Proceedings of IDETC/CIE 2006
ASME 2006 International Design Engineering Technical Conferences &
Computers and Information in Engineering Conference
September 10-13, 2006, Philadelphia, Pennsylvania, USA

IDETC2006-99535

A FUNCTION-BASED METHODOLOGY FOR ANALYZING CRITICAL EVENTS

Ryan S. Hutcheson, Daniel A. McAdams, Robert B. Stone
University of Missouri – Rolla
rhutch@umr.edu, dmcadams@umr.edu, rstone@umr.edu

Irem Y. Tumer
NASA Ames Research Center
itumer@email.arc.nasa.gov
(Corresponding Author)

Abstract

The objective of this research was to develop a function-based method for analyzing the critical sequences of events that must occur for complex space missions to be successful. The resulting methodology, the Function-based Analysis of Critical Events, or FACE, uses functional and event models to identify the changes in functionality of a system as it transitions between critical mission events. Two examples are presented that detail the application of FACE to prior mission failures including the loss of the Columbia orbiter and the failure of the Mars Polar Lander probe. The result of the research is a methodology that allows designers to not only reduce the occurrence of such failures but also analyze the specific functional causes of the failures when they do occur.

1) Introduction

Modern aerospace systems are extraordinarily complex and are often tasked to complete equally complex mission profiles. During these missions, critical sequences of events must be followed to result in successful missions. When designing such systems, knowledge of the functional interactions between sub-systems across multiple mission events is essential. This information is required not only to design systems capable of completing desired mission profiles but also to identify the causes of undesirable mission outcomes. The research presented in this paper outlines a structured method for identifying and modeling the functional interactions of complex systems across critical mission events. The first section provides motivating examples and further outlines the specific problem we attempt to solve. Section 2 introduces our proposed solution to this problem, the Functional Analysis of Critical Events (FACE) methodology. This methodology is to be used retroactively to explore the functional propagation of failures after they occur and proactively to prevent future failures. Case studies of the application of FACE to two mission failures are presented in Section 3. Finally, conclusions from the research and future work appear in Section 4.

1.1) Motivation for Research

On December 3, 1999 contact was lost with the Jet Propulsion Lab's Mars Polar Lander (MPL) after its planned touchdown on the surface of Mars. After several weeks without communication with the probe, it was determined that the probe had been lost. A review board was assembled to investigate the MPL's failure. The MPL review board concluded that a spurious touchdown signal generated during the deployment of the landing legs caused a premature shutdown of the probe's decent thrusters. The failure was replicated in a laboratory and corrective software updates were sent to other probes using similar software. It was assessed by the review board that a breakdown in systems engineering practices led to the software condition that likely resulted in the loss of MPL. Specifically, requirements for the touchdown monitoring (TDM) software were not properly communicated between the hardware designers of the landing systems and the software designers of the TDM system [1].

On February 1, 2003 Space Shuttle Columbia broke up upon reentry into Earth's atmosphere. As with the MPL failure, a review board was assembled to investigate the loss of Columbia and her crew of seven astronauts. The conclusion of the review board was that a foam strike during launch caused damage to one or more of the left wing leading edge reinforced carbon-carbon (RCC) panels. During reentry, the foam strike damage to the RCC panel allowed superheated gases to penetrate the shuttle's thermal protection system, which ultimately resulted in a failure of the left wing structure and the loss of Columbia [2].

Both accidents resulted from a failure during a mission event affecting the functionality of the system during a later event. In the case of the MPL, a software failure allowed an undesired system state to propagate from parachute descent to landing. In the Columbia accident, a failure during launch resulted in a catastrophic failure during reentry.

1.2) Critical Events

In order to prevent such accidents from occurring in future systems, it is necessary to both identify and model the changes in functionality that occur between critical mission events. Critical events are the events in a mission that must occur in sequence in order to achieve overall mission success. The following section further defines critical events and introduces how these events are currently being handled during mission design at the Jet Propulsion Laboratory and NASA's Goddard Space Flight Center and during operations at NASA's Johnson Space Center. Opportunities to improve critical event handling practices are also identified. Finally, FACE, a methodology that formalizes and structures the handling of critical events is proposed.

1.3) Critical Event Handling Practices

There are currently several methods available for modeling sequences of discrete events. Most of the work in this area has been focused on creating detailed dynamic models of systems that include discrete events [3]. The objective of this research is to use simple event models to represent the critical sequences of events a system must complete in order to result in a successful mission. These event models should be created during the early stages of the design of a system in conjunction with the functional modeling of the system. Functional modeling techniques will be used as the basis for creating these models. It is the objective of this research to use these event and functional models as a means for assisting the identification of requirements very early in the design of a system. An explanation of current NASA and JPL practices for defining such requirements during sequences of critical events follows.

1.3.1)Jet Propulsion Laboratory– Analysis of Critical Events During Early Design

JPL defines mission-critical events as those that, "if not executed properly, could result in failure to accomplish the mission" [4]. To handle these events, JPL uses fault tolerance design principles including event continuation in response to non-mission-threatening anomalies; safing in response to mission-threatening anomalies (this safing requires ground intervention for recovery); on-board backup of critical event telemetry in the event that upload capability is lost; as well as diagnostic engineering telemetry, including telemetry data downlinked for critical events [5-8].

During mission design, flight sequences for launch and early flight operations are developed along with a baseline version of mission-critical sequences. Identification of critical event sequences during flight sequence design ensures that telemetry is available for real-time monitoring as well as verification of successful competition of events [4-8].

1.3.2)Goddard Space Flight Center–Critical Events During Early Design

Goddard Space Flight Center designs and operates unmanned earth science missions. During the design of systems, GSFC uses the "Gold Book", a set of technical design standards for designing space flight systems. Standards are included in the Gold Book for establishing the telemetry required to ensure the completion of mission-critical events. The GSFC process for handling critical events follows.

Mission-critical events are first identified during a Mission Concept Review. During a Mission Definition Review, telemetry coverage for critical events is established. The telemetry information is then flowed to the Operational Procedures during an Operational Readiness Review. Finally, the capability to sufficiently monitor critical events is verified during mission operations checkout [4,9].

1.3.3)Johnson Space Center – Analysis of Critical Events During Operations

JSC's focus is on human spaceflight operations, specifically for the Space Shuttle and International Space Station (ISS). As a result, JSC does not explicitly define practices and procedures for the design, development and verification of critical events for flight systems. However, JSC does evaluate critical events during the operation of shuttle and ISS missions.

JSC's process for analyzing critical events during operations begins with a Hazard Analysis process to identify and analyze critical events. In this step, potential hazards are identified and documented in Hazard Reports. For example, "a hazard report may specify the risks of a space shuttle Reaction Control System thruster plume impacting an ISS solar array during docking, thereby damaging or breaking the solar array" [4,10]. These Hazard Reports are flowed to mission operations where the critical events in which the potential hazards can occur are identified. Additionally, controls and procedures are established to mitigate the potential of the hazard occurring. An example of such a control for the thruster plume example could be a "reconfiguration of systems to withstand shuttle thruster plumes" [4,10]. For a complete review of how mission-critical events are handled at NASA see [4].

1.4) Improving Current Practices

As shown in the previous section, each mission center uses its own method for identifying and handling critical events. In each case, procedures for *when* critical events should be identified and handled are well established. Unfortunately, specific procedures for *how* the events should be identified and handled are not as clear.

Three opportunities have been identified to clarify how critical events are identified and handled.

1) The first opportunity stems from a lack of a standard framework for handling critical events. Between three different mission centers, three different critical event handling procedures are used. Each center uses an in-house developed method to handle critical events that is tailored to the needs of the center. To promote commonality in the way critical events are handled, as well as to improve communications of lessons learned between centers, it is proposed that a standard methodology be used when analyzing critical events.

2) The second opportunity results from a lack of a standard method for determining and representing the information that must be monitored to ensure the successful completion of critical events. Functional modeling techniques are proposed as a solution for identifying and representing this information. Formal functional modeling practices provide a standard means for discovering the important functionality and energy, material and signal flows of a system during the transitions between critical mission events [12].

3) The third opportunity is the ability to assist the identification of requirements during mission design through the application of formal functional and event modeling practices.

1.4.1)Functional Analysis of Critical Events

To improve upon current critical event handling practices, FACE, a function-based method for analyzing critical events is proposed. FACE involves the application of formal functional modeling techniques to the development of event models, identification of critical functionality and analysis of boundary flows between events. Formal functional modeling practices form the framework of the FACE methodology. In the following sections, the FACE methodology is presented in detail along with clarifying examples.

1.4.2)Formal Functional Modeling

Functional modeling is used in the FACE method to capture the desired functionality of a system during critical events. Functional modeling is a technique used to represent the functionality of a system independent of its form [11]. Typically, functional models consist of functions and flows represented as verbs and nouns respectively. The flows are generally broken down into three categories: energy, material and signal. In the FACE methodology, functional models are made for each sub-system during each event and are used to analyze transitions in functionality and boundary flows between events.

Formal functional modeling techniques are suggested as a means to improve the quality of the models and to assist storage and reuse of the models. Formal functional modeling involves the use of a standard modeling procedure as well as a standard taxonomy. The use of a standard taxonomy and procedure enables the creation of "user-independent" [11] functional models. A user-independent functional model can be understood by any engineer with a copy of the procedures and taxonomy. The FACE method utilizes the Functional Basis (FB) taxonomy [12] and functional modeling process proposed in [11]. The Functional Basis is a hierarchical set of flows and functions that has been successfully applied to the functional modeling of numerous electro-mechanical systems [12]. The use of a standard taxonomy also enables the use of function-based failure mode identification methods and function-based engineering design repositories [13,14]. Additionally, the authors have been working to apply formal functional modeling techniques to the design of aerospace system, see [15] and [16] for more information.

The recommended functional modeling procedure consists of five steps: The first step (1) is to identify flows that are important to the overall functionality of the system. These input and output flows are expressed using Functional Basis terms. The next step (2) is to create a black-box model of the system. This model contains all of the inputs and outputs of the system along with an overall function that describes the system. Function chains are then created to represent the operations performed on a flow to transform it from an input to an output (step 3). These chains are then aggregated to produce an overall functional model (step 4). The next step (5) is to verify that all of the desired functionality of the system has been represented in the model. The result is a model that describes the function of a system separate from its form in a standardized language [12].

1.5) Comparison to Current Methods

Table 1 presents a comparison of the FACE methodology to existing methods including hazard analysis, event tree analysis and fault tree analysis. A hazard analysis begins with identifying a list of the hazardous elements of a system. Triggering elements, hazardous conditions and potential accidents are then identified (forward logic). Hazard analyses are failure-based; they are created by identifying potential failures and investigating their causes and effects [17]. An event tree analysis begins with an initiating event and includes the sequence of possible events that can occur thereafter (forward logic). Boolean logic is used to determine whether the event ultimately results in an accident or safe operation. Event trees are both event-based and failure-based (a failure initiates the tree and a sequence of events follows). Event trees are usually applied during the design of a system to investigate potential failure scenarios. Fault trees start with a top-level failure and use Boolean logic gates to work down to the specific component-level cause of the failure (backward logic). Fault trees are failure-based and generally used during the design of a system to investigate how a top-level failure could occur as a result of lower-level failures [18].

FACE uses functional models and event models to represent the changes in functionality a system undergoes as it performs a mission. As a result, it is both function-based and event-based. FACE can also be applied using forward or backward logic. An analysis can begin with an overall failure of a system and trace back to the specific functions that caused the overall failure (backward logic) or an analysis can start with a failure in a specific function and follow the subsequent failures that result (forward logic). FACE was originally developed as a tool to analyze the specific functional causes of a failure after it occurred (retroactive application) . We argue in this paper that FACE can also be applied during design to identify requirements and functionality necessary to mitigate future failures.

	Forward Logic	Backward Logic	Model-based	Event-based	Function-based	Failure-based	Proactive Application	Retroactive Application
Hazard Analysis	X					X	X	
Event Tree	X		X	X		X	X	
Fault Tree		X	X			X	X	
FACE	X	X	X	X	X		X	X

Table 1 – Method Comparison

2) Functional Analysis of Critical Events

The application of Functional Analysis of Critical Events to the design of a system involves the following three steps:

1. Event Modeling
2. Functional Modeling
3. Requirements Identification

The first step, event modeling, involves the creation of an event model for the mission. This model includes the sub-events that must successfully occur in sequence to result in a successful critical event as well as the energy, material and signal flows through each sub-event. The next step is to create functional models for the sub-systems of interest during each sub-event using the boundary flows into the event as a starting point. The final step is to identify the requirements for each boundary flow into and out from each functional model during each sub-event and to identify any changes in the functionality of sub-systems between sub-events. The following section presents each step in detail with a clarifying example based on the Mars Polar Lander landing leg deployment.

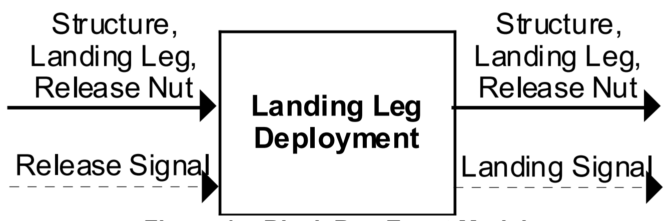
2.1) Event Modeling

2.1.1) Black Box Event Modeling - Methodology

The first step in the FACE process is to create a black box event model for the critical event being examined. This black box model represents the starting point for creating a complete event model and defines the overall event along with the energy, material and signal inputs and outputs of the event. The event name used in the model can be free language. However, the flows should be taken from the Functional Basis.

2.1.2) Black Box Event Modeling - Example

The critical event being analyzed for the MPL is the deployment of a landing leg. To create the event black box model, an event name and boundary flows must be identified. In this example, the event name was chosen as *Landing Leg Deployment*. Next, a boundary was established for the event and all energy, material and signal flows into and out from this boundary were identified. The material flows include the structure of the spacecraft (*solid*), the landing leg (*solid*) and the release nut used to release the leg (*solid*). During the deployment of the leg, no external energy is required other than an electrical input to trigger the release nut. Since this energy is used as a signal (its magnitude is not important) it will be represented as a signal (Release, *control signal*). Leaving the event is a Landing Signal (*control signal*). The selected event name and identified flows are represented in the black box event model that appears in Figure 1.


Structure, Landing Leg, Release Nut
Release Signal
Landing Leg Deployment
Structure, Landing Leg, Release Nut
Landing Signal
Figure 1 – Black Box Event Model

2.1.3) Detailed Event Modeling - Methodology

After the black box event model has been constructed, a complete event model is developed. This model consists of the chains of sub-events that must be completed in sequence in order for the critical event to be successful. These sub-events will be used to define changes in functionality for the sub-systems. To create the event model, an event sequence must first be generated. If available, an event tree can be used to define the event sequence. If not, a sequence can be generated by starting with the initial state of the system and determining the discrete changes in the system's state that occur as time progresses. Once sub-events have been identified, the input and output flows to the sub-event must be determined. As in the black box event model, these flows represent the energies, materials and signals required to complete the event and should be represented using FB terms.

2.1.4) Detailed Event Modeling – Example

For the MPL landing leg example, the event sequence was determined from the MPL mishap report [1]. The first sub-event represents the beginning of the event and is labeled *Begin Deployment*. This sub-event requires the Structure of the spacecraft, the Landing Leg and a Release Nut to be input and output (all sub-events in this model include these three material flows). The next step is the releasing of the nut that retains the landing leg. This sub-event includes an additional input flow of a Release Signal. After the release nut has been triggered, the landing leg is deployed, this is represented as the *Deploy Leg* sub-event and includes a Landing Signal output.

Next, the landing leg is latched in the landing position in the *Latch Leg* sub-event. Finally, the end of the event is represented using the *End Deployment* sub-event. This sequence of sub-events is represented in Figure 2.

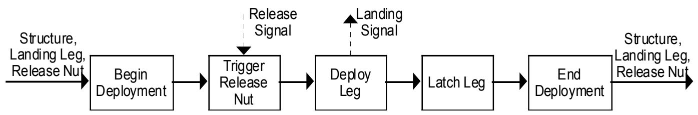


Figure 2 – Event Sequence for the Landing Leg Deployment Critical Event

2.2) *Functional Modeling*

Once the event model has been created, functional models need to be created for the sub-systems of interest during each sub-event. These functional models will be used to analyze changes in functionality and boundary flows between sub-events. Like the event modeling process, functional modeling includes a black box modeling step and a step for generating a complete functional model. The following section explains the functional modeling process and continues the MPL landing leg example.

2.2.1) Black Box Functional Modeling-Methodology

The first step in the FACE functional modeling process is to create a black box functional model for the sub-system(s) of interest. This model represents the overall functionality of the sub-system as well as the input and output flows to the system. The flows should be represented using the FB. If the sub-system is electromechanical, the FB should also be used to represent functions. If not, free language verb-noun pairs should be used. In the following example, FB terms are noted in underlined italics.

2.2.2) Black Box Functional Modeling- Example

For the MPL example, the sub-system being analyzed is the landing leg. In this example, two consecutive events are modeled. The first event is *Trigger Release Nut*. During this event, the overall functionality of the landing leg sub-system is to separate from the release nut that restrains its motion. The FB function-flow pair chosen to represent this functionality was *Separate Solid*. The inputs to this black box function are the Release Nut (*solid*) and a Release Signal (*control signal*). The outputs are the Release Nut (*solid*) and a signal indicating the separation of the Release Nut (Separation*, status signal*). The next event considered was *Deploy Leg*. During this event, the overall functionality of the landing leg sub-system is to deploy the landing leg using energy stored in a spring. The FB function and flow pair for this action was *Convert Mechanical Energy to Rotational Energy.* The inputs to this function are the Release Nut and a separation signal from the previous event. The outputs are the Release Nut and the *Rotational Energy* of the landing leg. The black box models of the functionality of the landing leg sub-system during these two events appear in Figures 3 and 4.

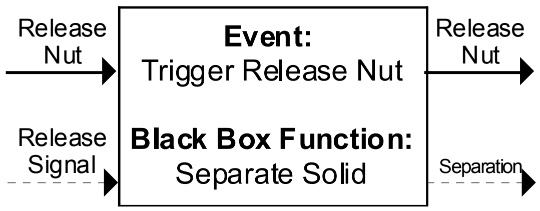


Figure 3 – Black Box Model During Trigger Release Nut Event

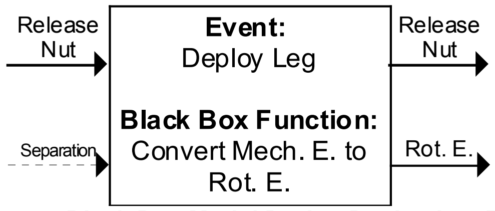


Figure 4 – Black Box Model During Deploy Leg Event

2.2.3) Functional Modeling-Methodology

Once the black box functional models have been created, complete functional models should be made. These models include all of the sub-functions that must be completed to transform the black box inputs into the desired outputs. Once again, FB function and flow pairs should be used if the system is electro-mechanical. To create the detailed functional models, a function chain should be constructed for each black box input. These chains include all of the individual functions that must by completed in order to arrive at the desired output(s). Once chains have been produced, they can be aggregated to create a functional model.

2.2.4) Functional Modeling-Example

To create the functional models for the two selected MPL landing leg events, a complete functional model was first made for the landing leg sub-system as a whole. This model was then partitioned to create functional models for each of the two sub-events. To create the complete functional model, function chains were created for each flow into the black box models for the two sub-events being analyzed starting with the flow of the Release Nut. The first function that is performed on the nut is *import* (bringing the flow from outside of the system). Next, the nut is *position*ed before being *secure*d to a bolt on the landing leg. While secured, the function of the nut is to *stop mechanical energy* from being transmitted through the system. When triggered by a release signal, the function of the landing leg system is to *separate* itself from the release nut. Finally, the release nut is *export*ed from the system. The next flow considered was the flow of *rotational energy* through the landing leg. This energy is first *import*ed into the system (this is the initial energy added to the system to store the landing leg). This rotational energy is converted to *mechanical energy* that is *store*d in a spring. The flow of energy out of the spring is restrained by the release nut until the *separate solid* function has been accomplished. At this point, the energy stored in the spring is *supplied* and *converted* into *rotational energy*. Finally, this energy is *export*ed from the system. The last flow chain analyzed was the release signal. This signal flow is first *import*ed into the system and then used to trigger the *separate solid* function. These three flows were aggregated to produce a functional model for the landing leg system. During the two critical events of interest (*Trigger Release Nut* and *Deploy Arm*), only a subset of the functions are actually used. The complete functional model with the functions active during each of these events are shown in Figures 5 and 6. A schematic of the landing leg appears in Figure 7.

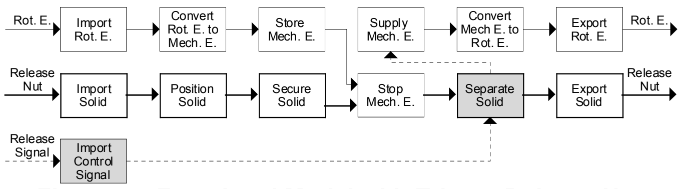


Figure 5 – Functional Model with Trigger Release Nut Functions Highlighted

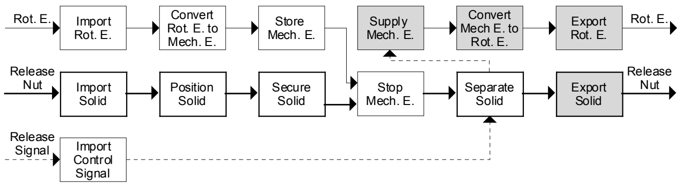


Figure 6 – Functional Model with Deploy Leg Functions Highlighted

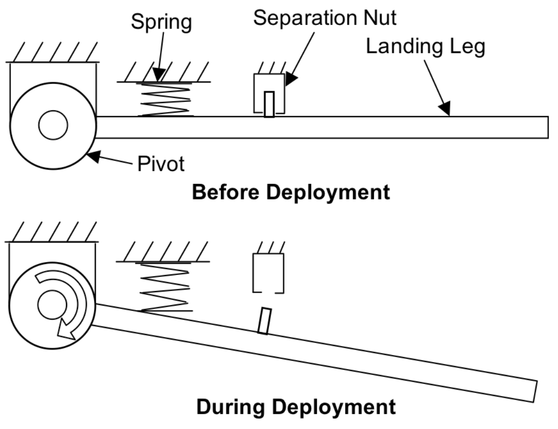


Figure 7 – Landing Leg Schematic

2.3)Requirements Identification

2.3.1) Requirements Identification– Methodology

The requirements identification step of the FACE methodology consists of the following four tasks:

1) Identifying changes in black box functionality between events
2) Identifying changes in functionality within the detailed functional models between events
3) Listing the critical functions that undergo changes between events as well as their input and output flows
4) Establishing requirements for the system based on the identified flows
    a. Defining nominal values for the flows during the event transitions
    b. Adding measurement functions to ensure that critical flows are monitored during the transitions
    c. Adding sensing functions to ensure that the states of critical functions are monitored

The first step (1) in identifying requirements is to identify changes in black box functionality and flows between events. This information represents the changes in overall functionality of the system and identifies boundary flows that must be measured to ensure that the system is ready to transition between states. The next step (2) is to identify changes in functionality within the detailed functional models of the sub-system being analyzed. This information is then used (3) to locate important functions and flows than must be monitored during the transition between events. Finally (4), requirements should be established for the functions and flows that undergo changes between critical events. Functions should then be added to the functional model to measure the identified transition functions and flows. *Measure* functions should be added to monitor flows and *sense* functions should be added to the model to determine the state of functions.

2.3.2) Requirements Identification – Example

Between the Trigger Release Nut and Deploy Leg events, the overall functionality of the landing leg system transitions between *separate solid* and *convert mechanical energy to rotational energy*. To ensure that the overall functionary transitions as desired, requirements should be placed on the input and output flows to each black box model. An example of requirements generated from the input and output flows of the Trigger Release Nut event appears in Table 2.

Table 2 – Trigger Release Nut Sub-Event Flow Requirements

Flow Type	Flow	Requirement
Solid Input	Release Nut	The release nut must be properly positioned and secured before the release event can occur
Control Signal Input	Release Signal	The Release Signal will initiate the Trigger release Nut event
Solid Output	Release Nut	At the completion of the event, the Release Nut will be separated from the landing leg
Signal Output	Separation	After completion of the event, the subsequent event will be initiated without a formal signal

After transitions in overall functionality and flows have been identified, transitions in sub-functions should be found. During the *Trigger Release Nut* event the sub-functions used in the landing leg sub-system are *import control signal* and *separate solid*. During the *Deploy Leg* sub-event, the active sub-functions are *supply mech. e.*, *convert mech. e. to rot. e.*, *export rot. e.* and *export solid*. To ensure that the system transitions as planned, additional functionality can be added. To determine the state of a function, a *detect* function can be added. The state of a flow is determined by a *measure*ment function. For example, to ensure that the *Trigger Release Nut* sub-event occurs as desired, a *detect* function should be placed on the *separate solid* function in the functional model. This detect function indicates that some hardware should be added to the design to positively confirm that the *separate* function has properly occurred. The output of the *detect* function will be a status signal.

2.4) FACE Methodology Discussion

The FACE method uses three general steps including event modeling, functional modeling and requirements identification. These steps are used to establish the sequence of events that must occur for a critical event to occur as planned, determine the functionality of the system during each sub-event and identify changes in functionality and flows between sub-events. A FACE analysis allows a designer to identify locations for establishing requirements and provides a means of identifying the measurement functionality required to assess the readiness of a system to transition between sub-events.

To illustrate the application of these steps, an example was performed on the landing leg sub-system of the MPL probe. This example demonstrates the application of event modeling, functional modeling and requirements identification to a spacecraft sub-system. In this example, locations for requirements were identified based on the transitions in functionality observed in the functional models. To further illustrate the application and usefulness of the FACE methodology, two case studies are included. The first case study is an application of FACE to the Columbia accident using standard functional modeling techniques. This case study demonstrates how the FACE methodology can model the functional propagation of a failure after it has occurred. The second case study demonstrates how FACE could have been applied to the MPL probe to prevent its failure.

3) Case Studies

3.1) Columbia

During the first stage of the January 16, 2003 launch of Columbia, a piece of insulating foam broke from the external tank and stuck the leading edge of the orbiter's left wing. It is believed that this impact caused significant damage to the RCC panels that protect the orbiter during atmospheric reentry. The foam strike was discovered by reviewing launch footage. The severity of the strike was assessed and based on previous experience with foam strike damage a decision was made to continue the landing as planned. Upon reentry into Earth's atmosphere, the damage to the RCC panel allowed superheated gas to penetrate the orbiter's thermal protection system resulting in a structural failure of the left wing and the ultimate loss of Columbia and her crew [2].

3.1.1) Event Modeling

The first step in the FACE analysis of the Columbia accident was to model the critical events that led to the accident. Two critical events were important in the Columbia disaster. The first event was the first stage of launch when the foam strike occurred. The second critical event was the orbiter's reentry into the Earth's atmosphere. The process used to create detailed event models for the accident follows.

During launch, the orbiter, the attached external tank and solid rocket boosters, the launch tower and atmosphere are all considered as material inputs to the event. All of these material flows are also outputs of the black box model. The energy inputs are the *chemical energy* stored in the solid rocket boosters (SRBs) and external fuel tank (ET), *translational mechanical energy* from gravity and *pneumatic energy* from the atmosphere. Within the black box, the *chemical energy* is transformed into the *translational* and *rotational energy* required to launch the orbiter as well as the *pneumatic energy* of the exhaust gases. During reentry, the orbiter and atmosphere are treated as material inputs and outputs of the event. The energy inputs were *translational energy* from gravity, the *translational* and *rotational energy* of the orbiter and *pneumatic energy* from the atmosphere. The energy outputs were *thermal energy* from the aerodynamic heating of reentry, *pneumatic energy* from the orbiter's thrusters, and the *translational* and *rotational energy* of the orbiter after reentry. For this analysis, no signal flows were included. The event names and flows are represented in the black box event models that appear in Figure 8 and 9.

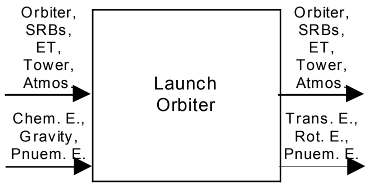

Orbiter, SRBs, ET, Tower, Atmos.
Chem. E., Gravity, Pnuem. E.
Launch Orbiter
Orbiter, SRBs, ET, Tower, Atmos.
Trans. E., Rot. E., Pnuem. E.

Figure 8 - Black Box Event Model During Launch

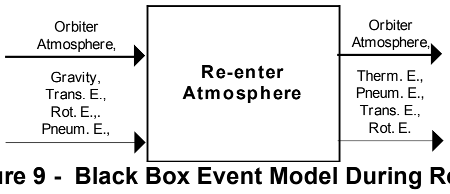

Orbiter Atmosphere,
Gravity, Trans. E., Rot. E.,. Pneum. E.,
Re-enter Atmosphere
Orbiter Atmosphere,
Therm. E., Pneum. E., Trans. E., Rot. E.

Figure 9 - Black Box Event Model During Reentry

The event sequence for the first stage of launch begins after the main engines have been ignited. The first sub-event in this sequence is the ignition of the solid rocket boosters. This sub-event is followed by the liftoff of the shuttle and its clearing of the tower. Once the shuttle clears the tower, it is oriented to its launch trajectory. This orientation is followed by a trajectory correction sub-event. During the trajectory correction event, two additional sub-events take place, a throttle down to reduce the aerodynamic stresses on the shuttle and a throttle up. To prepare for SRB cutoff, the shuttle enters a maintain attitude sub-event. This sub-event is followed by the separation of the SRBs. Once the SRBs have been separated, the first stage of launch is ended. This sequence of events is represent in the event model that appears in Figure 10. A timeline is included for reference [19]. As seen in Figure 10, the debris strike that damaged the RCC panels on the left wing occurred 81.9 seconds into launch during the trajectory correction event.

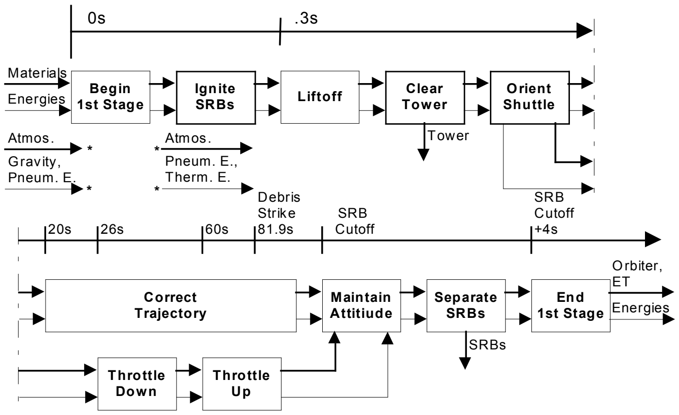


Figure 10 – Event Model of the First Stage of Launch

The sequence of sub-events for the reentry of the orbiter begins with an orientation sub-event followed by a deorbit burn. The next events are the end of the deorbit burn followed by an orientation for reentry. This orientation leads to the orbiter's entry into the atmosphere. Finally, the orbiter enters an aerobraking sub-event. It was during this aerobraking sub-event that the Columbia orbiter was destroyed. This sequence of events is represented in Figure 11.

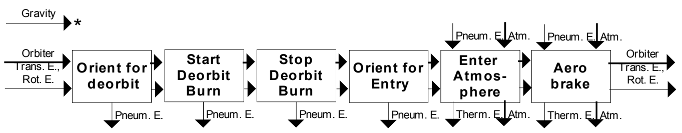


Figure 11 – Event Model of Reentry

3.1.2) Functional Modeling

In this example, the sub-system being analyzed is the thermal protection system, specifically the wing leading edge RCC panels. Black box functional models were created for the RCC panels during the maintain trajectory sub-event of launch and the aerobraking sub-event of reentry. During launch, the overall function of the RCC panels is to *guide gas* around the leading edge of the wing. In this sub-event, the RCC panel has material inputs of the wing structure, the atmosphere and any debris that may strike the panel. All three material inputs are also outputs from the event. The energy inputs are *pneumatic energy* from the atmosphere, and *mechanical energy* from any impacts. The energy outputs are *mechanical energy* in the form of reaction forces and *thermal energy* generated by the aerodynamic heating. The black box functional model for the RCC panels during the *correct trajectory* sub-event appears in Figure 12.

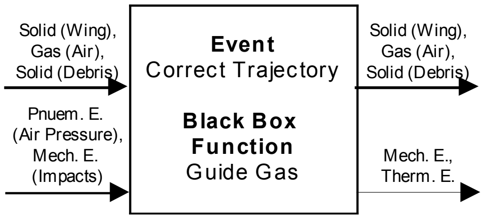

Solid (Wing), Gas (Air), Solid (Debris)
Pnuem. E. (Air Pressure), Mech. E. (Impacts)
Event
Correct Trajectory
Black Box Function
Guide Gas
Solid (Wing), Gas (Air), Solid (Debris)
Mech. E., Therm. E.

Figure 12 – Black Box Functional Model of the RCC Panels During Correct Trajectory Event

During the *reenter atmosphere* sub-event, the overall function of the RCC panels is to *inhibit* the flow of *thermal energy*. The inputs and outputs to this black box function are the same as the inputs and outputs to the RCC panels during the *maintain trajectory* sub-event. The black box model for the RCC panels during the *reenter atmosphere* event appears in Figure 13.

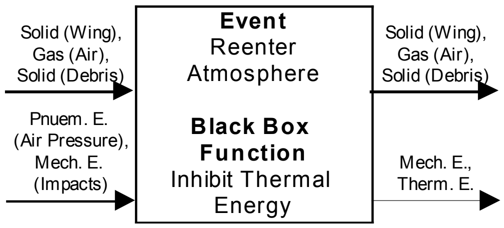

Solid (Wing), Gas (Air), Solid (Debris)
Pnuem. E. (Air Pressure), Mech. E. (Impacts)
Event
Reenter Atmosphere
Black Box Function
Inhibit Thermal Energy
Solid (Wing), Gas (Air), Solid (Debris)
Mech. E., Therm. E.

Figure 13 – Black Box Functional Model of the RCC Panels During Reentry Event

The next step in the functional modeling process was to create a detailed functional model for the RCC panels. To create this model, the input flows to the black box models were individually analyzed to create function chains. These chains were aggregated to produce the functional model that appears in Figure 14.

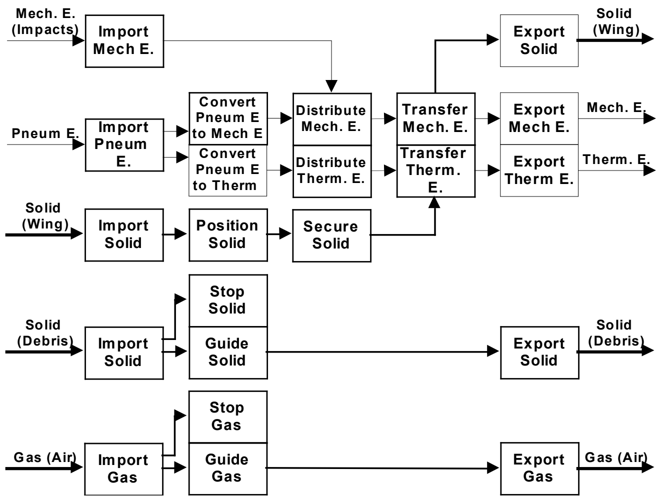


Figure 14 – Functional Model of RCC

3.1.3) Analysis of the Failure and Conclusions

The event and functional models created for Columbia were used to analyze the changes in flows and functionality that resulted from the foam strike between the first stage of launch and reentry critical events. During the first stage of launch, the *stop solid* function for the debris flow failed during the foam strike. Upon reentry, the loss of the *stop solid* function resulted in the loss of the *stop gas* function as well. The resulting flow of gas through the system caused an increased conversion to thermal energy which overwhelmed the *inhibit thermal energy* function. This ultimately led to an increased *thermal energy* flow to the wing, which resulted in a structural failure of the wing (the *secure solid* function) and the ultimate loss of the orbiter.

As shown in this example, the FACE methodology was used to represent the sub-events that must be completed for two critical events, the first stage of launch and re-entry, to be successful. Additionally, the functional models of the TPS system identified and captured the important functionality and flows during these two critical events. When combined, the event and functional models identified the origin of the failure (the material flow of the foam) and the propagation of the failure through the function and event flows. This case study demonstrates how FACE can be used to identify and model the functional propagation of a failure after is has occurred. By applying FACE to previous failures, it is possible to identify the specific functional causes of a failure. Identifying the functional causes and propagation of failures is necessary to prevent similar failures in the future.

3.2) FACE Example – MPL

The objective of this case study is to demonstrate how FACE could have been applied during the design of MPL to prevent the failure that ultimately led to the destruction of the craft. Contact with the Mars Polar Lander was never reestablished after its expected landing on Mars. The mishap review board concluded that the failure of the probe most likely resulted from impact damage due to the descent thrusters cutting off before touchdown. It was theorized that the thrusters were shut down by a spurious touchdown signal generated by the landing gear during deployment. During deployment at approximately 1500m above the Martian surface, false touchdown signals could be generated by the Hall effect sensors mounted on each landing leg (MPL had three landing legs). If these touchdown signals persisted through two sensor read cycles (at 100Hz), a flag would be set by the control software to indicate that the probe had landed. There was not a reset of the flag coded into the cycle. This allows a touchdown signal erroneously generated during landing gear deployment to persist until an event enable flag is set at approximately 40m above the surface. At this point, the persisting touchdown flag would trigger a shutdown of the descent engines causing the probe to fall the remaining distance and strike the ground with enough force to cause terminal failure.

It was determined that a breakdown in the systems engineering process resulted in the software condition that allows this particular failure. A system requirement was set to ignore the sensor data before 40m. This requirement did not, however, specify the particular failure mode (spurious touchdown signals from deployment) that originated the requirement. As a result, the software designers did not explicitly know that the requirement was proposed to eliminate the risk of spurious touchdown signals. For a complete description of this failure, see JPL D-18709, section 7.7.2 [1].

3.2.1) Event Modeling

The first step in applying FACE to the MPL touchdown monitoring system (TDM) would have been to create a black box event model. For this system, the critical event being analyzed is the landing of the craft on Mars. For this event, the name *Landing Sequence* was selected. Next, the input and output flows through this event were identified. The material flows into this event are the MPL itself and the surface of Mars. The material output flows from the event should also be the MPL and the surface of Mars. The energy flow inputs are the *mechanical energy* associated with the motion of the MPL and *translational energy* from gravity. The output energies are the *mechanical energy* from impacting the surface, *translational energy* in the form of reaction forces once the probe is stationary and finally, *pneumatic energy* from the probe's thrusters. An input signal to initiate the landing sequence and an output *status signal* are the only signal flows considered in this analysis. This critical event and its input/output flows are represented in the black box event model that appears in Figure 15.

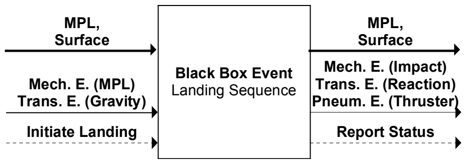


Figure 15 – MPL Landing Sequence Black Box Event Model

The next step in the FACE analysis for the MPL would have been to create a detailed event model for the system. As the probe begins its landing sequence, it is in a Parachute Descent. After Parachute Descent, the parachute is jettisoned and landing thrusters are used to complete the landing in a Terminal Burn event. The probe then lands on the surface of Mars (the Landing Event) and finally ends the landing sequence. At this point it reports its status. This sequence of events as well as the input and output flows for each event are represented in the event model that appears in Figure 16.

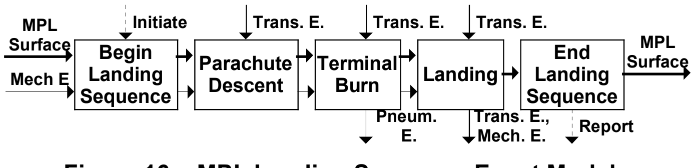


Figure 16 – MPL Landing Sequence Event Model

3.2.2) Functional Modeling

Once the event model was made for the MPL landing sequence, functional models could be made for the TDM system. Since the TDM system is software, not an electro-mechanical system, Functional Basis terms could not be used for the modeling. Instead, free language terms were used.

To continue the FACE analysis, a functional model must be made for the system during each event. The functional models for this system were made using system-level requirements for the touchdown monitoring software as well as the functional-flow diagrams generated during the design of the MPL [1]. This information would have been available during the design of the MPL. During the Parachute Descent event, the overall function of the TDM software is to maintain the load of processor (this is a system level requirement). During this phase of operation, the system is not actively checking for a touchdown but is running through the monitoring logic to prevent a significant increase in processor load once the system does need to check for touchdown. During the Terminal Burn phase, the overall function of the touchdown monitoring software is to check for a touchdown and shutdown the descent thrusters once it has been detected. The inputs and outputs of this function are a series of variables (represented as signals in the model). There are seven signals (see Table 3).

Table 3 – Touchdown Monitoring Signals

Name	JPL Name	Possible Values
Started	Touchdown Monitor	STARTED/NOT STARTED
Enable	Event Enabled	ENABLED/DISABLED
State	Indicator State	TRUE/FALSE
Health	Indicator Health	GOOD/FAILED
Last	Last Touchdown Indicator	TRUE/FALSE
Current	Current Touchdown Indicator	TRUE/FALSE
Input	I/O Card Discrete	TRUE/FALSE

The overall functions and input and output signals during the Parachute Descent and Terminal Burn events are represented in the black box functional models that appear in Figures 17 and 18 respectively.

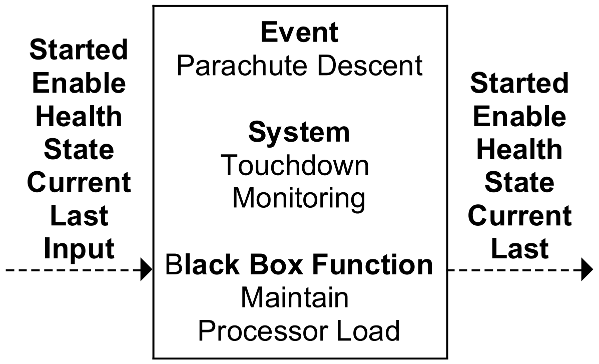


Figure 17 – TDM Black Box Functional Model During Parachute Descent

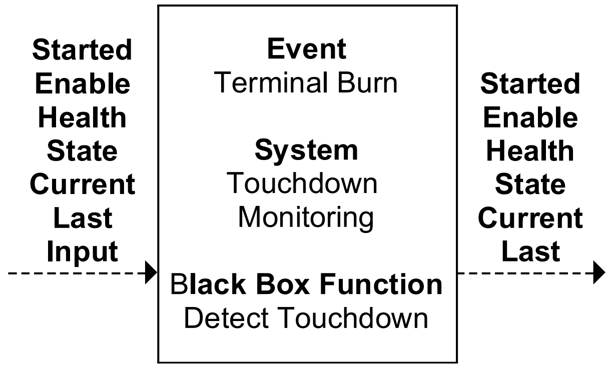


Figure 18 – TDM Black Box Functional Model During Terminal Burn

Once black box functional models have been made, detailed models can be made. For the TDM system, JPL functional-flows diagrams were used as the basis for creating the functional models of the system. During both the Parachute Descent and Terminal Burn events, the detailed functionality of the TDM software is identical (the same software is running during each event, a change in the Enable variable is the only difference between the two events). The functional model for the TDM system during both events appears in Figure 19.

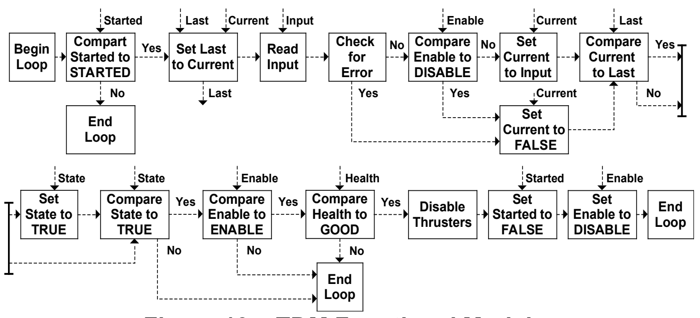


Figure 19 – TDM Functional Model

3.2.3) Requirements Identification

The final step in the FACE analysis of the MPL TDM system would have been to identify requirements and additional required functionality using the event and functional models. The first step in this process would be to identify the functions and flows that undergo transitions between events. Next, requirements should be established for the values of the flows and functionality should be added to the model to ensure that flows will meet the required values. A list of the important flows in the TDM software along with a set of required values at the transition between events appears in Table 4. These flows were known during the design of the TDM software. The required values for the flows at the transitions between events were generated from an analysis of the intended functionality of the TDM system.

Table 4 – Required Variable Values During Transitions

	Event Transistions			
Signal Flows	Begin Landing/ Parachute Descent	Parachute Descent/ Terminal Burn	Terminal Burn/ Landing	Landing/ End Landing
Started	FALSE	TRUE	TRUE	FALSE
Enable	Not defined	DISABLE	ENABLE	DISABLE
Health	Not defined	GOOD	GOOD/FAILED	GOOD/FAILED
State	Not defined	FALSE	TRUE	TRUE
Current	Not defined	Not defined	TRUE/FALSE	TRUE/FALSE
Last	Not defined	Not defined	TRUE/FALSE	TRUE/FALSE

As shown in Table 4, at the beginning of the Parachute Descent event, the only flow that needs a requirement is the Started variable. During Parachute Descent, the TDM software is started (Started is set to STARTED). During startup, several variable are initialized including Enable (set to DISABLE), Health (set to GOOD) and State (set to FALSE). During Terminal Burn, the Started variable should maintain its value of TRUE, the Enable variable should change to ENABLE, the Health variable should stay GOOD but may transition to FAILED if a failure is detected, the value for State should transition to TRUE (to indicate a landing) and the values of Current and Last should be TRUE (but may be FALSE if a landing sensor has failed or that specific leg did not touchdown first). Once the probe has landed, the Started variable should be set to FALSE and Enable set to DISABLE (the remaining variables should keep their values).

To ensure that the landing sequence occurred as planned, the functionality of the system should have been designed so that each flow was at its required value during the transition between events. For all flows except the State variable, this was the case. During the design of the MPL, it was known that the touchdown sensors could generate a false positive signal during deployment of the landing legs (in the Parachute Descent event). This false positive value would cause the State variable to be set to TRUE during descent (even though the Enable value is FALSE). As designed, if the State variable value was TRUE at the transition between the Parachute Descent and Terminal Burn events (there wasn't an explicit requirement for it to be FALSE) this value would propagate to Terminal Burn. As soon as the Terminal Burn event begins, the value for the Enable flow is ENABLE (this occurs at 40m above the surface) and the State value is TRUE. As long as the Health value is GOOD, the TDM software will disable the descent thrusters. The result would be a free fall for the remaining 40m to the surface. This scenario was determined to be the most likely cause of the destruction of the MPL probe. To prevent such a failure, a requirements table (such as the one shown in Table 4) should have been generated during the design of the TDM software. The software should have been designed in such a way to prevent the State variable's value from passing between Parachute Descent and Terminal Burn. To implement this change, additional functionality needs to be added to the model. The actual fix implemented by JPL in subsequent probes using the same TDM software was to add one additional function to the TDM loop that resets the value of State during each cycle.

3.2.4) MPL Example Discussion

The destruction of the MPL probe was most likely caused by the propagation of a spurious touchdown signal between the Parachute Descent and Terminal Burn events. The failure resulted from a lack of requirements for the specific values of variables at the transitions between events. This example demonstrate how a FACE analysis of the TDM system would have led to the generation of a requirements table for the variables of the TDM software. By comparing the required values to the values that could be produced from the designed functionality of the system, it was possible to identify missing required functionality (a function to reset the State variable).

3.3) Discussion of Case Studies

The two case studies presented show how FACE can be applied to identify and model the functionality of a system during critical mission events. The models generated during the application of FACE were used to identify changes in functionality between events and were used to trace how changes in functionality during an event propagate to later mission events. In the Columbia example, a functional model of the RCC panels of the Thermal Protection System was created to show how the effect of the debris strike propagated between launch and reentry. For the MPL case study, the functional models and event models illustrate how a spurious signal propagated between the two events that caused the failure of the probe. Both examples represent applications of FACE to past systems that have failed. For the FACE methodology to be successfully utilized, it must be applied during the design of a future system.

4) Conclusions and Future Work

Identifying and capturing the functionality of a complex system across critical events is an essential step in ensuring the successful completion of missions. The FACE methodology represents an opportunity to improve current critical event modeling and handling procedures at NASA. We propose to improve upon current critical event handling procedures by introducing structured functional and event modeling processes along with standard taxonomies for representing functionality. These improvements are implemented through the application of FACE to the design of a system. Additionally, as shown in the two case studies, FACE can applied after an accident has occurred as a means to identify and model the functional causes of an accident. By modeling the functional causes of previous accidents and applying the lessons learned through the application of FACE to new systems design, it is possible to reduce the occurrence of such accidents in these new systems.

This methodology has been presented along with three examples. The first example illustrates the application of the FACE methodology to a simple mechanical system, the landing legs of the MPL probe. The second example was an application to the Columbia disaster, the accident that prompted the research. Finally, the FACE methodology was applied to the software problem that likely resulted in the loss of the MPL probe. These examples illustrate the application of the FACE methodology to various aerospace systems.

Currently, several opportunities exist for future research in this area. As developed, the method is focused on ensuring that the system exhibits a nominal sequence of critical events. Eventually the method needs to be able to model and analyze the functionality of a system once it has deviated from the nominal event sequence. Additionally, as shown in the MPL example, a formal taxonomy for modeling the functionality of software systems needs to be investigated. Finally, research into integrating the method with current event tree, fault tree and hazard analyses should be done to reduce the amount of overlap between the methods.

A FACE analysis supplements and extends current failure analysis methods. Ideally, the nominal event sequence generated during a FACE analysis should be used as a starting point for creating an event tree. Additionally, the functional model can be used as a starting point for creating a fault tree for a system. Initial work has been done by the authors on the latter task. During retroactive investigation of failures, the event and fault trees created for a system during the system's design would be useful input for performing a FACE analysis of the failure.

Acknowledgements

The authors would like to acknowledge Sandra Hayden, Francesca Barrientos and the rest of the Complex Systems Design group at NASA's Ames Research Center for their assistance with this work. This work is supported by the NASA Ames Research Center under grant NCC 2-1380. Any opinions or findings of this work are the responsibility of the authors and do not necessarily reflect the views of the sponsors or collaborators.

References

[1] JPL Special Review Board (2000), "Report on the Loss of the Mars Polar Lander and Deep Space 2 Missions," JPL D-18709.
[2]The Columbia Accident Investigation Board (2003), "Columbia Accident Investigation Board Report."
[3]Cassandras, C. and Lafortune, S. (1999), Introduction to Discrete Event Systems, New York, Springer.
[4]Hayden, S.C. and Tumer, I.Y., "Preliminary Report on Mission Design and Operations for Critical Events", NASA #362EF1 2005.
[5]Jet Propulsion Laboratory (2003), "Flight Project Practices," D-58032, Rev 5.
[6]Jet Propulsion Laboratory (2003), "Design, Verification/Validation, and Operations Principles for Flight Systems," D17868, Rev 2.
[7]Jet Propulsion Laboratory (2005), "JPL Anomaly Resolution," D-8091.
[8]Jet Propulsion Laboratory (2001), "JPL Reliability Assurance," D-8671.
[9]Goddard Space Flight Center (2004), "Rules for the Design, Development, Verification, and Operation of Flight Systems," GSFC – STD – 1000.
[10]Kenny, T. and Carr, J. (2004), "Techniques and Strategies for Reducing Long-Term Operations Cost/Risk During Early Vehicle Design," Mission Operations Directorate, NASA JSC.
[11]Otto, K. and Wood, K. (2001), Product Design: Techniques in Reverse Engineering, Systematic Design, and New Product Development, New York, Prentice-Hall.
[12]Hirtz, J., Stone, R., McAdams, D., Szykman, S. and Wood, K. (2002), "A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts," Research in Engineering Design, 13(2): 65-82.
[13] Tumer, I., Stone, R. and Roberts, R. (2002), "Decomposition-Based Failure Mode Identification Method for Risk-Free Design in Large Systems," Submitted to Journal of Mechanical Design.
[14] Bohm, M., Stone, R. and Szykman, S. (2003), "Enhancing Virtual Product Representations for Advanced Design Repository Systems," Accepted to Journal of Computer Information Science in Engineering.
[15]Hutcheson, R. and Tumer, I. (2005), "Function-Based Design of a Spacecraft Power Subsystem Diagnostics Testbed," Proceedings of the ASME International Mechanical Engineering Congress and RD&D Expo, IMECE2005, Orlando, FL.
[16]Hutcheson, R. and Tumer, I. (2005), "Function-Based Co-design Paradigm for Robust Health Management," 2005 Stanford International Workshop on Systems Health Management, Palo Alto, CA.
[17]Kumamoto, H., and Henley, E. (1996), Probabilistic Risk Assessment and Management for Engineers and Scientists, New York, IEEE Press.
[18]Bedford, T., and Cooke, R. (2001), Probabilistic Risk Assessment: Foundations and Methods, Cambridge, Cambridge University Press.
[19]National Aeronautics and Space Administration (2000), "Mission Events Summary," http:// science.ksc.nasa.gov/ shuttle/technology/sts-newsref/sts_mes.html