Ares I-X Ground Diagnostic Prototype

Mark Schwabacher[*] and Rodney Martin[†]
NASA Ames Research Center, Moffett Field, CA 94035

Robert Waterman[‡]
NASA John F. Kennedy Space Center, Cape Canaveral, FL 32899

Rebecca Oostdyk[§]
ASRC Aerospace Corporation, Cape Canaveral, FL 32899

John Ossenfort[**] and Bryan Matthews[††]
Stinger Ghaffarian Technologies, Inc., Moffett Field, CA 94035

The automation of pre-launch diagnostics for launch vehicles offers three potential benefits: improving safety, reducing cost, and reducing launch delays. The Ares I-X Ground Diagnostic Prototype demonstrated anomaly detection, fault detection, fault isolation, and diagnostics for the Ares I-X first-stage Thrust Vector Control and for the associated ground hydraulics while the vehicle was in the Vehicle Assembly Building at Kennedy Space Center (KSC) and while it was on the launch pad. The prototype combines three existing tools. The first tool, TEAMS (Testability Engineering and Maintenance System), is a model-based tool from Qualtech Systems Inc. for fault isolation and diagnostics. The second tool, SHINE (Spacecraft Health Inference Engine), is a rule-based expert system that was developed at the NASA Jet Propulsion Laboratory. We developed SHINE rules for fault detection and mode identification, and used the outputs of SHINE as inputs to TEAMS. The third tool, IMS (Inductive Monitoring System), is an anomaly detection tool that was developed at NASA Ames Research Center. The three tools were integrated and deployed to KSC, where they were interfaced with live data. This paper describes how the prototype performed during the period before the launch, including accuracy and computer resource usage. The paper concludes with some of the lessons that we learned from the experience of developing and deploying the prototype.

I. Definitions

Many of the words, phrases, and acronyms that are used in the field of Integrated Systems Health Management (ISHM) are ill-defined, so we begin with some definitions. *Anomaly detection* is detecting that new data is different from what has been seen before. An anomaly may or may not be a fault. *Fault detection* is detecting that something has failed, resulting in a loss of function. *Fault isolation* is determining the location of the fault. *Diagnostics* is determining the specific failure mode. All of these methods take as input the steam of sensor values and commands, and output assessments of the system's health. The Ares I-X Ground Diagnostic Prototype performs anomaly detection, fault detection, fault isolation, and diagnostics.

[*] Computer Scientist, Intelligent Systems Division, MS 269-3, AIAA Member.
[†] Computer Engineer, Intelligent Systems Division, MS 269-1.
[‡] Command & Control Architecture Lead, Architecture Integration & Management Office, MS LX-S3.
[§] Electrical Engineer, Advanced Electronics and Technology Development Group, MS ASRC-25.
[**] Computer Scientist, Intelligent Systems Division, MS 269-1.
[††] Systems Engineer, Intelligent Systems Division, MS 269-1.

II. Introduction

THE automation of pre-launch diagnostics for launch vehicles offers three potential benefits. First, it offers the potential to improve safety by detecting faults that might otherwise have been missed so that they can be corrected before launch. Second, it offers the potential to reduce launch delays by more quickly diagnosing the cause of anomalies that occur during pre-launch processing. Reducing launch delays will be critical to the success of NASA's planned future missions that require in-orbit rendezvous. Third, it offers the potential to reduce costs, both by reducing launch delays and by reducing the number of people needed to monitor the pre-launch process.

Ares I is the launch vehicle that NASA is currently developing to bring the Orion capsule and its crew of four astronauts to low-earth orbit on their way to the moon. Ares I-X[1] was the first uninhabited test flight of Ares I. It launched on October 28, 2009 (see Figure 1). The Ares I-X Ground Diagnostic Prototype (GDP) is a prototype ground diagnostic system that provided anomaly detection, fault detection, fault isolation, and diagnostics for the Ares I-X first-stage Thrust Vector Control (TVC) and for the associated ground Hydraulic Support System (HSS) while the vehicle was in the Vehicle Assembly Building (VAB) at Kennedy Space Center (KSC) and while it was on the launch pad. The TVC is used to steer the vehicle during ascent by moving the nozzle of the first-stage solid rocket booster. The HSS provides hydraulic pressure for testing the TVC before launch. GDP is intended to serve as a prototype of a future operational ground diagnostic system for Ares I or other future launch vehicles.

Figure 1. Ares I-X launch. *Ares I-X launched from Kennedy Space Center on October 28, 2009.*

The prototype combines three existing diagnostic tools. The first tool, TEAMS (Testability Engineering and Maintenance System), is a model-based tool that is a commercial product from Qualtech Systems Inc. (http://teamqsi.com). It uses a qualitative model of failure propagation to perform fault isolation and diagnostics. We adapted an existing TEAMS model of the TVC in order to use it for diagnostics, and developed a TEAMS model of the ground hydraulics. The second tool, SHINE (Spacecraft Health Inference Engine), is a rule-based expert system that was developed at the NASA Jet Propulsion Laboratory[2]. We developed SHINE rules for fault detection and mode identification. The prototype uses the outputs of SHINE as inputs to TEAMS. The third tool, IMS (Inductive Monitoring System)[3], is an anomaly detection tool that was developed at NASA Ames Research Center and is currently being used to monitor three systems on the International Space Station. IMS automatically "learns" a model of historical nominal data in the form of a set of clusters and signals an alarm when new data fails to match this model. IMS offers the potential to detect faults that have not been modeled. The three tools were integrated and deployed to Hangar AE at KSC, where they were interfaced with live data from the Ares I-X vehicle and from the ground hydraulics. The

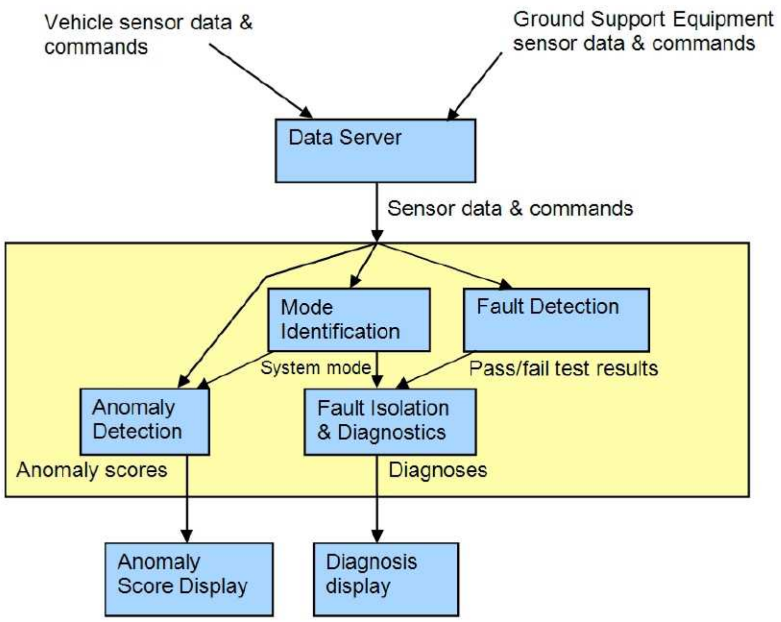


Figure 2. Ares I-X GDP Architecture. *The architecture combines anomaly detection, fault detection, and diagnosis for the vehicle and the ground support equipment.*

outputs of the tools were displayed on a console in Hangar AE, which is one of the locations from which the Ares I-X launch was monitored. The architecture of the prototype is shown in Figure 2.

In a previous publication[4], we discussed how we selected the three tools based primarily on their ability to be certified for human spaceflight, and described our plans for the prototype. This paper describes how the prototype performed during the period before the launch. Section III describes the data that was used to test the prototype, including the simulated failures that were inserted into historical Shuttle data. Section IV describes how we used TEAMS and SHINE for fault detection and fault isolation, and includes a summary of the results from those tools. Section V describes how we used IMS for anomaly detection, and includes a summary of the results from that tool. Section VI describes the graphical tool that was used to display the outputs of the prototype. Section VII summarizes the computational performance of the prototype. The prototype was not certified, but Section VIII presents a brief summary of how we considered the need for certification in the design of the prototype and how we would propose to get it certified if it were deployed as an operational system. The paper concludes with some of the lessons than we learned from the experience of developing and deploying the prototype.

III. Data Used for Testing the Prototype

During the development and testing of Ares I-X GDP, Ares I-X data was not yet available. We therefore used historical Space Shuttle data to test the entire prototype. The Space Shuttle Solid Rocket Booster (SRB) TVC is virtually identical to the Ares I-X first-stage TVC, so the SRB TVC data was expected to be very similar to the Ares I-X TVC data. Similarly, the ground hydraulic system used with the SRB TVC is virtually identical to the ground hydraulic system used with the Ares I-X TVC. These expectations held up modestly well after our post-flight analysis, in consideration of all the tools that were deployed to support failure and anomaly detection, as will be discussed in subsequent sections. The differences that we found in the data were caused by differences in operations between Shuttle and Ares I-X, rather than by differences in the TVC or HSS hardware.

The SRB TVC and the associated ground hydraulic system have had very few failures. We thus had available to us an abundance of nominal data, but very little failure data. We therefore decided to develop a set of failure simulations that could be used to test the ability of the prototype to detect and diagnose failures. We inserted simulated failures into the historical Shuttle data, and used the resulting data sets to test the prototype before the Ares I-X launch. We developed simulations of the following four failure modes:

- FSM (Fuel Supply Module) pressure drop due to N_2H_4 (Hydrazine) leak
- Hydraulic pumping unit over-temperature failure
- Hydraulic fluid reservoir level drop due to hydraulic fluid leak
- Actuator stuck during actuator positioning test

For the FSM failure, we developed a physics-based simulation. For the other three failure modes, we did not have sufficient information available to develop physics-based simulations, so we used simple linear approximations. More details of the simulated failures can be found in Ref. 5.

After the Ares I-X vehicle was assembled and powered up in the Vehicle Assembly Building (VAB), we began testing the prototype using live data from the vehicle. This testing continued until launch.

IV. Fault Detection, Fault Isolation, and Diagnostics

The software tool that was chosen to provide fault isolation and diagnostics for GDP is called TEAMS. TEAMS is a software package for performing model-based diagnostics based on the system design, component definitions, inputs, outputs, connectivity, signal flow and hierarchy. The TEAMS suite of tools includes TEAMS Designer, a graphical tool that is used to build TEAMS models, and TEAMS-RT, a real-time fault isolation and diagnosis engine that uses the TEAMS models. We used TEAMS Designer to build TEAMS models of the TVC and HSS, and deployed TEAMS-RT to Hangar AE.

TEAMS-RT requires as its inputs a set of test results and the current system mode. Each test result has a "pass" or "fail" value, and typically represents the result of a test that is performed on the TVC. Most of the tests involve comparing a sensor value with a threshold. GDP uses a combination of SHINE rules and C code to compute the test results and the system mode. This "wrapper" code performs fault detection and mode identification. The diagnoses produced by TEAMS-RT label all of the components in the model as "good" (definitely not failed), "suspect" (possibly failed), or "bad" (definitely failed).

TEAMS models are hierarchical; it is possible to "zoom in" on any of the modules displayed in TEAMS Designer to see the components within them. At the lowest level, the failure modes and test points are modeled. (The test results described above are input to the model at the test points.) We built separate TEAMS models of the TVC and the HSS, and then integrated them into a single model by using a simple top-level model. The failure modes and components were modeled based on information provided in the Failure Modes and Effects Analysis for the vehicle, along with schematics, diagrams, users' manuals, and knowledge from TVC engineers and systems engineers. The integrated model has 655 components, 893 failure modes, and 263 tests. The tests use data from 281 measurements.

Prior to the Ares I-X launch, we tested the TEAMS model and the wrapper code using historical Shuttle data from seven flights, into which we inserted simulated failures. This testing revealed some bugs in the SHINE rules. In some cases, we had initially made some assumptions about the TVC testing procedures that turned out to be false for at least one of the seven Shuttle flights. Other bugs were simply coding errors. After fixing all of the bugs, we were able to run the prototype on the data from all seven Shuttle flights with no false alarms, and with all of the simulated failures correctly detected.

We obtained the first Ares I-X data shortly after Ares I-X had its initial power-up in the VAB, approximately six weeks before the launch. When we tested the prototype on this Ares I-X data, it produced a small number of false alarms, caused by differences in the test procedures between Shuttle and Ares I-X. Most of these false alarms were caused by incorrect mode identification. For example, some tests were performed in a different order for Ares I-X than they were for Shuttle. Prior to the launch, we fixed the SHINE rules for mode identification. Shortly before the launch, the prototype had a small number of false alarms caused by data dropouts. We had expected data dropouts during ascent, but had not expected data dropouts before launch. After the launch, we modified the wrapper code to detect data dropouts. After making this modification, we were able to run the prototype on the recorded data from the launch without any false alarms. Ares I-X did not have any failures in the systems we modeled, so the prototype had no correct detections and no missed detections.

V. Anomaly Detection

We ran IMS in parallel with the TEAMS/SHINE combination. We expected that TEAMS and SHINE would detect all of the known failure modes that we modeled, while IMS would have the potential to detect unknown failure modes and anomalies that are not yet failures. IMS has been proven as a mature tool by supporting various NASA Johnson Space Center (JSC) Shuttle and International Space Station (ISS) operations for four years as well as being licensed commercially for three years. It is currently running on a console at JSC Mission Control Center to monitor live data from the ISS Control Moment Gyroscopes, Early External Thermal Control System, and Rate Gyroscopic Assembly, and has been certified as Class C software for those three systems.

The tool works under the principle of a one-class modeling algorithm by building a model of the nominal historical data on which it is trained. Because IMS only models the nominal data, and does not model any failure modes, it can potentially detect unknown failure modes. The model takes the form of a knowledge base (KB) of clusters. Once the KB has been learned, unseen data points are evaluated against the KB and assigned anomaly scores based on how different the data points are from the training data. If a new point falls within an existing cluster, then it is assigned an IMS score of zero. If it does not fall within an existing cluster, then the distance to the nearest cluster is used as the IMS score. IMS also calculates a contributing score for each measurement, representing each measurement's contribution to the overall IMS score. When an anomalous period of the testing data is localized, these contributing IMS scores can be used to help diagnose the issue.

Prior to the Ares I-X launch, we trained IMS on historical Space Shuttle data, and tested it using historical Shuttle data into which we had inserted simulated failures. During the Ares I-X pre-launch period, IMS processed live Ares I-X data, using the knowledge base that was the result of training IMS on historical Shuttle data. The remainder of this section describes the selection of measurements for use with

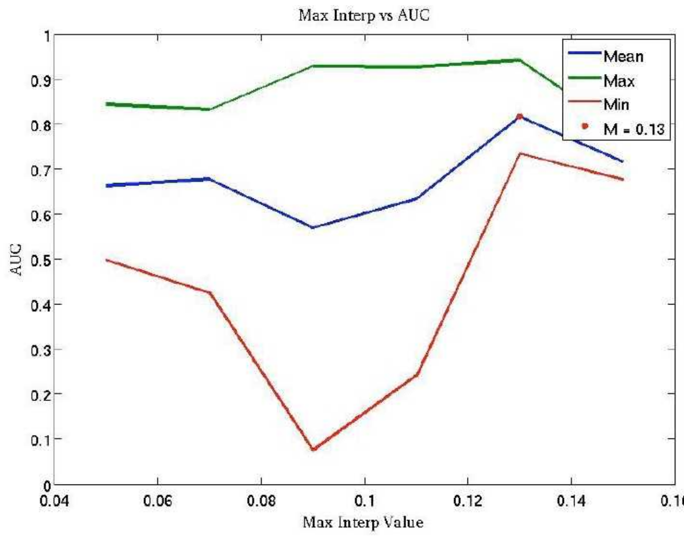


Figure 3. AUC as a function of IMS Parameter Max Interp for Shuttle data from the VAB. *The red dot indicates the optimal value.*

IMS, the training and testing procedures used, and the results obtained both on Shuttle data and on Ares I-X data. The section concludes with a summary of the results.

A. Training and Testing Procedures

For the purpose of training and testing IMS, we used historical Space Shuttle data into which we inserted simulated failures. Although the main purpose of using IMS in GDP is to detect unknown failures, we tested it by using simulations of known failures.

IMS has a number of tunable input parameters. One key parameter that was very important to tune was the maximum interpolation (max interp) parameter. This parameter governs the threshold in the learning phase that determines if a new data point should be placed in the current cluster or used to generate a new cluster. The parameter directly influences the number of clusters created in the learning phase and therefore has a major influence in the final anomaly score calculated by IMS. As the max interp value increases the total number of clusters formed becomes smaller.

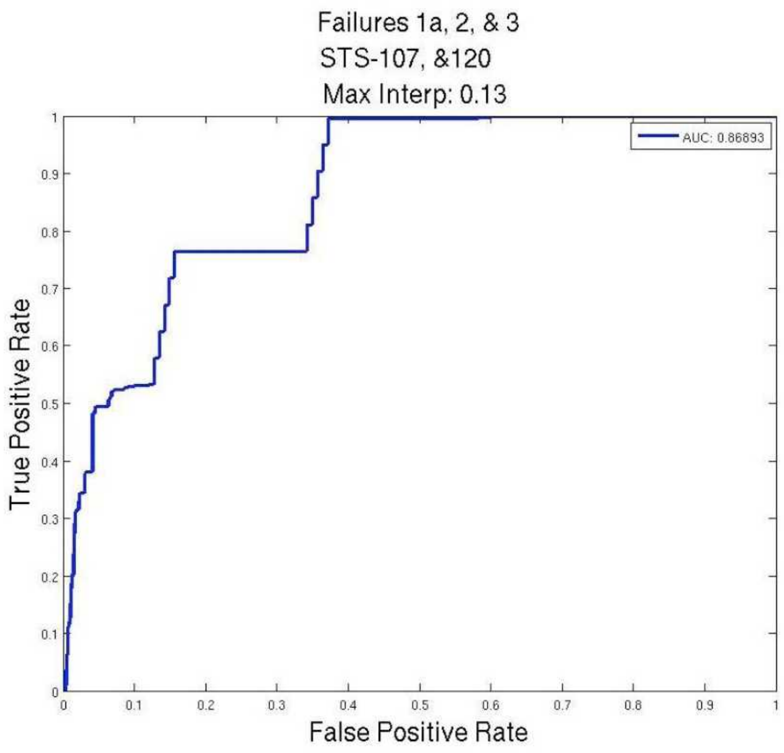


Figure 4. ROC Curve for Shuttle data from the VAB. *This ROC curve was generated using the optimal max interp value from Figure 3.*

To determine the optimal max interp value and corresponding number of clusters a set of cross validation runs was performed on a set of Shuttle VAB and launch pad data, using the Area under the ROC (Receiver Operating Characteristic) curve (AUC) as the governing metric for optimization. Cross validation is a technique for estimating the accuracy of a machine learning algorithm, by training and testing the algorithm multiple times, each time using different subsets of the available data for training and testing, and then averaging the results. The ROC curve is a plot of true positive rate against false positive rate, and can be used to help make the tradeoff between these two rates. The AUC is loosely a measure of accuracy over all possible tradeoffs between true positive rate and false positive rate. More formally, the AUC represents the probability that a randomly chosen failure data point is more suspect than a randomly chosen nominal data point.[6] An AUC of one thus indicates perfect ranking of these two randomly selected data ponts.

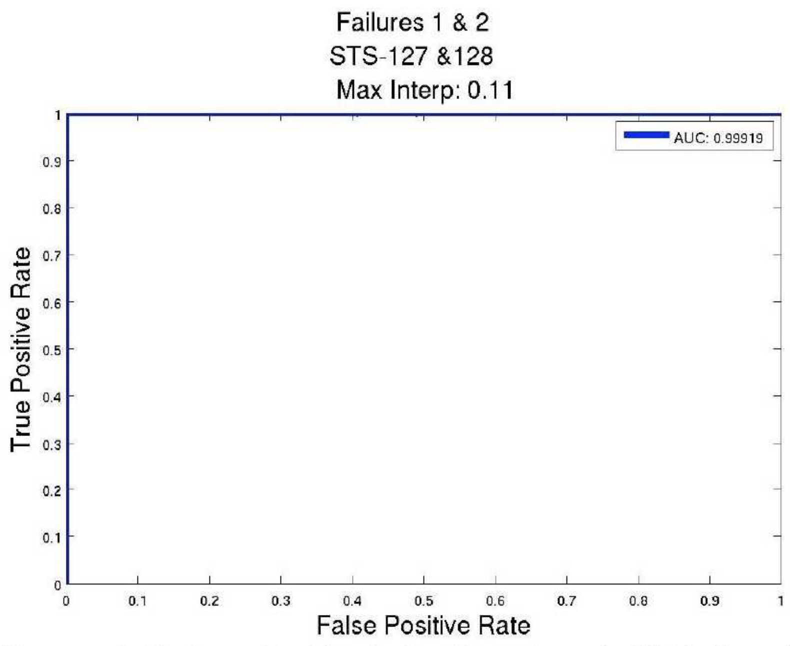


Figure 5. ROC Curve for Shuttle data from the pad. *IMS had nearly perfect accuracy on the Shuttle pad data.*

B. Results on Shuttle Simulations

Once the cross validation runs were complete, the areas under the

ROC curves were calculated. Figure 3 shows the maximum, minimum, and average AUC over the three-fold cross validations and three fault scenarios for each max interp value. The optimal max interp value that was chosen is marked in the plots. The ROC curve with the optimal max interp value of 0.13 can be seen in Figure 4. The AUC is only 0.86893, because IMS had difficulty detecting one of the three simulated failures that were used. The increase in IMS score resulting from this simulated failure was not much larger than the nominal variation in the IMS score, so it was not possible to select a threshold that would allow IMS to detect all of the simulated failures without having any false alarms. Some failure modes are easily detected using IMS' distance-based approach with clustering, while others are not. When IMS is used in parallel with TEAMS-RT, TEAMS-RT should detect all of the failures that are modeled in the TEAMS model; the advantage of using IMS in addition is that it has the potential to detect failures that were not modeled, as well as anomalies that are not yet failures. For the pad, IMS performed much better. After optimizing the value of max_interp, the ROC curve was generated and can be seen in Figure 5. Here the AUC is 0.99919, indicating that IMS does an excellent job of detecting the two simulated failure modes at the pad.

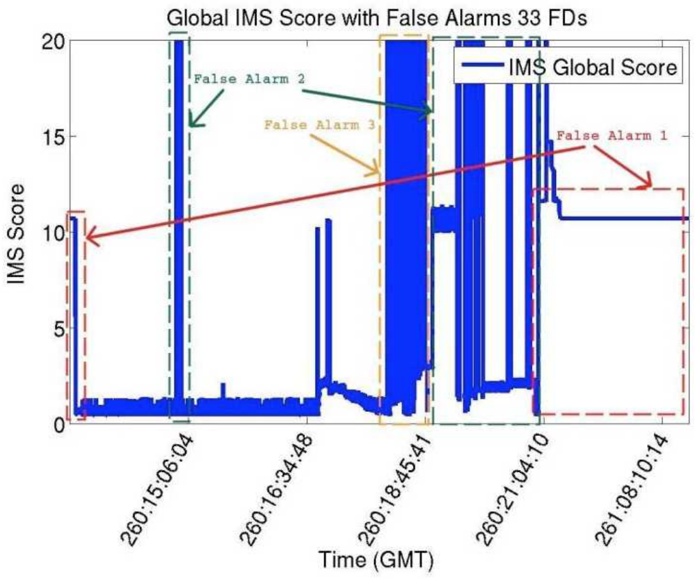


Figure 6. Ares I-X Sept 17th VAB Global IMS Score. *Periods during which IMS detected anomalies that were not failures are labeled as "false alarms"*

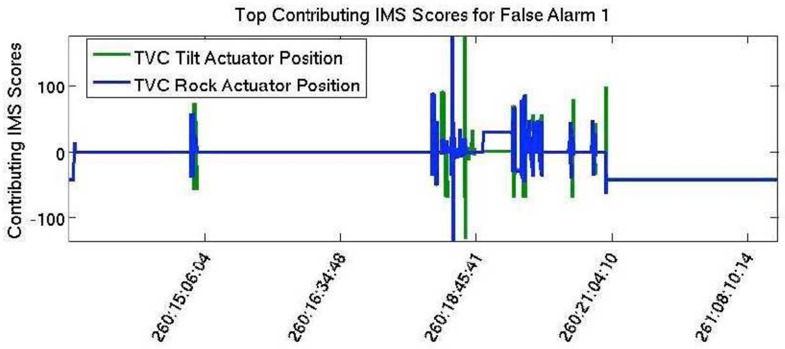


Figure 7. Sept 17th VAB Top Contributing IMS Scores For False Alarm 1. *False Alarm 1 was primarily caused by the TVC rock and tilt actuator positions, which moved through a larger range of motion on Ares I-X than they did on Shuttle.*

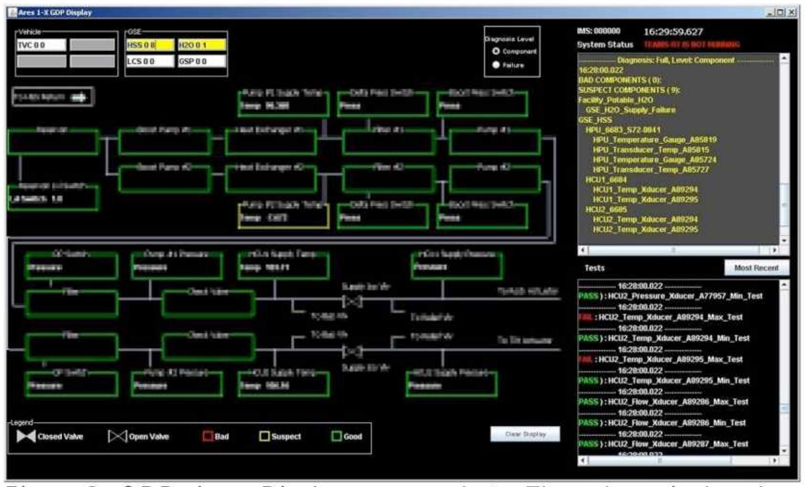
C. Results on Ares I-X

Once the optimal max interp parameter was determined from the Shuttle data, IMS was trained on 33 measurements using Shuttle data from seven flights. After building the KB, the Ares I-X data was evaluated against it. The resulting IMS scores for the VAB are shown in Figure 6. With the initial set of 33 measurements, 3 periods of anomalous behavior were flagged by IMS; they are labeled as three "False Alarms" in Figure 6. We performed an analysis of each "false alarm"; here we present the analysis of False Alarm 1 as an example. We determined that False Alarm 1 was primarily caused by two measurements, the TVC rock and tilt actuator positions. The contributing IMS scores for these two measurements are plotted in Figure 7.

False Alarm 1 was caused by a difference between the Space Shuttle and Ares I-X. In recent years, the TVC actuator tests performed in the VAB have all been "pinned" tests, meaning that the actuator is physically pinned to the nozzle during testing, so that the nozzle moves during the test. The first TVC actuator position test performed in the VAB for Ares I-X was an "unpinned" test, meaning that the actuator was detached from the nozzle, and the nozzle did not move during the test. Because the actuator was unpinned, it was able to move through a larger range of motion that is not possible during pinned testing. IMS therefore saw rock and tilt position values that it had never seen in the Shuttle data, which it flagged as anomalies. These anomalies are "false alarms" in the sense that they are not failures, but they do illustrate the ability of IMS to detect new data that is different from what it has seen before. We performed a similar analysis for the launch pad, where there were fewer anomalies identified by IMS. Like the anomalies detected at the VAB, the anomalies detected at the launch pad were caused by operational differences between Shuttle and Ares I-X.

D. Summary of IMS results

The experiments that we ran before the Ares I-X launch using historical Space Shuttle data with simulated failures demonstrated that IMS is able to detect most of the simulated failures, but not all of them. In particular, it had difficulty detecting one of the simulated failure modes in the VAB. That is not surprising. IMS is not trained to detect specific failure modes; it detects data that is anomalous according to its cluster-based model. We expect that many known and unknown failure modes will be detected as anomalies by IMS, but it is not guaranteed to detect all possible failure modes. The advantage of using IMS together with a model-based diagnosis system such as TEAMS is that it adds the potential to detect unknown failure modes and to detect precursors of failures.



```
IMS: 000000     16:29:59.627
System Status   TEAMS-RT IS NOT RUNNING
------------ Diagnosis: Full, Level: Component ------------
16:28:00.022
BAD COMPONENTS ( 0):
SUSPECT COMPONENTS ( 9):
Facility_Potable_H2O
  GSE_H2O_Supply_Failure
GSE_HSS
  HPU_6683_S72-0841
    HPU_Temperature_Gauge_A85819
    HPU_Transducer_Temp_A85815
    HPU_Temperature_Gauge_A85724
    HPU_Transducer_Temp_A85727
  HCU1_6684
    HCU1_Temp_Xducer_A89294
    HCU1_Temp_Xducer_A89295
  HCU2_6685
    HCU2_Temp_Xducer_A89294
    HCU2_Temp_Xducer_A89295
```

```
------------ 16:28:00.022 ------------
PASS ) : HCU2_Pressure_Xducer_A77957_Min_Test
------------ 16:28:00.022 ------------
FAIL : HCU2_Temp_Xducer_A89294_Max_Test
------------ 16:28:00.022 ------------
PASS ) : HCU2_Temp_Xducer_A89294_Min_Test
------------ 16:28:00.022 ------------
FAIL : HCU2_Temp_Xducer_A89295_Max_Test
------------ 16:28:00.022 ------------
PASS ) : HCU2_Temp_Xducer_A89295_Min_Test
------------ 16:28:00.022 ------------
PASS ) : HCU2_Flow_Xducer_A89286_Max_Test
------------ 16:28:00.022 ------------
PASS ) : HCU2_Flow_Xducer_A89286_Min_Test
------------ 16:28:00.022 ------------
PASS ) : HCU2_Flow_Xducer_A89287_Max_Test
```

Figure 8. GDP Java Display screen shot. *The schematic has been blurred to protect confidentiality.*

The results of running IMS on Ares I-X data, using a KB that was trained on historical Space Shuttle data, confirm our hypothesis that the Ares I-X TVC data is reasonably similar to the Space Shuttle SRB TVC data. Most of the time, IMS produced small anomaly scores when run on the Ares I-X data. IMS did detect some "anomalies" in the Ares I-X data. These anomalies were "false alarms" in the sense that they were not failures but rather caused by operations performed differently for Ares I-X versus Shuttle; hence, they illustrate the ability of IMS to detect new data that is different from what has been seen in the past.

VI. Java Display

Java Display is an application that is used to display in real time the following information:

1. An interactive schematic of the HSS and TVC subsystems
2. Values for sensors present in the HSS and TVC subsystems
3. Outcome of tests performed by the wrapper code (SHINE)
4. Diagnostic results of TEAMS at both the component and failure levels
5. A tally of suspect and bad components both at the component and failure levels
6. The IMS score

A screen shot of the Java Display is shown in Figure 8.

The two scrollable text areas on the right hand side display information pertaining to TEAMS diagnoses and SHINE wrapper code test results. On the top right hand corner of the display are fields that display the IMS score and the current time. A large part of the Java Display is dedicated to a schematic representation of the HSS and TVC subsystems. The schematic representation is hierarchical and the user is provided with the ability to click on a particular subsystem to view details at a component level. The subsystem schematics also display in real time the associated sensor values. Each component of the schematic is colored with an outline depending on the outcome of the diagnosis from TEAMS. In particular, good, suspect and bad components are colored green, yellow and red respectively.

VII. Computational Performance of the Prototype

The prototype performed very well from a computational perspective. During development and testing, we ran the prototype on a Dell Precision M4400 laptop with an Intel Core 2 Quad Q9300 CPU running at 2.53 GHz and 4 GB of DRAM, running Windows XP 32-bit. Table 1 shows the memory and CPU usage of each of the processes within the prototype, when running on the laptop with historical Space Shuttle data and a simulated fault ("Hydraulic pump over-temperature failure") fed into it at simulated real-time (25 Hz). The CPU usage numbers indicate what percentage of the quad-core CPU was used. So, for example, a process that used all of the CPU time on one core would be listed as 25%.

Table 1: GDP memory and CPU usage

Process	CPU	DRAM
TEAMS (includes TEAMS-RT, the SHINE rules, the C test logic, and the data interface code)	8%	12 MB
IMS (including its data interface code)	1%	11 MB
Java display (including JVM)	1%	29 MB
Data playback software	18%	56 MB
Plotting tool	1%	12 MB
Windows XP Operating System	5%	410 MB
Total	34%	530 MB

In summary, the prototype could be run on a single laptop PC using only a small fraction of the available CPU and memory. It could have easily run on a PC with half as many cores and half as much memory. It used a substantial size model (with 893 failure modes), and ran at 25 Hz. We expect that future NASA ground diagnostic applications will receive data at a much lower rate (possibly 1 Hz), but will model dozens of systems (compared with the two systems modeled in our prototype).

VIII. Verification, Validation, and Certification

Ares I-X GDP is a prototype. It was not certified, and as such its outputs cannot be used to make operational decisions. The team did, however, seek to develop the tool in such a way that it could be certified. The three tools that were integrated (TEAMS-RT, IMS, and SHINE) were selected based in large part on their potential to be certified. These three tools were judged to be easier to certify than competing tools because of their past usage in relevant applications and because of their relative simplicity. Decisions regarding how the vehicle and ground models would be integrated were also made based in part on ease of certification. The team wrote a draft verification and validation plan that describes how we would have verified, validated, and certified the prototype if it had been deployed as operational software.

In lieu of a formal verification and validation process, the team performed informal testing of the prototype and its components. Much of testing used historical Shuttle data into which we inserted simulated failures. In addition, the team performed testing of the interface code by running large quantities of historical Shuttle data through the different pieces of interface code and verifying that the data that came out of each interface matched the data that went into each interface. Before deploying the prototype to Hangar AE at KSC, the team performed integrated testing of the entire prototype (including TEAMS-RT, SHINE, IMS, and the Java display) using historical data from several Shuttle flights. Shortly after the initial power-up of Ares I-X in the VAB, the prototype was tested using recorded Ares I-X data from the initial power-up. Finally, after the prototype was installed in Hangar AE, it was tested using live Ares I-X data from the VAB.

IX. Lessons Learned

We learned many lessons from our experience of developing and deploying the prototype. We expect that these lessons will be useful to future NASA efforts to build automated diagnostic systems, and may also be useful to people beyond NASA. The most important lessons are summarized in this section.

A. Need for the system to provide system mode and event information

TEAMS-RT requires mode and event information as part of its inputs (since certain failure modes can only occur in certain system modes). The system modes that we needed were the major system modes of the vehicle, specifically VAB or pad (and were easy to infer). The events and subevents were generally indications of what test was being performed or what step within a test was being performed. These were often more difficult to infer. If the system gets into the wrong mode or event, it is unlikely to ever recover automatically. When GDP got into the wrong mode or event, we stopped the software, manually set the mode and event, and then restarted it.

The Space Shuttle and Ares I-X both provided us with inadequate system mode and event information. Because of that, we needed to develop complex SHINE rules to infer the system mode and events from the sensor data and the command stream. That caused five problems:

1) The development of these SHINE rules was very labor intensive, which resulted in increased costs.

2) The complexity of these SHINE rules increased the risk that an error in the SHINE rules would result in a false alarm. (And in fact, we did have false alarms caused by errors in these SHINE rules during the Ares I-X pre-launch period. We have since fixed all of the errors of which we are aware.)
3) We faced the risk that if any of the data on which these SHINE rules depended was missing or incorrect, the prototype would infer the system mode or event incorrectly, resulting in false alarms.
4) Certain failure modes could be mistakenly interpreted as mode changes, resulting in both missed detections and false alarms. For example, suppose an event that needs to be detected is a valve being closed manually by a person turning a knob. Suppose that the SHINE rules use a pressure sensor downstream of the valve to detect when the valve has been closed. If there is a system failure that results in a decrease in pressure, such as a leak, then the SHINE rules will mistakenly conclude that the valve has been closed.
5) There were some events that we determined could not be detected using the available sensor data and command stream. We therefore had to remove from our scope the failure modes that depended on these events.

We believe it is very important for future launch vehicles, such as Ares I, to transmit system mode and event information to the diagnostic software.

B. Cost of developing wrapper code

In addition to developing the TEAMS models and the IMS KBs, the GDP team also needed to develop "wrapper code." The wrapper code consists of:

1) The interface code that gets live data and feeds it to TEAMS-RT, the SHINE rules, and IMS
2) The system mode and event identification code
3) The "pass/fail" test logic.

The cost of developing the "wrapper" code was higher than we had expected it to be. We estimate that the cost of developing the wrapper code was comparable to the cost of developing the TEAMS models. We expect that the relative cost of developing wrapper code will be lower in the future, for two reasons:

1) We hope that future systems will provide better mode and event information, reducing the cost of developing system mode and event identification code (see Lesson A).
2) A significant part of the cost of developing the wrapper code was the cost of developing the interface code. This interface code is not specific to the systems we modeled. Since we only modeled two systems (TVC and HSS), we were only able to amortize this cost over these two systems. We expect that future diagnostic systems will model more systems, and will therefore be able to amortize the cost of the interface code over more systems.

C. Utility of using rule-based systems for wrapper code

The use of SHINE in development of the wrapper code had several upsides but also many downsides. Once the basic syntax of coding in SHINE was understood, the SHINE framework allowed for very fast development of rules for event mode detection. After the rules were compiled into C code, the forward-chaining aspect of SHINE greatly simplified what would otherwise have been a complicated series of nested-if statements. Unfortunately there were a number of problems as well that may have been avoided if SHINE had not been used, or if it had been used in a more limited context. The biggest downside turned out to be the difficulty of debugging the SHINE rules. This could have been improved upon in a number of ways:

1) Begin testing SHINE rules on smaller datasets.
2) Break the SHINE code into smaller, more manageable "chunks".
3) An Integrated Development Environment (IDE) for SHINE could potentially have helped to debug the rules prior to testing on real data. It could have enhanced the visibility of what rules are associated and triggered by certain variables, and also helped to spot potential "infinite loops".
4) Move all TEAMS tests outside of SHINE. While SHINE was fully capable of evaluating TEAMS test logic, most of these tests were more easily coded and managed using table lookups.

Overall, SHINE proved to be a useful but cumbersome tool. If future work were to be done in this area, SHINE could certainly help but it is recommended to be used in some combination with straightforward C/C++ programming to minimize complexity. In general, the things that can easily be done in C or C++, such as table lookups for threshold values, should be done in C or C++. SHINE rules should only be used for tasks that naturally fit the rule-based paradigm, such as some of the more complex event identification rules.

D. TEAMS modeling lessons learned

The process of modeling the TVC and HSS in TEAMS and integrating the two models has resulted in several lessons learned. First, our experience highlights the need for a comprehensive Interface Control Document (ICD) for each subsystem that is modeled in TEAMS-Designer. Without an ICD that is agreed upon prior to the start of the modeling tasks, the model designer will likely spend many hours modifying the models to allow the proper propagation of information between interacting subsystems.

Second, the testability analysis of the individual models revealed the disparity between the usefulness of the fault isolation for different types of subsystems. There is better fault isolation in the TVC model than the HSS as a result of how the subsystems have been designed and are operated. (Flight systems generally have more instrumentation than ground systems, resulting in better fault isolation.) In the future, criteria related to the degree of automation in the system along with instrumentation placement may aid in deciding which subsystems should be modeled for the greatest benefit to an automated fault detection and isolation application.

Finally, the model integration process between the TVC and HSS demonstrated the need for a common approach regarding the use of functions and test points. There are several strategies for naming and using functions in the context of local and global failure propagation, bidirectional flow, switches, and function mapping. A set of modeling conventions for TEAMS has been developed which addresses common look and feel considerations for the model, but the conventions have so far left function naming and usage to the discretion of the modeler. In the future, a standard convention for the usage of functions and test points should be developed, and all ground and vehicle models intended to be used in an integrated fashion to perform fault isolation should be required to follow the new convention.

E. Appropriate roles of model-based diagnosis and anomaly detection

Each tool should be leveraged to promote its strengths rather than re-adapting the tool to solve a problem that is outside of its domain of relevance. Model-based systems such as TEAMS work very well for detecting and diagnosing the known failures that have been modeled. As we mentioned in the Introduction, we believe that the value of including a one-class anomaly detection algorithm such as IMS alongside a model-based diagnosis system such as TEAMS in a diagnostic system is that the anomaly detection system has the potential to detect anomalies that cannot be detected by the model-based diagnosis system, either because they are unknown failures and therefore unmodeled, or because they are not failures. Furthermore, IMS may detect known failures in advance of the time that TEAMS would detect them, and in general IMS requires less modeling effort than TEAMS (although it does require a sufficient quantity of historical and/or simulated training data). But anomaly detection methods such as IMS are not well suited for detecting some types of failures. As we mentioned in the IMS section, we used simulations of known failure modes to test IMS. For some of these simulated failures, we expended a lot of effort in tuning IMS to get IMS to detect the simulated failures.[7] This tuning process included reducing the set of measurements that were used to train IMS.

With IMS, we know that its strengths lie in a great potential to detect faults that are unknown or that otherwise have not been modeled and to detect anomalies that are precursors of faults before a model-based system detects the fault. We believe that it would be better to rely on TEAMS to detect certain known failure modes, rather than tuning IMS to detect them. Reducing the set of measurements that are used to train IMS did allow IMS to successfully detect the simulated failures, but it reduced IMS' potential to detect other unknown failures.

F. Metrics

We had a lot of discussions about the appropriate metrics for measuring the accuracy of diagnosis and anomaly detection. We considered different ways of defining false positives and false negatives in both cases. We ended up using the Area under the ROC curve (AUC) to measure the performance of IMS and enumerating the false alarms to assess the performance of TEAMS. Any future operational deployment of diagnostic technology will require careful consideration of metrics.

X. Conclusion

Automated pre-launch diagnostics can help increase safety, reduce cost, and reduce launch delays. The Ares I-X Ground Diagnostic Prototype helped to demonstrate and mature automated fault detection and diagnostic software that can be used in future missions. GDP successfully demonstrated the feasibility of integrating three very different fault detection and diagnostic methods, and of integrating diagnosis of the vehicle with diagnosis of the ground systems.

Although the prototype had a small number of false alarms in TEAMS, we believe that an operational system could have avoided these false alarms by having mode identification provided by the system, and by having a formal verification and validation process. As an anomaly detection system, IMS can be expected to have some false alarms (since not all anomalies are failures), but we expect that the number of false alarms will decrease over time as more data becomes available for training IMS.

Acknowledgments

We thank all current and past members of the Ares I-X GDP team, including Charles Lee (SGT at ARC), Vijay Baskaran (SGT at ARC), Pedro Vazquez (ASRC at KSC), Peter Robinson (ARC), Robert Trent (ARC), Mark James (JPL), Pat Chang (JPL), John Wallerius (formerly QSS at ARC), Larry Markosian (SGT at ARC), Martin Feather (JPL), Tim Taylor (Analex at MSFC), Andrew Riddle (Wright State University), and Gloria Floyd (formerly Miltec at KSC).

We thank the people who have helped us interact with our intended future customers and with other related tasks, including Barbara Brown (ARC at KSC), Stephen Johnson (MSFC), and Mike Watson (MSFC).

We thank Nate Wood (KSC) for helping us interface our prototype with the WinPlot Archive Server in Hangar AE.

We thank the domain experts who helped us to obtain relevant data and to understand the systems that we modeled, including Roland Lebon (USA at KSC), Ben Hayashida (MSFC), and Jim Herndon (KSC).

We thank Dave Iverson (ARC) for providing IMS and for helping us with IMS.

We thank Somnath Deb, Sudipto Ghoshal, Venkat Malepati, Chuck Domagala, and Mohammed Azam of QSI for their help with TEAMS.

We thank the people who have provided useful feedback during discussions about the prototype, including Lilly Spirkovska (ARC), David Hall (SGT at ARC), Bob Ferrell (KSC), and Ryan Mackey (JPL).

We thank Lilly Spirkovska (ARC) for designing the concept and the human factors elements of the GDP Java Display.

We thank the managers who helped get the task started and who helped guide it, including Serdar Uckun (formerly ARC), Ann Patterson-Hine (ARC), David Korsmeyer (ARC), Dwight Sanderfer (ARC), Dougal Maclise (ARC), and George Sarver (ARC).

We thank Lilly Spirkovska and Ann Patterson-Hine for reviewing a draft of this paper and providing valuable feedback.

Ares I-X GDP was funded by the NASA Constellation Program, by the NASA Exploration Technology Development Program, and by the NASA KSC Ground Operations Project.

References

[1]Davis, S. R., "Ares I-X Flight Test - The Future Begins Here," *AIAA SPACE Conference*, 2008.

[2]James, M. and Atkinson, D., "Software for Development of Expert Systems," NASA Technology Briefs, vol. 14, no. 6, 1990.

[3]D. L. Iverson, R. Martin, M. Schwabacher, L. Spirkovska, W. Taylor, R. Mackey, and J. P. Castle, "General Purpose Data-Driven System Monitoring for Space Operations," *AIAA Infotech@Aerospace Conference*, 2009.

[4]M. Schwabacher and R. Waterman, "Pre-Launch Diagnostics for Launch Vehicles," *IEEE Aerospace Conference*, 2008.

[5]R. Martin, M. Schwabacher, and B. Matthews, "Investigation of Data-Driven Anomaly Detection Performance for Simulated Thrust Vector Control System Failures ," In *Proceedings of the 57th Joint Army-Navy-NASA-Air Force Propulsion Meeting*, Colorado Springs, CO, May 2010. To appear.

[6]Bradley, A. P. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145 – 1159, 1997.

[7]Rodney A. Martin. "Evaluation of Anomaly Detection Capability for Ground-Based Pre-Launch Shuttle Operations", chapter in *Aerospace Technologies Advancements*, pages 141–164. IN-TECH, January 2010.