Modeling and Performance Considerations for Automated Fault Isolation in Complex Systems

Bob Ferrell
NASA Kennedy Space Center
Mailstop: NE-E9
Kennedy Space Center, FL 32899
321-867-6678
bob.a.ferrell@nasa.gov

Rebecca Oostdyk
ASRC Aerospace Corporation
Mailstop : ASRC-25
Kennedy Space Center, FL 32899
321-867-5637
rebecca.l.oostdyk@nasa.gov

Abstract—The purpose of this paper is to document the modeling considerations and performance metrics that were examined in the development of a large-scale Fault Detection, Isolation and Recovery (FDIR) system. The FDIR system is envisioned to perform health management functions for both a launch vehicle and the ground systems that support the vehicle during checkout and launch countdown by using a suite of complimentary software tools that alert operators to anomalies and failures in real-time.

The FDIR team members developed a set of operational requirements for the models that would be used for fault isolation and worked closely with the vendor of the software tools selected for fault isolation to ensure that the software was able to meet the requirements. Once the requirements were established, example models of sufficient complexity were used to test the performance of the software.

The results of the performance testing demonstrated the need for enhancements to the software in order to meet the demands of the full-scale ground and vehicle FDIR system. The paper highlights the importance of the development of operational requirements and preliminary performance testing as a strategy for identifying deficiencies in highly scalable systems and rectifying those deficiencies before they imperil the success of the project.

TABLE OF CONTENTS

1. INTRODUCTION

Integrated System Health Management (ISHM) encompasses the design, tools, strategies, devices and algorithms that are employed to improve safety, reduce costs, and increase reliability and availability of a system. At its core, ISHM provides some measure of health of a system that allows maintenance and operations to be performed on a conditional basis to ensure the system will continue to operate safely and reliably within a given confidence interval.

One significant aspect of ISHM is the capability to detect failure conditions in the system and isolate the failure to its root cause. Fault detection methods are wide-ranging and include a range of strategies from simple limit checking of measurements to data mining and statistical analysis to intelligent devices and built-in tests that identify failures at the source. Once a fault or off-nominal behavior is detected, fault isolation techniques are employed to locate the failure mode or modes of the system and implicate bad or suspected components for further testing and replacement.

The Fault Detection, Isolation and Recovery (FDIR) project funded by NASA's Exploration Technology Development Program (ETDP) is purposed to mature fault detection, fault isolation, anomaly detection, and prognostics technologies for use in the new Constellation Program and future extra-planetary missions. FDIR is intended and designed to be integrated with Ground Operations to automate fault detection and isolation during maintenance and checkout as well as launch countdown activities of ground and launch vehicle systems. The FDIR architecture supports the integration of several ISHM capabilities, but this paper will focus on the fault isolation feature.

Model Selection

Automated fault isolation requires the operational behavior of the system and its components to be defined so that it can be compared to the real-time system operation. The operational behavior is best captured in a model, and the type of model that is selected will be dependent on the application, resources and computing platform for the ISHM system.

For the FDIR project, several model-based diagnostic approaches were considered before the functional fault model (FFM) was chosen to represent the ground and launch vehicle systems. A physics-based model by which live data can be compared to theoretical values is the most accurate approach to detecting failures of the system, but the complexity of the ground and vehicle systems in all of their potential configurations and mission phases would make the physics model difficult to verify by subject matter experts.

If the modeled systems had archives of historical data, the physics model could be compared to past operational behavior, but the Constellation ground and vehicle systems have yet to be built or operated. Therefore, the physics-based model was not chosen due to the lack of verification methods. Another candidate model was a rule-based expert system. The intent of the models is to aid engineers and operators who are monitoring the systems by providing information about the health of the components and the entire system. An expert system would be able to determine the state of the components and system in a reliable, repeatable manner assuming that its knowledgebase was comparable to that of the engineer or operator. However, the process of translating the engineer's expertise into a model requires a significant amount of the engineer's time. Since the modeling effort was meant to be carried out by a group of non-subject matter experts, the rule-based expert system modeling approach was discarded.

The FFM was selected because it allowed the modeling team to review system design documentation independently from the operators and engineers, create a model that resembles the schematic diagrams that are familiar to the experts, and have the experts verify the model without a large time commitment. Functional fault modeling involves capturing failure modes that have been identified both at design time and operationally. The Failure Mode Effects Analysis (FMEA) provides design-time failure modes, and problem reporting and corrective action databases and manufacturer datasheets provide insight into operational failures that are likely to occur or have been encountered for similar components in system. Once the failure modes have been catalogued, they can be placed in a model that maps the effects of the failures on system operation. Essentially, a FFM is responsible for identifying the failure effect propagation paths (FEPPs) from a failure mode to the observation point where it is detected. The FFM is then used in real-time to infer which failure modes or failed components could cause the observed behavior of the system.

Although the FFM has many advantages, one weakness of the technique is that the FFM only captures known failures. In order to supplement the functional fault model, the FDIR architecture also includes a data-driven model that detects anomalies. The data-driven model was trained on data from similar ground and vehicle systems to create a knowledgebase of in-family behavior. After sufficient training on nominal data, the data-driven model provides a measure of how closely the data it is monitoring matches the training data. The data-driven model is not able to use anomalous scores to isolate to a suspect component or system, but it does provide information about which measurements are contributing the most to the anomalous scores so that an operator or engineer can be alerted to a potential problem. The data-driven and functional fault models are complementary technologies that cover both known and unknown conditions of the system.

Testability Engineering And Maintenance System (TEAMS)

A trade study evaluating current functional fault modeling software was executed by the FDIR project. The trade study involved a survey of papers on state-of-the-art model based diagnostic, including [References]. The outcome of the FDIR trade study was a recommendation to use Qualtech Systems Incorporated's (QSI's) Testability Engineering And Maintenance System (TEAMS) software suite. The TEAMS software products include TEAMS Designer, TEAMS-RT and TEAMS Remote Diagnostic Server (RDS). TEAMS Designer is the user interface for building and analyzing the FFMs. It provides a graphical environment where the system's failure modes, components, hierarchical structure, interconnections, and observation points are captured in block diagrams that closely resemble system schematics. TEAMS-RT is the software engine that interprets a set of pass or fail results at the observation points to determine whether the failure modes and components in the model are good, bad, suspect or unknown. TEAMS RDS is a server application which provides session management and archival services for one or more instances of TEAMS-RT. Although TEAMS-RT is a standalone product, a customer may opt to use TEAMS RDS for its management features in lieu of writing custom software that performs the same functions. The TEAMS commercial off-the-shelf (COTS) products have more capability than FFM creation and real-time fault isolation, but the tool was chosen for the FDIR project specifically for those functions.

After a model is created in TEAMS Designer, the model information is exported in the form of a dependency matrix (D-Matrix). The D-Matrix contains rows of failure modes and columns of observation points, or test points. If a particular failure mode is detectable by a test point, a one is placed in the D-Matrix at the intersection of the failure mode and test point. A model will export one D-Matrix for each system mode or configuration that is identified by the modeler. The system modes are defined by unique combinations of switches in the model that enable or disrupt the FEPPs.

Once the D-Matrices have been defined, the TEAMS-RT engine uses them to isolate faults to a failure mode or modes based on the pass or fail state of the test points. The TEAMS-RT engine outputs lists of failure modes that are "bad", "suspect" and "unknown", and all other failure modes are assumed "good." A failure mode which is "good," or not failed, is one which is mapped to any test point with a pass result. A "bad" failure mode is one which has a FEPP to a test point with a failed result, where the test point is not mapped to any other failure modes besides failure modes that have already been determined to be "good." A "suspect" failure mode is one which has a FEPP to a test point with a failed result and the failure mode is not the only failure mode mapped to that test point. "Suspect" failure modes are by definition an ambiguity group for a single test

point. Finally, "unknown" failure modes are those failure modes whose FEPPs map only to test points for which there are no pass or fail results.

Since the model includes hierarchy information that relates a failure mode to a component or other higher level (such as a line replaceable unit or subsystem), the same D-Matrix can be used to diagnose the system at different levels of resolution depending on the needs of the application.

2. FDIR Operational Requirements

The COTS TEAMS products were required to meet the operating criteria of other software tools selected for the FDIR application as well as customer requirements for the functionality of the fault isolation features. These requirements include:

(1) Model shall have clear mapping back to physical system to aid in initial model validation by system experts and maintainability and sustainability by system design engineers

(2) The model shall be capable of isolating to multiple levels of resolution for the vehicle and ground systems (i.e. failure mode, component, line replaceable unit, etc.).

(3) Modeling techniques and practices shall be scalable for a large, integrated model that encompasses vehicle systems, ground systems, and facility infrastructure. The integrated model will have an estimated 40,000 failure modes and 50,000 test points if it includes the ground systems, launch vehicle and Orion capsule.

(4) The model shall perform real-time fault isolation (TEAMS-RT will return a diagnosis with unknown, good, suspect and bad lists) within 1 second of fault detection (pass/fail of tests). The model shall perform an update of the health status within one second of being passed the pass/fail test results.

(5) The model shall be re-configurable on-the-fly to accommodate different mission phases, system modes, or vehicle configurations. The re-configuration of the large, integrated model shall not exceed the performance requirement of < 1 second to fault isolation

(6) The real-time fault isolation tools shall be certified in accordance with GOP507007 Ground Element Command, Control and Communication Software Development Plan. Ground Operations will have to recommend which parts of GOP507007 apply to the tools and how the tools will be certified.

Scalability Questions

The operational requirements raise two important questions which the FDIR project sought to answer before TEAMS was used to model a large, integrated system. The answer to these questions must be answered with both in mind.

(1) What model integration strategy should be employed for a large, integrated ground and vehicle system?

(2) What configuration of TEAMS tools should be used to accommodate the model integration strategy?

3. Model Integration Strategies

Several integration strategy options were considered for integrating Ares and Ground Operations FFMs into a solution that would support FDIR operations for combined Flight and Ground operations.

At the turnover of the Ares vehicle to Kennedy Space Center (KSC) in Florida, there will be three data products provided to FDIR, as shown in the diagram below.

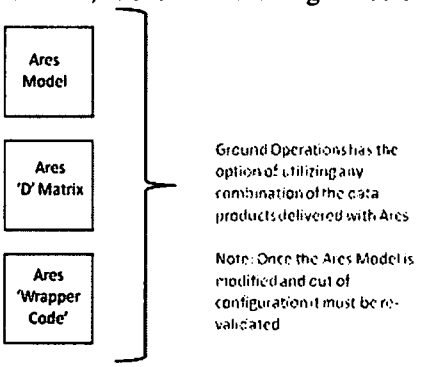

Figure 1 – Ares Vehicle Data Products

As the note above states, once any of these data products are modified then that data product is no longer in configuration. The Verification and Validation (V&V) of that product no longer is applicable once this occurs.

The first integration strategy option is shown below and consists of utilizing the single Ares FFM provided at turnover, along with a single Ground Operations FFM. A high level model Interface Control Document (ICD) would be maintained in order to ensure compatibility and integration between the two FFMs.

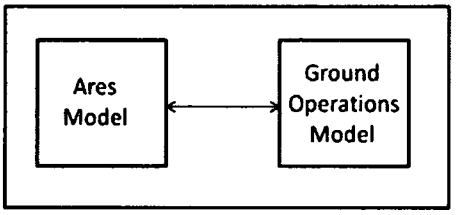

Figure 2 – Model Integration Strategy #1

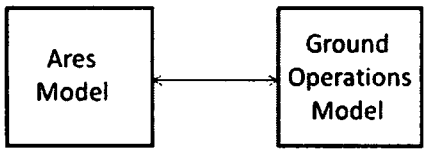

Figure 3 – Model Integration Strategy #2

In order to increase the performance of option #1, integration strategy option #2 shown above was instituted. This option isolates the Ares model and Ground Operations model into separate reasoners, thereby reducing the load on a single reasoner.

In the case where option #2 does not provide enough performance then the Ground Operations Functional Fault Model may be separated into sub-models. This is shown below as Integration Strategy Option #3.

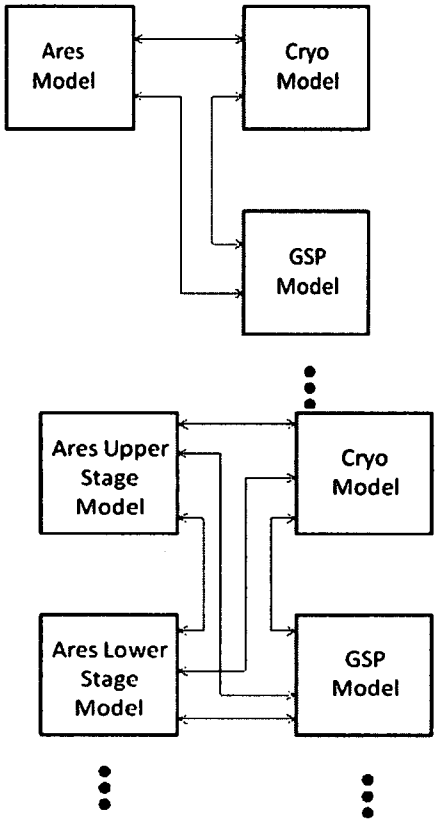

Figure 4 – Model Integration Strategy #3

In the case where option #3 does not provide enough performance, integration strategy option #4 was devised for ultimate performance. It does this by breaking both the Ground Operations FFM into sub-models as well as the Ares FFM into sub-models. While this does provide performance via "distributed computing" (each sub-model in its own reasoner), there are several disadvantages to this option. One of the most significant drawbacks is the fact that the integrity of the Ares FFM is violated and therefore is no longer in a certified V&V configuration. These four integration strategy options were identified as being worthy for further examination. Each of these integration strategy options has advantages and disadvantages that deal with performance, model ICDs, and the number of reasoners that will be required.

Model Interface Control Documents

4. Model Real-Time Performance

After the model integration strategies were identified, the next issue to be resolved was whether or not it is feasible to place the Ares and ground models into a single large model due to performance constraints in TEAMS-RT. As previously stated, the estimated worst-case size of the D-Matrix is 40,000 failure modes by 50,000 test points with 500 system modes. The main performance constraint is the operational requirement that TEAMS-RT would need to return a diagnosis of bad, suspect and unknown failure modes in less than one second from the time that the pass and fail test results were received by TEAMS-RT.

Although performance is known to be better while running multiple TEAMS-RT reasoners independently, the FDIR project intends to use the TEAMS RDS management and archival services for its implementation so TEAMS RDS was allowed to manage multiple instances of TEAMS-RT during the performance benchmarking. Although the FDIR project is expecting to run TEAMS RDS on an AIX server, a Linux machine was used for the benchmarking because TEAMS RDS has not yet been ported to the AIX platform. The Linux machine has less power than the AIX server, so any results would collected would be worst case. The performance testing was completed on a Dell Desktop with the following specs:

(1) 2 Quad Core Xeon Proc E5420, 2.50GHz, 2X6MB L2 Cache,1333MHz Processors

(2) 4GB, DDR2 ECC SDRAM Memory 667MHz, 4X1GB System Memory

(3) 80GB SATA 3.0Gb/s, 7200RPM HardDrive with 8MB DataBurst Cache

(4) Red Hat Linux OS

The models used for the performance benchmarking were required to be as realistic as possible to validate the performance metrics. D-Matrix size, density and subsystem coupling all have effects on the performance of the TEAMS-RT engine. Interactions between subsystems in a model affect the density off-diagonal regions of the D-Matrix. These interactions represent the propagation of failure modes from one subsystem to another. When two models share FEPPs, the resultant D-Matrix will have cells

populated in regions A-B and B-A as shown in the figure below. If the two models are completely independent, no 1's will appear in the A-B and B-A regions.

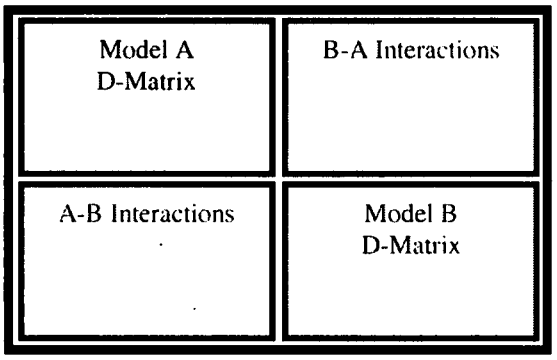

Figure 5 – Integrated Model D-Matrix

In order to simulate models with appropriate densities and cross-subsystem interactions, the large models used in the performance benchmarking were created by copying two ground and Ares vehicle models from a previous prototype multiple times and linking them together to simulate the cross-coupling between subsystems.

The TEAMS RDS CPU usage was monitored for four different model sizes: approximately 1,000 failure modes, 5,000 failure modes, 10,000 failure modes and 35,000 failure modes. Detailed specifications for each of the four models follow:

1000 failure modes
RUNTIME_INPUT_DATA
TEAMS_RT_VERSION 11.00
NUM_ASPECT_ROWS 785
NUM_SWITCH_ROWS 108
NUM_AND_ROWS 0
NUM_ACTUAL_TESTS 348
NUM_TEST_COLUMNS 348
NUM_SWITCH_COLUMNS 138

5000 failure modes
RUNTIME_INPUT_DATA
TEAMS_RT_VERSION 11.00
NUM_ASPECT_ROWS 5495
NUM_SWITCH_ROWS 756
NUM_AND_ROWS 0
NUM_ACTUAL_TESTS 2436
NUM_TEST_COLUMNS 2436
NUM_SWITCH_COLUMNS 966

10000 failure modes
RUNTIME_INPUT_DATA
TEAMS_RT_VERSION 11.00
NUM_ASPECT_ROWS 10990
NUM_SWITCH_ROWS 1512
NUM_AND_ROWS 0
NUM_ACTUAL_TESTS 4872
NUM_TEST_COLUMNS 4872
NUM_SWITCH_COLUMNS 1932

35000 failure modes
RUNTIME_INPUT_DATA
TEAMS_RT_VERSION 11.00
NUM_ASPECT_ROWS 38465
NUM_SWITCH_ROWS 5292
NUM_AND_ROWS 0
NUM_ACTUAL_TESTS 17052
NUM_TEST_COLUMNS 17052
NUM_SWITCH_COLUMNS 6762

Performance testing for a model with 50000 failure modes was attempted, but the model size reached the limit for a 32-bit machine on the size of the D-Matrix and TEAMS Designer was unable to export the model files required for use by TEAMS-RT. If the integrated Ground Operations and Vehicle models are expected to be > 35000 failure modes, then QSI will need to implement a 64-bit version of TEAMS to support the large integrated models.

For the 1,000 failure mode model, CPU utilization on the Linux server for any of the processes was negligible. The 5,000 failure mode model typically had less than 2% CPU (single core) utilization for any realistic fault scenario and up to 4% CPU utilization for mySQL. By simulating a catastrophic failure with 1553 failures we were able to take CPU utilization for the reasoner increase to 15%.

When operating the 10,000 failure mode model with realistic faults (<100 faults at a given time), CPU usage showed typical values from 5-6%. The catastrophic failure scenario increased CPU utilization to a peak of 72%. System Mode changes for this model and the smaller models were all performed in less than 1 second.

The 35,000 failure mode model performed similarly to the 10,000 failure mode model with 5-6% CPU utilization for realistic faults. The catastrophic failure scenario increased response time to around 5 seconds and CPU (single core) utilization to 100%. The most significant performance delta occurred with system mode changes. The 35,000 failure mode model has 4,018 switches in it and any system mode change alters the state of approximately half of these. When system mode changes occur CPU utilization goes to 100% for nearly 60 seconds. Switching a more realistic 100 switches is accomplished in less than 15 seconds and 300 switches in less than 30 seconds.

As stated previously, the 50K matrix was not tested due to 32-bit OS limitations.

In addition to measuring CPU utilization by TEAMS RDS for individual models, we ran three 10,000 failure mode models and a 35,000 failure mode model simultaneously as four instances of TEAMS-RT in TEAMS RDS. As expected, each instance of TEAMS-RT was constrained to a

single core and the results were consistent with the previously obtained performance numbers.

5. Model Validation and Verification

The FDIR software will be utilized within the Launch Control System (LCS) of Ground Operations, and therefore must meet the LCS requirements for COTS software. The LCS requirements are:

(1) The COTS must be developed by an organization with software CMM Maturity Level 3 or higher.

(2) The COTS must be developed by an organization with a CMMI SE/SW Capability Level 2 or higher as measured by an SEI authorized lead appraiser form an external organization in the following areas:

   a. Requirements Management

   b. Configuration Management

   c. Process and Product Quality Assurance

   d. Measurement and Analysis

   e. Project Planning

   f. Project Monitoring and Control

   g. Supplier Agreement Management

(3) In the event that number one or two above cannot be satisfied by the organization that developed the COTS, the alternative is to provide, as a minimum, the following documentation to NASA. NASA will then determine whether the processes utilized in creating the software allow the software to be utilized in a safety critical system.

   a. Product Software Design and Implementation

   b. Software Management Processes

      i. Configuration Management

      ii. Verification and Validation

      iii. Software Testing

   c. Testing

      i. Acceptance Test Plan

      ii. Test Results (both current and subsequent releases)

      iii. Test Procedures

   d. Defect Tracking Process

   e. Defect History (both current and subsequent releases)

While QSI does not have CMM or CMMI certification, they do have processes in place that are ISO certified. Continuing ISO audits ensure that they are following their documented processes. The ISO certification does not meet the requirements as outlined above; however by examining the ISO controlled processes involving components described in (3) above for suitability in the areas outlined in (2), one can determine the suitability of utilizing the COTS software. QSI's processes were examined with this in mind. QSI's primary processes utilize Mantis, CVS, and electronically controlled documentation. The initial examination proved successful and no deficient areas were identified. In addition, QSI provided electronic copies of all areas that were examined.

6. Conclusion

The performance benchmarking numbers prove that any of the four model integration strategies would be feasible for the FDIR project as long as QSI is able to provide a 64-bit implementation of TEAMS-RT and TEAMS RDS. In the absence of a 64-bit implementation, two instances of TEAMS-RT would have to be instituted in TEAMS RDS – one for the ground system models and one for the Ares vehicle model. However, it is more favorable to work with QSI to implement 64-bit TEAMS in order to maintain the cross-coupling between ground and vehicle models without custom external code. As for real-time performance, the benchmarking confirms the ability of a Linux computer running TEAMS RDS to return fault isolation diagnoses within the one second time frame and with reasonable CPU usage within the constraints of the operational requirements.

The paper highlights the importance of the development of operational requirements and preliminary performance testing as a strategy for identifying deficiencies in highly scalable systems and rectifying those deficiencies before widespread model building begins. Discovering scalability problems early in the design cycle and establishing modeling methods to avert or alleviate their impact are vital to the success of any sizeable modeling undertaking.

References

Biographies

ACKNOWLEDGEMENTS

We would like to thank NASA's Exploration Technology Development Program for their past and future funding of the Fault Detection, Isolation and Recovery project to develop and mature fault isolation technologies for future space missions