MSFC-973

The Marshall Space Flight Center Fault Detection Diagnosis and Recovery Laboratory

Bradley T. Burchett[*]
Rose-Hulman Institute of Technology, Terre Haute, IN, 47803

Jonathan Gamble[†]
Analex Corporation, Huntsville, AL, Zip Code

and

Michael Rabban[‡]
Miltec Corporation, Huntsville, AL, 35806

The Fault Detection Diagnosis and Recovery Lab (FDDR) has been developed to support development of fault detection algorithms for the flight computer aboard the Ares I and follow-on vehicles. It consists of several workstations using Ethernet and TCP/IP to simulate communications between vehicle sensors, flight computers, and ground based support computers. Isolation of tasks between workstations was set up intentionally to limit information flow and provide a realistic simulation of communication channels within the vehicle and between the vehicle and ground station.

I. Introduction

THE Fault Detection, Diagnosis and Recovery laboratory (FDDR) is a unique computing environment dedicated to the development, testing and validation of fault detection algorithms. Its primary objectives are to facilitate the development and evaluation of FDDR algorithms for use on the Ares I launch vehicle system and ground systems supporting the Ares I launch vehicle, and to evaluate advanced FDDR algorithms, technologies, techniques, sensors and instrumentation for use with other mission systems intended for travel to the moon and beyond. Within each objective, FDDR algorithms provide three capabilities to satisfy vehicle requirements: 1) Detect failures and abort conditions during flight or mission operations, 2) perform pre flight ground diagnostics and checkout, and 3) perform post flight ground diagnostics and mission analysis.

The Ares I launch vehicle and Orion spacecraft have been designed to allow safe recovery of the crew throughout the launch. Davidson et al[1] provide a summary of the abort strategies intended for the ascent phase.

Figure 1 provides an overview of the FDDR lab software architecture. The three large rectangles labeled Vehicle Simulator, Flight Emulator, and Ground Emulator represent separate workstations as well as specific software tasks. These components were designed to operate in 'real time' using Red Hat Linux Each software is described in detail in the sequel. The Scenario Library / Scenario Generator is essentially a file system and infrastructure software allowing the user to easily reformat and splice sensor data from multiple source as necessary to stimulate the algorithms intended for testing. The scenario generator currently runs independently of the other software.

II. Scenario Library and Scenario Generator

The purpose of the Scenario Generator is to develop the datasets containing the nominal and failure sensor data for the Ares-I vehicle. Simulated sensor data may come from several sources including flight data from legacy systems similar to the Ares I, existing vehicle simulations of varying fidelity, and models specifically developed for this task.

[*] Associate Professor, CM 147, 5500 Wabash Ave., Associate Fellow. AIAA.
[†] Insert Job Title, Department Name, Address/Mail Stop, and AIAA Member Grade for third author.
[‡] Systems Engineer, 689 Discovery Drive, MRabban@miltecsystems.com

The scenario library will contain representative data for the performance of important sensors during nominal and abort situations. A finite set of monitored abort scenarios has been defined by NASA. The FDDR group is tasked with finding or developing models of these scenarios for upper stage subsystems not including the engine. Nominal and borderline behavior is provided to test the algorithm for false positive diagnosis. A companion paper describes the development of a representative model for the main propulsion system[2].

The Ares I vehicle will have over 5000 sensors, including vehicle guidance and navigation sensors, subsystem performance and health sensors, such and tank pressures, battery temperatures, etc., and status measurements such as tank float switches, current valve positions, etc. Simulating all of these signals would be an insurmountable and unnecessary task. The Scenario generator allows the developer to choose which signals will be included in a scenario, and splice together signals from multiple sources.

A. Scenario Generator Software

Include specifics about the scenario generator software here (language, interface, anything novel, etc.)

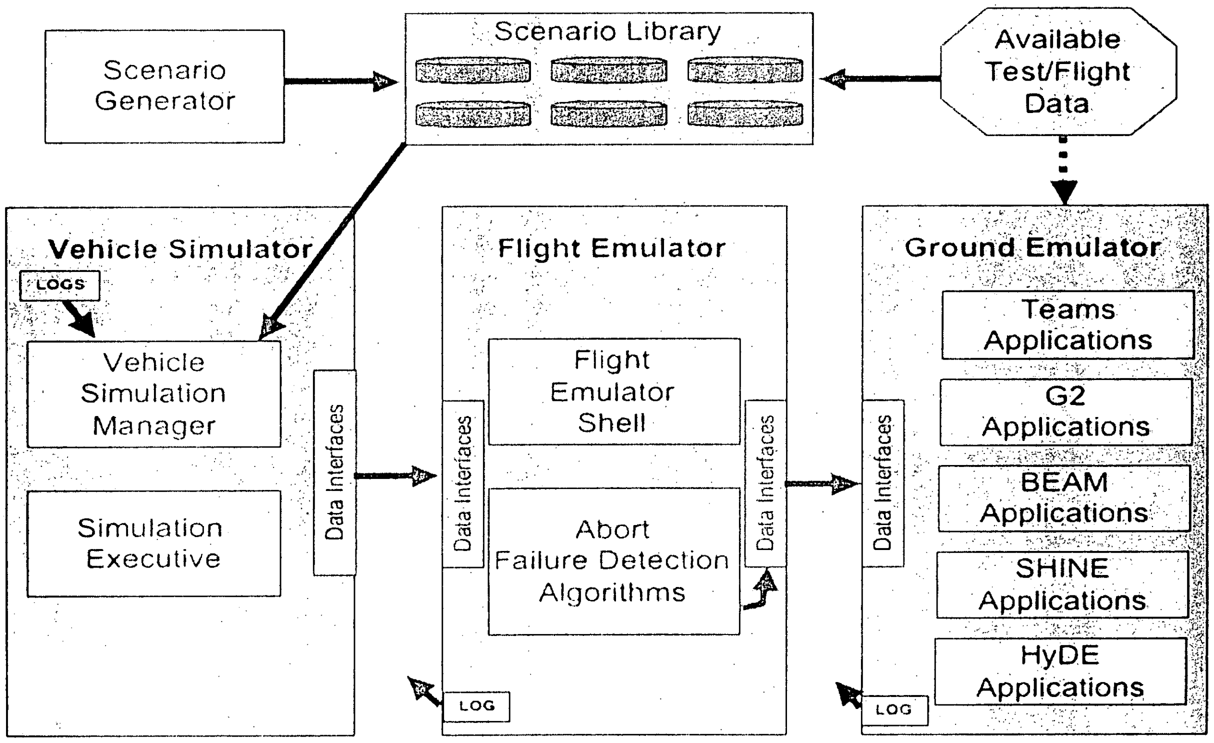

Figure 1: FDDR Software Overview

III. Vehicle Simulator

The vehicle simulator controls the simulation by encoding sensor data, and passing it sequentially to the flight emulator.

A. Vehicle Simulator Software

The Vehicle simulator is a Windows application developed in Java? *Need lots of help here*

B. Vehicle Simulator Interface

Really not sure what the next subheading should be, how many sub-sections do we need?

IV. Flight Emulator

The Flight Emulator simulates the flight computer tasks of listening for sensor data, and reacting by executing the active detection algorithms. It is also expected to relay sensor data and diagnoses to the ground emulator(s).

The flight emulator was written in C, and is implemented in the Linux OS.

More specifics about the flight emulator here.

V. Ground Emulator

A ground emulator shell was developed in C

Multiple ground emulators may be run simultaneously on one or more workstations as depicted in Figure 1. Several NASA developed fault detection applications may be tested including HyDE, TEAMS, etc.

VI. Conclusion

Appendix

An appendix, if needed, should appear before the acknowledgements.

Acknowledgments

References

[1]Davidson, J., Madsen, J., Proud, R., Merritt, D., Raney, D., Sparks, D., Kenyon, P, Burt, R., and McFarland, M., "Crew Exploration Vehicle Ascent Abort Overview," *AIAA Guidance, Navigation and Control Conference and Exhibit*, 20 - 23 August 2007, Hilton Head, South Carolina.