Using Auto-Generated Diagnostic Trees for Optimized Fault Handling

Tolga Kurtoglu, Robyn Lutz, and Ann Patterson-Hine

***Abstract*—The launch of a NASA spacecraft depends on a complex set of ground procedures that must be successfully verified and executed. These procedures are a complicated mix of software checks and calibrations, manual inputs and checks of console data, and inspection of physical devices. Development, inspection and execution of these test procedures are currently labor-intensive and critically dependent on human expertise. In this paper, we study how to use the auto-generated diagnostic trees from existing diagnostic models to improve the verification of safety critical launch procedures. Our goal is to identify potential adjustments in the procedures that might offer savings in terms of reduced complexity, increased efficiency and autonomy, and/or reduced cost. The application of the presented method to a spacecraft electrical power system shows the feasibility of the approach and its range of capabilities.**

***Index Terms*—fault handling, verification of operational procedures, model-based diagnosis.**

I. INTRODUCTION

Prior to launch of a NASA spacecraft a complex set of ground procedures must be executed to verify that all launch commit criteria have been met. These procedures are a complicated mix of software checks and calibrations, manual inputs and checks of console data, and inspection of physical devices. Development, inspection and execution of these test procedures are currently labor-intensive and critically dependent on human expertise. They generate thousands of pages of documentation and many opportunities for human error.

There are various techniques developed that can help verify and validate procedures. However, most current verification techniques are largely manual (inspection and reviews) and focus primarily on the conformance of command programs to standard operating procedures. Automated approaches to verification provide much needed support to mission operations, but these approaches are also limited to the verification of syntactic and semantic differences in procedure scripts, and simulation capabilities to validate the equivalence and correctness relations between different system representations [1].

In this work, we are interested in moving beyond correctness toward *optimized correctness*. By optimized correctness, we mean alternative, better ways to achieve tasks intended by standard operating procedures. To accomplish this, we focus on two specific classes of procedures: *procedures to diagnose/isolate a particular failure in a system*, and *procedures to recover from a failure in a system.*

The research problem that we are studying is how to use the auto-generated diagnostic trees from existing software models to improve a procedure's sequence of diagnostic checks and recovery actions. We use the insights gained from exploration of the auto-generated diagnostic trees to identify possible optimizations in the procedures. Our goal is to identify any adjustments to the procedures that might offer savings in terms of reduced complexity (e.g., fewer steps, fewer branches), increased efficiency (e.g., faster results), increased autonomy (e.g., more fault isolation done in software), and/or reduced cost (e.g., reduced usage of scarce resources). This is especially useful for complex systems with redundant elements or functional redundancy. For example, in a system with twelve sensors, the manually developed procedure may check all twelve, whereas the diagnostic tree shows that results from checking only a certain three of the sensors covers the same fault space. In this case, we can consider adjusting the procedure to take advantage of this improved efficiency option. Insights into such possible optimizations offer opportunities for improved fault handling in critical launch procedures.

II. METHOD

Our approach is based on leveraging knowledge and 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 [2]. This information provides an independent perspective that we aim to use in order not only to verify the completeness and consistency of ground test procedures, but also to improve the

Manuscript received March 31, 2009.

Tolga Kurtoglu is with the Mission Critical Technologies/NASA Ames Research Center, Moffett Field, CA 94035 USA (phone: 650-604-1738; e-mail: tolga.kurtoglu@ nasa.gov).

Robyn Lutz is with Jet Propulsion Laboratory/Caltech, Pasadena, CA, 91109 USA and with Iowa State University, Ames, IA 50011 USA. (e-mail rlutz@cs.iastate.edu).

Ann Patterson-Hine is with the NASA Ames Research Center, Moffett Field, CA 94035 USA (e-mail: ann.patterson-hine@nasa.gov).

development of procedures themselves, and thereby to move towards optimized correctness.

Our method uses the Testability Engineering and Maintenance System (TEAMS) tool suite [3] as the primary platform for modeling. TEAMS is built upon the multi-signal modeling formalism [4], 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 procedures 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. In addition, *actions* corresponding to recovery and maintenance tasks can be included in the TEAMS model.

Our investigation uses existing TEAMS models to systematically explore the diagnostic trees that can be produced from these relationships between faults, tests, and actions in the model. A *diagnostic tree* describes an optimized sequence of checks based on the results of prior checks. 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.

For faults that cannot be detected and/or isolated, the diagnostic tree shows the set of indistinguishable failures, called *ambiguity groups*. Thus, all the nodes of the diagnostic tree at the top-level and the intermediate levels are tests in the model. The leaf nodes of the diagnostic tree are either fault-handling actions (e.g., remove/replace a failed part/component) that have been specified for each component in the model, faults that have been isolated but for which there is currently no recovery possible, or an ambiguity group of faults that cannot currently be detected and isolated by the available tests.

These diagnostic trees can be auto-generated from existing TEAMS models, thus making our new optimization technique easy to adopt. By dynamically exploring the model using the set of diagnostic trees generated, we gain insight into the efficacy and efficiency of alternative fault-isolation and recovery paths. This process is illustrated in Figure 1. We are especially interested in finding any shorter sequence of tests and actions (i.e., shorter path) that produces the same results (i.e., isolates the same fault, or recovers from a malfunction), for comparable or lesser cost.

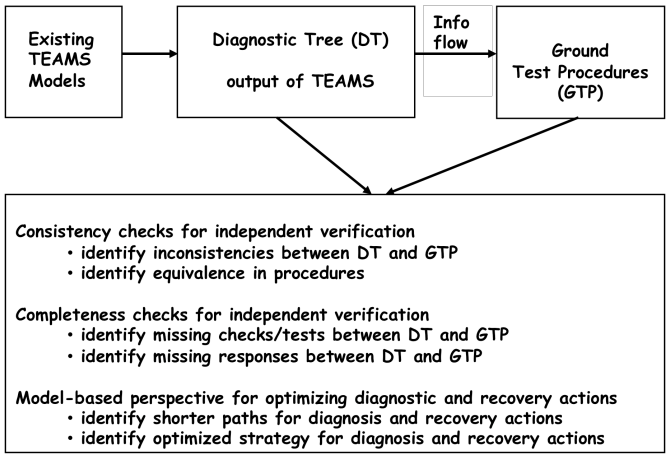

Fig. 1. Overview of the developed method for optimized fault handling and its specific objectives.

By associating *tests* and *actions* in the diagnostic tree with a number of cost parameters (resource usage/timing/agent responsible), we provide a method that can be used to identify improvement opportunities in existing procedures, and to develop more optimal ones. This extends our previous work on three NASA applications (an unpiloted aerial vehicle, MER critical pointing software, and a spacecraft electrical power system testbed) that showed how early consideration of potential anomalies using the diagnostic tree could help build in robustness for handling software contingencies [5-8].

III. PRELIMINARY RESULTS

To demonstrate the feasibility of our technique, we are applying it to portions of a representative electrical power system (EPS), called the Advanced Diagnostic and Prognostic Testbed (ADAPT), located and maintained at the NASA Ames Research Center.

The TEAMS model developed for this purpose consists of basic hardware components including batteries, relays, circuit breakers, a set of operational loads, etc., and sensors measuring the physical characteristics of the system. In addition, the model includes the software architecture of the system, mainly a data acquisition and control system that sends commands to and receives data from the testbed. Overall, the model was developed for detection and isolation of faults including basic software faults. A simplified schematic of the modeled system is shown in Figure 2.

The operational procedures for the EPS system are based upon an advanced caution and warning system developed as an interface concept for a crewed vehicle [9]. Current procedures do not include software faults. However, developing procedures concerning software faults is one of our immediate future tasks. For this preliminary investigation, we selected those procedures focusing on recovery actions for analysis.

In running these analyses, we first identify the *symptoms* (or observable states) of the system that correspond to off-nominal conditions in the system. An example is a “relay

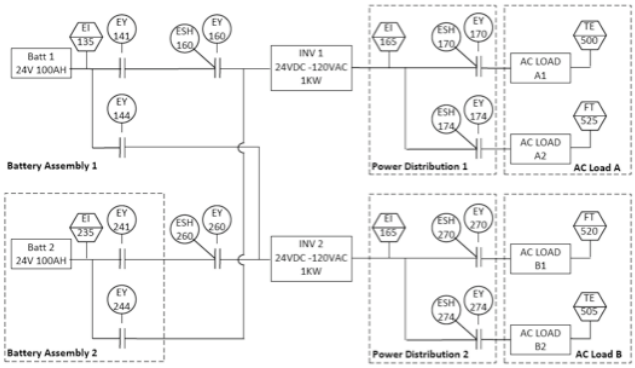

Fig. 2. The simplified schematic of the ADAPT EPS System [10]

failure". We then set particular symptoms to be active in the TEAMS model by using the user interface menu provided for all symptoms defined in the model. This automatically generates a diagnostic tree by forcing the analyzed symptom to be the root node of the tree. As described before, the diagnostic trees capture a sequence of tests, checks, and set-up that needs to be performed in order to diagnose/isolate the fault that is causing the analyzed *symptom*. The isolated fault(s) may also be annotated with recovery actions in the model, in which case these actions also appear in the appropriate intermediate or leaf nodes of the diagnostic tree.

Similarly, the procedures in [9] document describe the steps required to recover from a failure in the system. These operational steps 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 as illustrated in Figure 3.

By comparing the sequence of steps described in the procedure to isolate the causes of the anomaly and recover from it with the diagnostic paths of the diagnostic tree, we seek to find relationships among the two representations. Our current effort is aimed at finding any shorter sequence of tests and actions (i.e., a shorter path) that produces the same results (i.e., isolates the same fault, or recovers from a malfunction), for comparable or lesser cost in time or resources. Challenges with which we are currently contending largely involve: (1) management of the relationships between the many-to-many elements in the model and in the procedural steps, (2) checks in the procedures involving human elements (e.g., intervention) not currently represented in the model and (3) scalability of the approach for larger systems. We hypothesize that our future work will show that Challenges 1 and 3 can be handled within the existing TEAMS framework and that Challenge 2 will encourage fuller representation of human-computer interactions in the models (i.e., will consider the human as part of the system to be modeled).

IV. KEY CONCLUSIONS

We use auto-generated diagnostic trees to identify possible improvements in critical launch procedures. Preliminary results indicate that the method presented here facilitates the identification of potential gaps in the coverage of the ground test procedures.

Procedure: Battery Output Voltage Anomaly
(checks for battery failure and reconfigures the system to use the redundant battery)

- Step 1: ***Verify*** Config Battery 1 to AC Load A1
- Step 2: ***If*** Battery1 Output Voltage **(EY 141 reading)** < $V_{threshold}$
- Step 3: ***Turn Off*** Battery 1
- Step 4: ***Turn On*** Battery 2
- Step 5: ***Command*** EY 241, EY 260, EY 274 closed
- Step 6: ***Verify*** TE 505 within operational limits

Fig. 3. A procedure for recovering from a battery failure in the EPS system.

Specifically, our method helps identify: (1) inconsistencies between the information generated by diagnostic trees and ground test procedures, (2) missing checks/tests and responses between diagnostic trees and ground test procedures, (3) shorter paths for diagnosis/isolation and recovery actions than those suggested by operating procedures, and (4) an optimized strategy for diagnosis/isolation and recovery actions.

V. ACKNOWLEDGEMENT

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

[1] G. Brat, M. Gherorghiu, D. Giannakopouluo, C. Pasareanu, "Verification of Plans and Procedures" in Proceedings of IEEE Aerospace Conference, 2008.

[2] 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.

[3] QSI, Testability Engineering and Maintenance System (TEAMS) Tool, www.teamsqsi.com.

[4] 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.

[5] R. Lutz, A. Patterson-Hine, S. Nelson, C. Frost, D. Tal, and R. Harris, "Using Obstacle Analysis to Identify Contingency Requirements on an Unpiloted Aerial Vehicle", Requirements Engineering Journal, 12(1), Jan, 2007, pp. 41-54.

[6] R. Lutz, A. Patterson-Hine, S. Poll, C. Domagala and S. Ghosal, "Tool-Supported Software Contingency Analysis," 1st International Workshop on Aerospace Software Engineering, in conjunction with 29th International Conference on Software Engineering (ICSE 2007), May 20-21, 2007, Minneapolis, MN.

[7] R. Lutz and A. Patterson-Hine, "Tool-Supported Verification of Contingency Design: Poster and Abstract", NASA SMD/PSD Fault Management Workshop, April 14 – 16, 2008, New Orleans, LA.

[8] R. Lutz, and A. Patterson-Hine, "Using Fault Modeling in Safety Cases", ISSRE 2008, pp. 271-276.

[9] 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.

[10] S. Ghosal, and M. Azam, "Technology Transfer of Contingency Software Process", Phase III, Final Report, 2008, Qualtech Systems Inc.