

The Ares I Abort Failure Detection, Notification, and Response System: An Overview of the Development Process

Greg Pisanich
Perot Systems Government Services
NASA Ames Research Center
MS 269-3
Moffett Field, CA 94035
650 604 5334
Gregory.M.Pisanich@nasa.gov

Anupa Bajwa
University of California
NASA Ames Research Center
MS 269-4
Moffett Field, CA 94035
650 604 1851
Anupa.R.Bajwa@nasa.gov

Dwight Sanderfer
Intelligent Systems Division
NASA Ames Research Center
MS 269-3
Moffett Field, CA 94035
650 604 5334
Dwight.T.Sanderfer@nasa.gov

Abstract—Timely detection and response to catastrophic events during the launch and ascent phase of the Ares I launch system is of paramount importance to crew safety. This requires an abort system capable of detecting and confirming conditions that may lead to catastrophic failure, notifying the crew of the problem, and responding in time to allow the crew to escape safely.

The Ares I Abort Failure Detection, Notification, and Response System is being developed through an iterative approach that analyses the vehicle design and identifies potential abort conditions, characterizes those conditions through modeling and simulation, then uses this data to develop algorithms capable of detecting and alerting the crew in time. It is expected that this same process will be applied in the development of future NASA crewed vehicles. It should also be applicable to other complex vehicle systems.^{1 2}

CONTENTS

1. INTRODUCTION.....	1
2. UNDERSTANDING THE ABORT DEVELOPMENT CHALLENGES	2
3. IDENTIFICATION AND CLASSIFICATION OF ABORT CONDITIONS	3
4. CHARACTERIZATION OF ABORT CONDITIONS	5
5. ABORT ALGORITHM DEVELOPMENT	7
6. ITERATIVE REVIEW DURING DEVELOPMENT	9
7. SUMMARY	9
REFERENCES	9
ACKNOWLEDGEMENTS	9
AUTHORS	10

1. INTRODUCTION

NASA has embarked on a return to the Moon and eventually to Mars. As a first step, NASA’s Constellation Program is developing a new launch vehicle (Ares I) and crew capsule (Orion). This system will initially take astronauts into Earth orbit to visit the International Space Station and eventually to rendezvous with other Constellation vehicles that will take them to the Moon [1].

Getting to the Moon requires the development of several Constellation systems, including the Ares V heavy launch booster, Earth Departure Stage (EDS), and a Lunar Surface Access Module (LSAM). All of these complex systems will require the development of Integrated Vehicle Health Management (IVHM) systems. In addition to the in-flight diagnostics and response, ground-based diagnostics and fleet supportability applications are also needed.

Arguably, the most critical application is the in-flight abort of Ares I and Orion—the capability to detect and confirm conditions that may lead to catastrophic failure, notify the crew of the problem, and respond in time to allow the crew to escape safely.

This paper describes the process being used to develop the Ares I Abort Failure Detection, Confirmation, and Response System. It initially presents some of the Ares I and Orion system and physical requirements that are driving the development of this onboard system. It then outlines the development process, and provides details for each of the development steps: classification, characterization, and algorithm development. Finally, it provides an overview of the iterative process being followed, and outlines some of the analyses required.

¹ 1-4244-1488-1/08/\$25.00 ©2008 IEEE.

² IEEEAC paper #1404, Version 5, Updated December 7, 2007

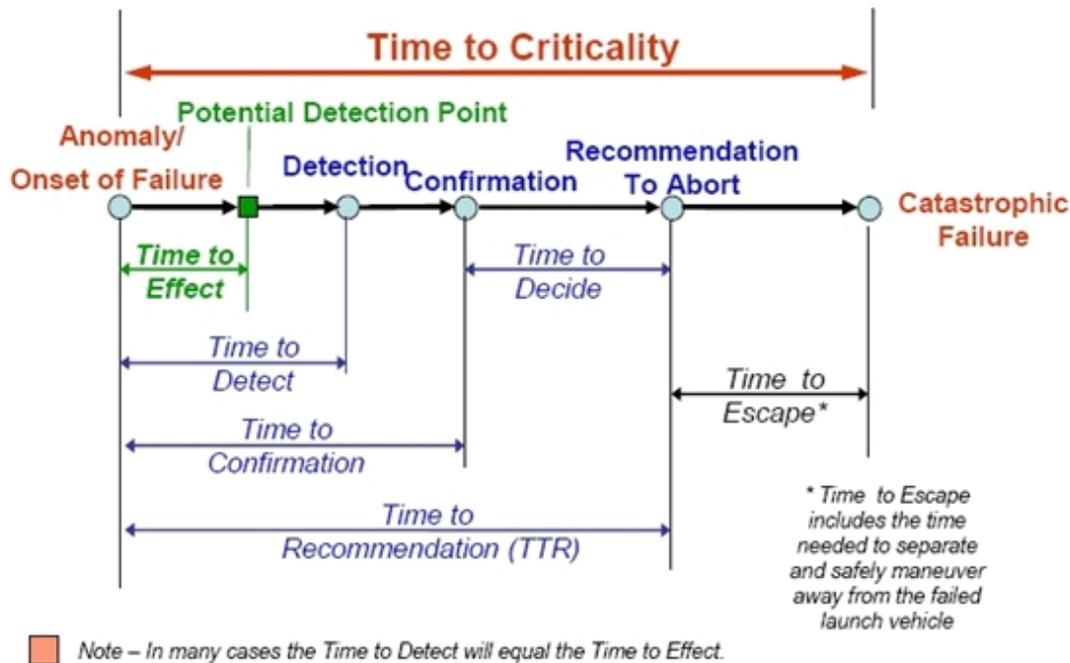


Figure 1. Comparison of Physical vs. Detection and Response Timelines for Onboard Abort

2. UNDERSTANDING THE ABORT DEVELOPMENT

CHALLENGES

Timely fault detection and confirmation of catastrophic events is of paramount importance to ensure crew safety. Launch and ascent is one of the most critical phases of space missions, and system failures at this time can rapidly propagate to Loss of Vehicle (LOV) and Loss of Crew (LOC). The purpose of the abort development process is to analyze methods and technologies for detecting catastrophic failures during the launch phase which will allow sufficient time for the crew to abort safely. The process described in this paper is focused on the Ares I onboard system. A similar system is responsible for detecting abort conditions on the Orion vehicle (for example, cabin pressure problems or life support issues). An interface between the Ares I and Orion abort systems allows the exchange of information related to abort situations. The Orion system maintains an Onboard Abort Executive that processes this information and advises the crew. The Orion Abort Executive and crew are ultimately responsible for abort decisions.

Lessons Learned from Prior Systems

As part of a larger trade study, a review of historical launch vehicle crew abort fault detection systems [2] was undertaken to support the definition of this process. This study looked at prior U.S. launch systems including Mercury, Gemini, Little Joe, Apollo, and Shuttle, along

with the Soviet Soyuz and Chinese Shenzhou. For each launch system, the study looked at the abort system concept, fault detection philosophy, parameters monitored, faults detected, and redundancy design. It also looked at testing that was performed prior to flight and any aborts that were experienced in flight. The results of the study helped to identify three groups of launch vehicle hazards that require crew abort: 1) loss of control, 2) loss of thrust, and 3) explosion. The review also identified the need to understand the physics of failure and how this interacts with the detection, confirmation, and response system.

Onboard Abort Detection Design Challenges

The Ares I in-flight abort system must be able to detect and confirm conditions that may lead to catastrophic failure, notify the crew of the problem, and respond as necessary to save the vehicle. This system must be capable of operating during the entire Ares I flight profile, from the time that the Orion hatch is closed through the separation of Orion from the Ares I vehicle at the end of the boost phase.

There are many challenges in the design of such a system. The first involves detecting a failure that requires an abort in enough time to allow the crew to escape. Figure 1 illustrates the “race” between the physics of a failure and the time required to recognize and act upon it.

The current Ares I Concept of Operations [3] states two available abort options for onboard abort: automated abort and manual abort. How quickly the failure progresses to the point of being catastrophic determines if the abort command

should be automatic or manual. For failure modes with a time to criticality $\leq X$ seconds, automated aborts will be initiated. For time to criticality $> X$ seconds, the crew can initiate a manual abort at a convenient point in the trajectory. Onboard logic will select between automated abort and manual abort, based on a predetermined list of conditions that are derived from a time to criticality analysis.

A systematic process is used to analyze each catastrophic failure mode and determine the simplest and most reliable method to detect and confirm that a crew abort condition exists with sufficient time to safely abort the crew. Figure 1 illustrates the timing associated with fault detection, and shows two timelines that define the crew abort fault detection concept. The top timeline displays the timing associated with the physical attributes of the failure mode. This defines the Time to Criticality (TTC) as the time between the onset of the failure and the time of occurrence of the LOV/LOC event. Each failure mode's TTC, which is a function of flight phase and mission time, defines the allowable time to detect the failure, confirm that it is propagating to a catastrophic event, decide to abort, initiate the abort, separate the crew module from the launch vehicle, and maneuver it sufficiently far away for safety.

The second timeline defines the Time to Recommendation (TTR) to abort as the sum of the *time to detect* the fault or failure, the *time to confirm* or corroborate that the fault or failure exists and is propagating to a catastrophic event, and the *time to decide* to recommend an abort command. In this context *confirmation* is defined as affirmation of the fault or failure condition by measuring similar sensors multiple times, and *corroboration* is defined as affirmation of the fault or failure condition by measuring dissimilar sensors or assessing the health status of other vehicle components or subsystems. The recommended process classifies failure modes as having sufficient time to escape when $TTC - TTR > 2$ (TBR) seconds.

Time to Detect (TTD) is the elapsed time from the onset of the failure to the time at which the fault detection system senses the fault or failure. It is a function of the physical attributes of the failure mode, the type and placement of sensors, and the avionics architecture. TTD includes both the actual sensing of the anomalous measurements and the decision that this anomalous behavior is a real fault and not merely a transient. This typically is longer than *time to effect*, the time from the initiation of a fault until its effects (symptoms) become potentially detectable, as designers may decide that it is better to detect the fault later in the fault propagation path (which generally requires fewer sensors).

Therefore in the development of the abort system, it is important to understand both the time to criticality for each possible fault that could lead to catastrophic failure, and the time required to detect the fault, respond to it, and escape

from the vehicle.

A second challenge is that many faults may lead to similar failures. An analysis must be performed to understand the operation of the vehicle and the catastrophic failures that may occur. A process must be followed to link the faults (which can be detected) to the catastrophic failures (which must be avoided). A fault occurring within a subsystem could lead to a system-wide catastrophic failure in addition to the loss of functionality within that subsystem.

A third challenge is the need to reduce the false positive and false negative indications for abort. A *false negative* describes a situation where an abort is necessary to avoid loss of crew but is not detected by the system. A *false positive* describes an incorrect indication for the need to abort, where none is necessary. During launch and ascent, this could lead to loss of mission and loss of vehicle, and possibly loss of crew since the abort process is not without its own risk. Ideally, both false negatives and false positives need to be as low as possible. In reality, the choice of abort detection thresholds can influence the rates of false negatives and false positives but cannot minimize both at the same time.

Onboard Abort Development Process Overview

Based on the findings of this early work and good engineering practice, the following process and components are being used to develop the Ares I onboard abort system.

1. Identify the relevant Ares I conditions that require an abort (*abort conditions*).
2. Understand the important characteristics of the abort conditions.
3. Develop the algorithms that can detect and properly respond to the abort conditions.
4. Develop and refine the system in an iterative manner, which will continue throughout the design and development of the vehicle.

Probably the most important part of this process is the fourth component. A description of the components and their interrelationships is the focus of this paper.

3. IDENTIFICATION AND CLASSIFICATION OF ABORT CONDITIONS

The first process involves identifying the Ares I abort conditions. The flowchart in Figure 2 illustrates the steps used to develop a comprehensive Abort Conditions List (ACL). The ACL is documented in an Abort Conditions Report (ACR) [4].

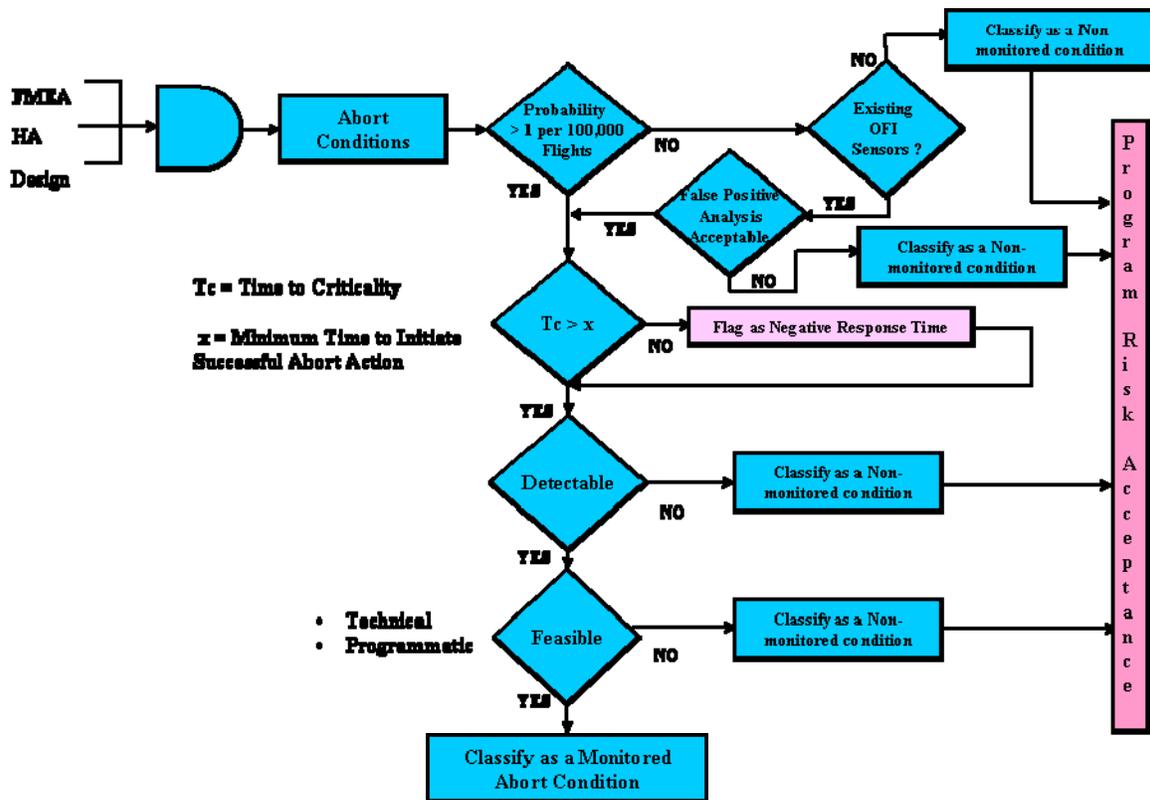


Figure 2. Abort Condition Classification Process

Using engineering data that include a Failure Mode and Effects Analysis (FMEA), Hazard Analysis (HA), and design data, the list of abort conditions is initially populated by the system engineers and reviewed by the engineering team.

Next, the probability of occurrence is developed and evaluated for each abort condition. A credible failure is a failure whose probability of occurrence is greater than some preselected value (the Credibility Limit). Failures whose probability is less than the credibility limit are classified as non-credible conditions. All conditions classified as credible conditions continue to the next step of the abort conditions identification process.

Conditions classified as non-credible are evaluated to determine if the condition is detectable with the current Operational Flight Instrumentation (OFI). A False Detection and Reliability analysis is performed to determine if the detectable non-credible condition should be classified as a not-monitored condition or as a monitored condition. A high probability of a non-credible, but monitored, condition generating a false positive may indicate that the condition should, in fact, not be monitored.

Any non-credible condition classified as non-detectable or not-monitored must be accepted as a risk by the program. These conditions are collected in the Not Monitored list.

The next step in the process is to determine if the propagation of a credible condition is occurring too quickly to allow the detection, confirmation, and abort decision process to be completed with sufficient time for Orion to initiate a safe abort. If the propagation speed is too fast, but existing OFI can detect the condition, then further analysis is performed to assess if the condition should be classified as a monitored condition. If the propagation speed is too fast and the OFI cannot detect the condition, then the condition is classified as a not-monitored condition. Conditions which can be detected but do not provide the minimum time necessary to safely abort may still be monitored in order to provide a chance for Orion to separate and attempt an escape. More detailed analysis on the fault conditions and response timing is conducted to determine if monitoring these conditions would provide an increase in crew survivability.

For credible conditions where the timing analysis indicates that the rate of propagation allows for the detection, confirmation, and abort decision process to be completed with sufficient time for Orion to initiate a safe abort, the next step is to assess the detectability of the condition. Preference is given to detection using existing OFI. However, additional sensors will be recommended for detectable credible conditions which cannot be unambiguously separated from non-abort conditions using OFI sensors.

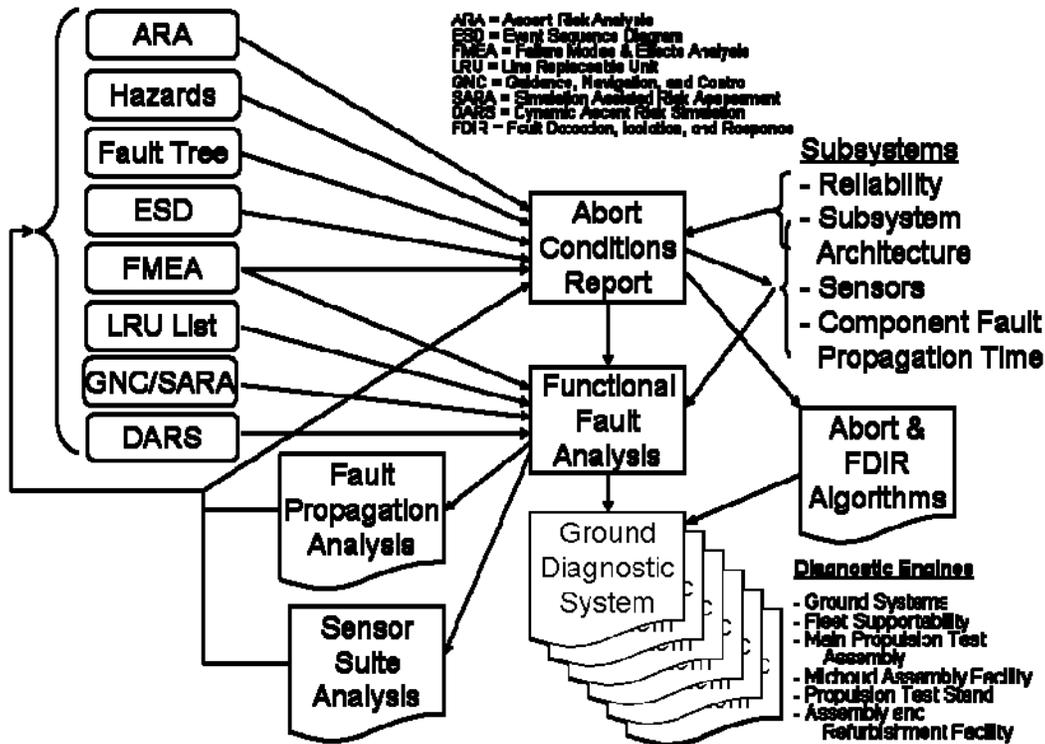


Figure 3. Data processes and interrelationships used in the development of the Ares I abort and diagnostic algorithms

If a credible abort condition is physically non-detectable, the condition is classified as a not-monitored abort condition. If the condition is theoretically detectable, the last step is to assess the feasibility of actually detecting the abort condition. Feasibility of the detection method is defined as either *technical* or *programmatic*. Technical feasibility addresses the ability to place a sensor on the vehicle in the correct location. In some instances a condition is detectable, but the sensor cannot be placed in the proper location for an unambiguous detection. In this case the detection is physically possible but not implementable, or the additional sensor may adversely impact vehicle reliability by a significant amount. Programmatic feasibility addresses both schedule and cost feasibility.

The final products of this process are the Monitored and Not Monitored Abort Conditions lists. All of the data described above is documented in the Abort Conditions Report.

4. CHARACTERIZATION OF ABORT CONDITIONS

The second process in developing the in-flight abort system involves characterizing the abort conditions, primarily through the use of models. The four main models being developed are the Functional Fault Analysis (FFA) model using TEAMS software, a Maveric GN&C simulation in C, an Ares model in Simulink, and a blast analysis model

developed by the Simulated Assisted Risk Assessment (SARA) group. These models play differing, and complementary, roles in simulation, diagnosis, and analysis of the integrated vehicle stack.

Figure 3 shows a schematic of the many analyses and data products used in the development of the Ares I abort detection process. The diagram also shows the downstream processes that use the results of these analyses. Also evident in this diagram is how data from many of the processes flows back to be used in further analysis cycles. The simulation processes are the main consumers of this data.

Functional Fault Analysis

The primary function of the Functional Fault Analysis (FFA) is to collect the timing data of the individual system components in a functional model and use this data to determine the propagation time for the individual faults. The timing data is at a finer level of granularity than that available from FMEAs (fractions of a second, rather than seconds or minutes).

The FFA takes as input system reference information such as architecture diagrams, Concept of Operations Documents, Mission Phases and Operational Modes, and Ares I Line Replaceable Unit (LRU) List. The FFA also takes as input data developed by Subsystem experts,

including subsystem FMEAs and Sensor Lists.

To analyze the propagation of abort conditions, the FFA uses timing data from the Fastest Credible Failure Mode Case (FCFMC), Component Fault Propagation Timing (CFPT), Avionics processing latencies, and Vehicle dynamic timing from the Simulated Ascent Risk Analysis and Ares Guidance, Navigation, and Control (GN&C) groups.

The FFA uses this data to develop a functional model of the Ares I components and systems. This model includes the functions that are performed by the different components, along with the interconnections and flows (power, fuel, information) between them. Loss of a function and its effect on components “down stream” are modeled, showing how a stream of events could lead to a catastrophic result. Since sensors are included in the model, timing estimates for propagation to various sensors is available, which allows several important analyses to be performed. FFA models are developed using a commercial tool called TEAMS Designer from Qualtech Systems Inc. A few custom features have been added to TEAMS to support the FFA objectives.

The primary output of the FFA is the timing analysis. The FFA models at least one, fastest credible failure mode for every Abort Condition in the Abort Conditions List. For each failure mode modeled, the FFA produces much of the timing data described in Figure 1. This includes Time to Effect, Time to Detect., Time to Confirm, a Time to Abort recommendation, and portions of Time to Criticality.

Another important analysis that FFA performs is the development of a list of sensors for each abort condition that can detect that condition. From that list, primary sensors and confirmation sensors are selected for each abort condition.

A third analysis that the FFA supports concerns sensor placement. Sensors can be moved, or added, in the model to reduce ambiguity or reduce the time to detect and confirm faults.

Each of these outputs is used in the Ares ACR, and fed back to the Ares I System Elements, the Subsystem Experts, the Event Sequence Diagram (ESD) task, the Ares Integrated Aborts Team, the Constellation Integrated Aborts Team, Safety and Mission Assurance, and the Ascent Reliability Analysis Team.

Equation-Based Simulations

The Maveric (Marshall Aerospace Vehicle Representation in C) software simulates the flight dynamics of the combined Ares I and Orion Vehicle, in various configurations, throughout the flight envelope. It provides vehicle performance data and trajectory information. It can also simulate certain failures such as flight-related effector

failures.

Although Maveric can show the resulting effect of a failure (for example, a vehicle tumbling, or veering off a trajectory), details of the underlying component failure that caused that effect on the vehicle are not modeled.

The Maveric models will be augmented by a complete Ares I model in Matlab/Simulink that is being developed at NASA Ames. This model simulates the operation of major subsystems in the Upper Stage, as well as the Upper Stage Engine and the First Stage. It accounts for varying atmospheric properties through a flight profile and also accounts for a changing center of gravity of the vehicle stack as the propellants deplete. It incorporates guidance and control of the vehicle through thrust vectoring and a reaction control system. It also includes structural response effects.

There is sufficient detail in the Simulink model to allow it to include real-world effects such as system noise and sensor noise. The simulation outputs will be datasets that mimic sensor data on a launch vehicle for various nominal and failure scenarios. These scenario datasets will be inputs to the detection and confirmation logic in the abort algorithms.

The abort algorithms will be assessed for false positive and false negative indications for a given set of failure scenarios. The Simulink model can then be used to generate scenarios that exercise these algorithms in critical scenarios such as subsystems approaching the failure limits, jittery sensor readings near the threshold value, and other such operational scenarios.

Simulation Assisted Risk Assessment

The Simulation Assisted Risk Assessment (SARA) is another simulation that provides data to the abort decision and detection processes. The SARA group provides data on the results and timing of the results of catastrophic failures. SARA uses a blast analysis approach to simulate the destruction of the Ares I vehicle due to many failures at different points in the flight. The SARA simulations provide support for timing estimates of the time required for the crew to safely escape the situation using the Launch Abort System (LAS)

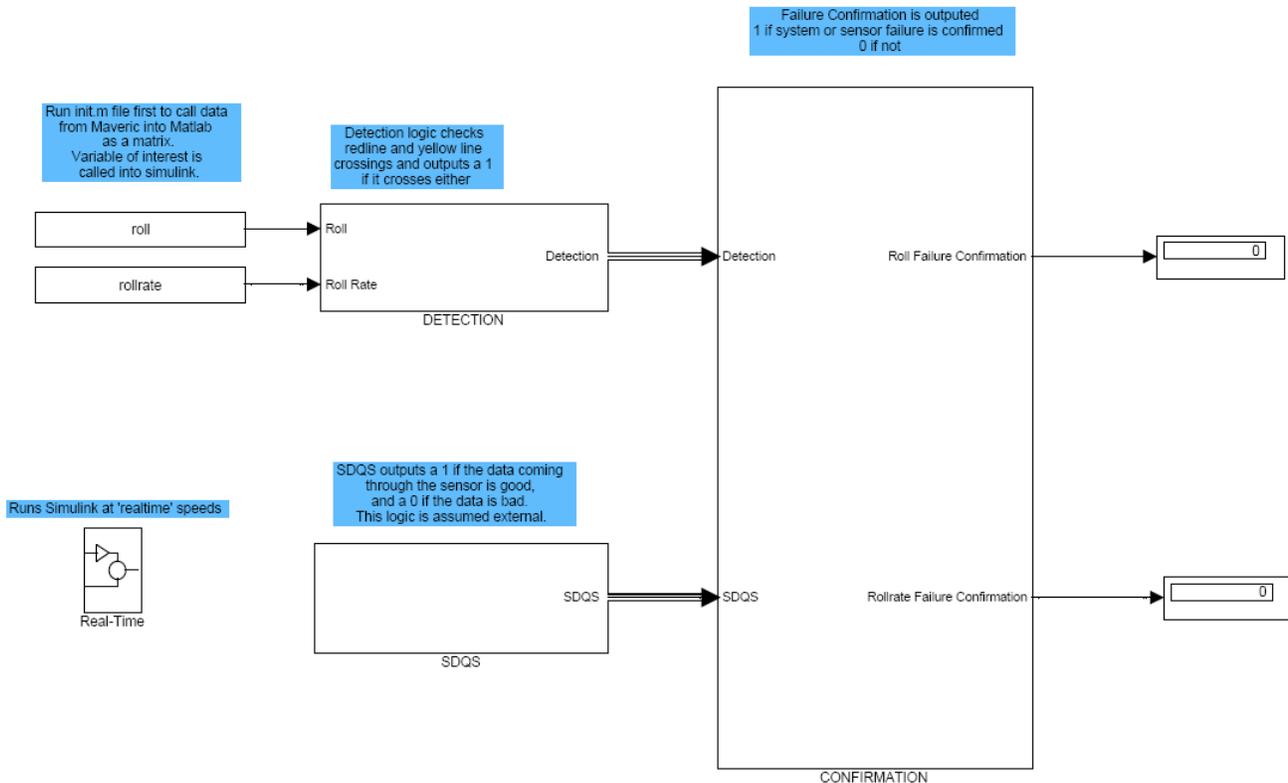


Figure 4. Example high-level model and representation of detection and confirmation logic using Simulink

5. ABORT ALGORITHM DEVELOPMENT

The objective of the abort algorithm development process is to define the detection, confirmation, and response processes needed to develop this system. Algorithm development takes input data, as a telemetry stream, from the vehicle simulation and uses it to test the algorithms, sensors, and timing requirements for detecting in-flight failures that could lead to abort.

The abort algorithm development process is documented within a System Definition Document (SDD) [5]. The SDD provides requirements and guidance to the Avionics development team in implementing the abort system and algorithms. The SDD describes the abort algorithm requirements at a system and implementation level.

At the algorithm level, the SDD describes the algorithms, sensors, phase of flight and other data required to detect and confirm each abort condition. It takes as input the data provided by the analysis and simulation processes previously performed. The development of these algorithms is currently being performed using the commercial Matlab/Simulink tool, which allows the algorithms to be tested individually using simulation and modeling data also developed in the previous tasks. Figure 4 shows a high-level block diagram as generated by Simulink. The algorithms will also be tested together to determine and resolve any

interactions between them before they are delivered.

At the architecture level, the logic flow is described for the high-level abort functions. This describes how the abort system operates and interacts with the flight software and the other system components (for example, the engine or the Orion vehicle). Functions described at this level involve tasks such as monitoring for abort conditions or abort commands from the Orion, notifying Orion and Ground of the abort status, and performing automatic or commanded tasks related to abort. The high-level abort functions are described in UML using a commercial tool (Enterprise Architect). Documenting this work in UML provides a well-understood representation as a deliverable to the flight software group. Figure 5 shows an example of a high-level UML use case. In this diagram, CLV refers to the Ares I, CEV refers to the Orion capsule, and AFDNR refers to the Abort system.

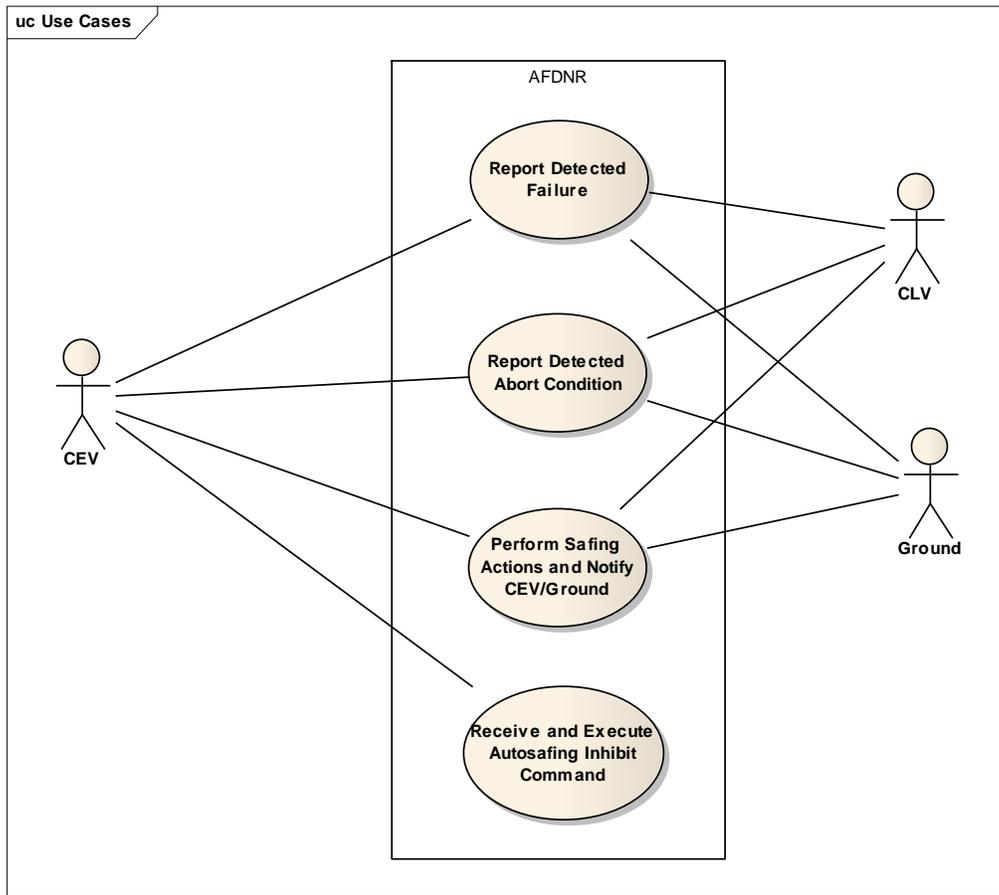


Figure 5. Example of a UML Use Case representation used to model the high-level abort system architecture

Verification and Validation (V&V)

Verification and Validation (V&V) of the Abort algorithms and system design are an integral part of the process. A V&V plan has been developed to support three separate areas: Process Verification, Artifact/Product Verification, and Validation.

The first part of Process Verification involves defining a software V&V process by creating a plan that follows accepted industry standards (for example IEEE standards, and NASA Procedure Requirements). The second part is verifying that the defined process is actually being followed.

Artifact & Product Verification involves verifying that models and other software artifacts match what is required. Performing this process during development simplifies the V&V required of the deliverables. This is being accomplished by developing the algorithms using UML, which provides several advantages during development: UML diagrams can be verified through reviews and by using NASA-developed and other UML verification tools.

Supporting the Validation of the resulting system is also accomplished during the development by generating and documenting test cases and using these to test the abort

failure functions.

Algorithm Testing through Higher Level Simulation

Several levels of testing will be performed on the software algorithms and abort architecture. The algorithms will be tested in the Fault Detection, Diagnosis, and Response (FDDR) integration laboratory at NASA Marshall, where the individual algorithms will be integrated and tested in a larger context. This laboratory will simulate data over areas critical to the flight. Changes found necessary at this point will flow back to improve the design of the algorithms prior to their delivery to the software development team.

Additional validation of the abort detection algorithms and architecture will be performed within a System Integration Laboratory (SIL) at NASA Marshall. The SIL will provide an “iron bird” flight hardware simulation of the Ares I vehicle that will support software and hardware component testing. In the case of abort, the laboratory will test the implementation of the abort system as developed within the full avionics and flight software architecture under multiple abort scenarios.

Probably the highest level of simulation will involve testing the algorithms on a flight vehicle. Plans are in place to operate the abort algorithms on Ares I developmental flights, which would provide real data and results.

6. ITERATIVE REVIEW DURING DEVELOPMENT

It is important to make the point that the processes above are performed almost in parallel with increasing definition at each cycle. This process is expected to continue through Preliminary (PDR) and Critical (CDR) Design Review with review and examination of the requirements and design.

We also expect the review and reexamination process to continue through avionics implementation, ground and flight testing, and material changes required by fleet supportability analyses. This need for continual review of modifications has been brought to the forefront by several major launch system failures. The Ariane V [6] and Atlas IV [7] failures might have been avoided if the changes that had been made to their systems had been brought back through the formal design process to be reevaluated, simulated, and tested before these systems were flown on the vehicle.

7. SUMMARY

This paper has described the need for the onboard abort system, and has described the abort condition classification, characterization, and algorithm development processes. It also described the need for an iterative process that continues throughout the development of the vehicle.

The Ares I Onboard Abort Detection, Confirmation, and Response System is arguably the most important system on the vehicle system. It is important that this system operates properly when it is called upon. We feel that the process being followed in its development will ensure that this occurs.

REFERENCES

- [1] Constellation Architecture Requirements Document, CxP-70000 CARD Pre ICPR Release, Marshall Space Flight Center, Huntsville, AL, March 31, 2007.
- [2] Trade Study Report, AT-0004 Crew Abort Failure Detection and Decision Making, Marshall Space Flight Center, Huntsville, AL, July 2006.
- [3] Constellation Program, Ares I Concept of Operations (CONOPS) Overview, Marshall Space Flight Center, Huntsville, AL, November, 2005.
- [4] Constellation Program, Ares I Abort Conditions Report (ACR), Baseline Draft, CxP-72245, Marshall Space Flight Center, Huntsville, AL March, 2007.
- [5] Constellation Program, Ares I Abort Failure Detection, Notification, and Response, System Definition Document, (SDD), Baseline Draft, CxP-72244, March, 2007.
- [6] Nuseibeh, Bashar. Ariane 5: Who Dunit?, *IEEE Software* **14** (3): 15-16., May 1997.
- [7] Root Cause Declared for Delta IV Heavy Demo Mission, Spaceref.com, March 15, 2005.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the contributions of Mike Watson, Stephen Johnson, and Jon Patterson of the NASA Marshall Space Flight Center and Serdar Uckun and Dougal Maclise of the NASA Ames Research Center for enabling and managing this work.

It's also important to site and thank the many engineers that have contributed to the development of this work, including, but not limited to: Peter Berg, Jon Breckenridge, Eric Barszcz, Anita Cooper, Martin Feather, Ben Hayashida, Jeremy Johnson, Larry Markosian, and Chris Neukom.

AUTHORS



Greg Pisanich is a Technical Area Liaison for the QSS Group Inc./ Perot Systems Government Services within the Computational Sciences Division at the NASA Ames Research Center. He holds Master's degrees in Aeronautical Science from Embry Riddle Aeronautical University and Computer Engineering from Santa Clara University. His background and technical interests include aviation, unmanned aerial vehicles (UAVs), simulation, robotics, autonomy, cognitive modeling, and human factors.



Dr. Anupa Bajwa is a Project Scientist with the University of California. She has been working at Ames Research Center for over ten years. She is a specialist in Integrated Systems Health Management (ISHM) for spacecraft. She is working on methods for failure detection and diagnosis for NASA's Ares I. This involves modeling and simulating spacecraft subsystems. Her earlier projects in ISHM include Propulsion IVHM Technology Experiment, model-based health monitoring on Earth Observer-1, Space Launch Initiative's Orbital Space Plane, and diagnostic models of Jupiter Icy Moons Orbiter's propulsion system. Dr. Bajwa has a Ph.D. in Aerospace Engineering from the Pennsylvania State University.



Dwight Sanderfer is a Senior Systems Engineer with the NASA Ames Research Center. He has worked at NASA Ames for over 28 years. The last nine years he has worked in the in the Integrated Systems Health Management (ISHM) group in the NASA Ames Intelligent Systems Division. He is currently leading the Ames team developing the crew abort condition detection algorithms for the Ares I launch vehicle. He has a Masters in Mechanical Engineering from the University of Wisconsin, Madison.

..

