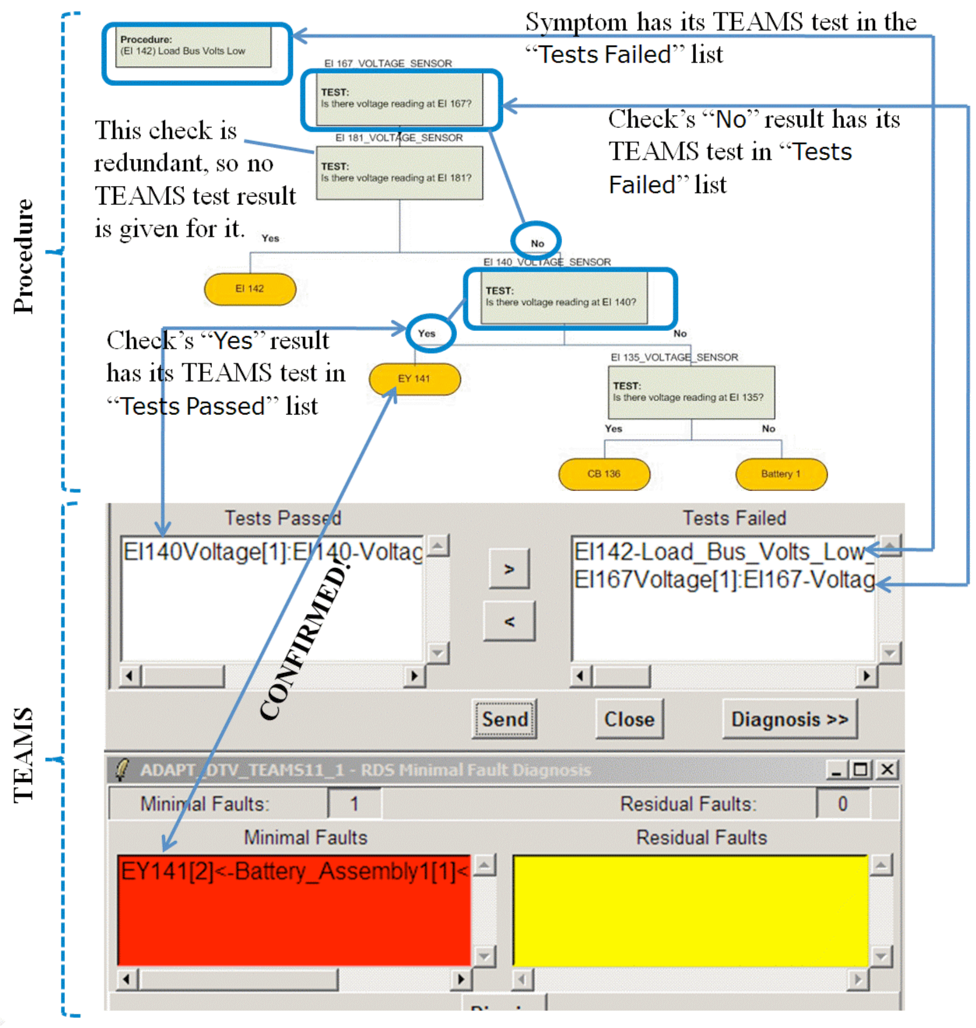
Procedure
TEAMS
Procedure: (EI 142) Load Bus Volts Low
EI 167_VOLTAGE_SENSOR
TEST: Is there voltage reading at EI 167?
EI 181_VOLTAGE_SENSOR
TEST: Is there voltage reading at EI 181?
Yes
No
EI 142
EI 140_VOLTAGE_SENSOR
TEST: Is there voltage reading at EI 140?
Yes
No
EY 141
EI 135_VOLTAGE_SENSOR
TEST: Is there voltage reading at EI 135?
Yes
No
CB 136
Battery 1
Symptom has its TEAMS test in the "Tests Failed" list
This check is redundant, so no TEAMS test result is given for it.
Check's "No" result has its TEAMS test in "Tests Failed" list
Check's "Yes" result has its TEAMS test in "Tests Passed" list
CONFIRMED!
Tests Passed
EI140Voltage[1]:EI140-Voltag
Tests Failed
EI142-Load_Bus_Volts_Low
EI167Voltage[1]:EI167-Voltag
>
<
Send
Close
Diagnosis >>
ADAPT_DTV_TEAMS11_1 - RDS Minimal Fault Diagnosis
Minimal Faults: 1
Residual Faults: 0
Minimal Faults
Residual Faults
EY141[2]<-Battery_Assembly1[1]<

Figure 6. Using TEAMS for path verification analysis

In one instance we first thought we had discovered an error – the TEAMS result did not match the procedure's diagnosis. On more careful scrutiny, we realized it was because we had not noticed that the procedure's check was written in a non-standard way such that the "Yes" answer to the check matched the "Failed" (rather than the "Passed") answer to the corresponding TEAMS test. Once we had corrected for our misunderstanding, the path was confirmed. However, perhaps this indicates the potential for others to make the same mistake. On this basis it might be worthwhile to consider either highlighting the somewhat unusual nature of this check in the procedure, or rephrasing it to be the usual way around.

In several cases we were able to identify *redundancy* in the procedure's diagnostic tree. An instance of this is the pair of checks at the root of the diagnostic tree seen in Figure 6. Using TEAMS we were able to show that just the first of this pair of checks would suffice; the second check is redundant. We did this by verifying each of the paths *without* including TEAMS test result equivalent to the second of the pair of checks. Furthermore, we found that if those two checks return *opposite* results, then TEAMS reveals that it takes more than one component's failure to explain the results.

Figure 6 illustrates the use of TEAMS to explore a path in a procedure. The upper half of the figure displays the procedure's diagnostic tree for the Load Bus Volts Low symptom. The lower half of the figure displays a (partial) screenshot of TEAMS. The path through the tree under investigation is that in which:

- the first check returns result "No".
- the second check is ignored (it's the redundant check discussed above),
- the third check returns the result "Yes", and
- the leaf node EY 141 is the resulting diagnosis.

The TEAMS test equivalents of the symptom and the first test have been placed in the "Tests Failed" list, while the TEAMS test equivalent of the third test (with the “Yes” result) has been placed in the “Test Passed” list. From these test results, TEAMS concludes that a single component’s failure, EY141, completely explains the test results. This confirms the diagnosis of EY141 at the leaf node in the procedure’s diagnostic tree.

6.4 Efficiency Analysis

The efficiency analysis compares the total number of checks in all paths of a procedure’s diagnostic tree with the total number of tests in all paths of the TEAMS generated diagnostic tree for the same symptom. The goal of the efficiency analysis is to see whether it is possible to identify the same set of root causes with fewer tests. The efficiency metric for our investigation at this point was simply the number of checks/tests, without regard to their potentially different resource usage (e.g., some tests may require more computation or power than others), duration (some may take longer than others), or need for human action (some may require a person to observe a setting or threshold). This metric sufficed for our efficiency analysis except in one case, where the procedure needed a person to observe whether or not a change was abrupt or gradual.

• *Same efficiency*. We found that five of the procedures used the same number of checks to identify the same set of root causes as the diagnostic tree generated from the TEAMS model. These tend to be the simplest procedures.

• *Efficiency/isolation tradeoffs*. Six of the procedures’ diagnostic trees had more checks than the number of tests in the corresponding TEAMS-generated diagnostic trees. In each case, however, the TEAMS diagnostic tree involved ambiguity groups that were not present in the procedure’s diagnostic tree. Four of these six diagnostic trees had one ambiguity group, while the remaining two had two ambiguity groups. This suggests that, as we would expect, there are tradeoffs to be made between isolating the root cause to a single fault and keeping down the number of tests. It may be that in some domains, in those cases where the recovery action will be the same for all the ambiguity group’s root causes, that fewer tests will be preferred to some degree of ambiguity. Certainly, the use of the model-driven diagnostic tree to capture and display these tradeoffs was one of the useful results of our study.

• *Better efficiency*. In two other cases the TEAMS diagnostic tree achieved the same result (uniquely identify the root causes) in fewer tests than in the procedure’s tree. One of these cases has been described above and is illustrated in Figure 4. The other case is similar in that the procedure’s tree uses two sequential tests, with both branches of the first test leading to the same second test, whereas the model’s diagnostic tree only uses one test. In these two cases the TEAMS-generated diagnostic tree is more efficient than the procedure’s diagnostic tree. Such results are promising, because they can identify potential, alternate paths to uniquely identifying root causes.

While the goal of our work was to verify the procedures, we also found that the comparison in some cases also indicated possible improvements in the model. Specifically, in the six cases where the diagnostic tree generated from the TEAMS model had fewer tests than the procedures but resulted in an ambiguity group, the additional check needed to disambiguate the root causes in the TEAMS generated tree was present in the procedure’s tree. For example, a test that was present in a procedure but absent in the TEAMS diagnostic tree was to see if Circuit Breaker 162 had tripped. Adding this test to the TEAMS model would not only bring the trees closer together but would also remove the ambiguity group in the model-generated tree.

7 CONCLUSIONS AND FUTURE WORK

We have presented a technique, namely the Diagnostic Tree for Verification (DTV), developed with the goal of leveraging the information contained within auto-generated diagnostic trees in order to assist the verification of diagnostic procedures.

Our most significant finding from the application was that that the model-based diagnostic tree often showed feasible, alternate paths to isolation of the same root cause. We also found two procedures that each had a superfluous step that did not move the diagnosis forward (under the assumption of a single-fault). Conversely, we discovered cases in which the procedures could isolate the symptom to a single root cause while the model-based diagnosis lacked one or more tests needed to do so. We also identified a procedural check that was written in a non-standard way, such that a "Yes" answer corresponded to a "Failed" test and vice versa, leading us to recommend that such mismatches be avoided in order not to risk confusing operators during missions. Perhaps most importantly, the comparisons between the model-based diagnostic tree and the diagnostic steps in the procedure led us (as non-experts in electrical power systems) to have increased understanding and confidence in the operational procedures.

Our approach to using TEAMS to verify a procedure’s diagnosis assumes that performing the checks of the procedure does not cause any change to the system itself before the diagnosis is complete. This assumption would not hold if one of the steps of a procedure reconfigured the system (e.g., switched to a different source of power) partway through diagnosis. We currently have some preliminary and as yet untested ideas on how to extend the DTV approach to handle procedures that do make such changes partway through diagnosis.

ACKNOWLEDGMENT

The research described in this paper was carried out in part at the Jet Propulsion Laboratory, California Institute of Technology, and Ames Research Center under a contract with the National Aeronautic and Space Administration and funded by NASA's OSMA Software Assurance Research Program.

REFERENCES

(Brat et al., 2008) Brat, G., M. Gherorghiu, D. Giannakopouluo, C. Pasareanu, "Verification of Plans and Procedures" in Proceedings of IEEE Aerospace Conference, 2008.

(Button and Chicatelli, 2005) Button R.M. and A. Chicatelli, "Electrical Power System Health Management", In Proc. 1st International Forum on Integrated System Health Engineering and Management in Aerospace, November 2005, Napa, CA.

(Deb et al., 1995) Deb, S., Pattipati, K.R., Raghavan, V., Shakeri, M., Shrestha, R. "Multisignal flow graphs: a novel approach for system testability analysis and fault diagnosis", IEEE Aerospace and Electronics Systems Magazine, Vol.10, No. 5, pp. 14 -25, 1995.

(Frank, 2008). Frank G., "Automation for Operations", Proceedings of AIAA SPACE Conference and Exposition, September 9-11, 2008, San Diego, California.

(Hayashi et al., 2008) Hayashi, M., U. Ravinder, B.Beutter, R. S. McCann, L. Spirkovska and F. Renema, " Operator Performance Evaluation of Fault Management Interfaces for Next-Generation Spacecraft", In Proc. of the 38th International Conference on Environmental Systems, Jun, 2008.

(Kortenkamp et al., 2008) Kortenkamp, D., R. Peter Bonasso and D. Schreckenghost, "Developing and Executing Goal-Based, Adjustably Autonomous Procedures," in Proceedings of the AIAA InfoTech@Aerospace Conference 2007.

(Kurtoglu et al., 2009) Kurtoglu, T., R. Lutz and A. Patterson-Hine, "Towards Verification of Operational Procedures using Auto-Generated Diagnostic Trees," in Proc. of the Annual Conference of the Prognostics and Health Management Society, 2009.

(LTSA 2008) http://www.doc.ic.ac.uk/ltsa/eclipse/.

(McCann et al., 2006) McCann, R., Beutter, B. R., Matessa, M., McCandless, J. W., Spirkovska, L., Liston, D., Hayashi,M., Ravinder, U., Elkins, S., Renema, F., Lawrence,R., & Hamilton, A. "Description and Evaluation of a Real-time Fault Management Concept for Next-generation Space Vehicles", 2006, Internal Report to Johnson Space Center.

(Patterson-Hine et al., 2005) Patterson-Hine, A., Narasimhan, S., Aaseng, G., Biswas, G., Pattipati, K., "A Review of Diagnostic Techniques for ISHM Applications." 1st Integrated Systems Health Engineering and Management Forum. Napa, CA. November 2005.

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

(Polyspace 2008) http://www.polyspace.com

(QSI 2009) QSI, Testability Engineering and Maintenance System (TEAMS) Tool, www.teamsqsi.com.

(Verma et al., 2008) Verma, V., V. Baskaran, H. Utz, R. Harris and C. Fry, "Demonstration of Robust Execution on a NASA Lunar Rover Testbed." In Proc. of International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS), 2008.

(Visser et al., 2003) Visser W., K. Havelund, G. Brat, S. Park and F. Lerda. "Model Checking Programs." In the Automated Software Engineering Journal, Vol. 10, number 2, April 2003.

Tolga Kurtoglu is a Research Scientist with Mission Critical Technologies at the Intelligent Systems Division of the NASA Ames Research Center working for the Systems Health Management group. His research focuses on the development of prognostic and health management systems, model-based diagnosis, design automation and optimization, and risk and reliability based design. He received his Ph.D. in Mechanical Engineering from the University of Texas at Austin in 2007 and has an M.S. degree in the same field from Carnegie Mellon University. Dr. Kurtoglu has published over 40 articles and papers in various journals and conferences and is an active member of ASME, ASEE, AIAA, and AAAI. Prior to his work with NASA, he worked as a professional design engineer at Dell Corporation in Austin, Texas.

Robyn R. Lutz received the Ph.D. degree from the University of Kansas (1980). She has worked at Jet

Propulsion Laboratory/Caltech since 1983, currently as a senior engineer in the Flight Software and Data Systems section. She is also a professor in the Department of Computer Science at Iowa State University. Her work focuses on software safety, software product lines, defect analysis, and formal modeling and analysis, especially for fault detection and recovery. Her research is supported by NASA and the National Science Foundation. She is a member of ACM and a Senior Member of IEEE.

Martin S. Feather is a Principal in the Software Assurance Technology and Reliability group at the Jet Propulsion Laboratory, California Institute of Technology. He works on developing research ideas and maturing them into practice, with current activities in the areas of software and system V&V, safety cases, early phase requirements engineering, and risk informed decision making. He works collaboratively with other researchers, and with software and systems engineering practitioners in cross-disciplinary settings at JPL. He obtained bachelors and masters degrees in mathematics and computer science from Cambridge University, England, and a Ph.D. in artificial intelligence from the University of Edinburgh, Scotland.