# Conceptual Design of Sacrificial Subsystems: Failure Flow Decision Functions[[1]](#footnote-1)

Ada-Rhodes Short[[2]](#footnote-2), Ann D. Lai2, Douglas L. Van Bossuyt2

## Acknowledgements

This research was partially supported by United States Nuclear Regulatory Commission Grant Number NRC-HQ-84-14-G-0047. 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. The authors wish to acknowledge the work of the undergraduate research assistants in the Van Bossuyt lab and specifically wish to thank the following students for their contributions: Alexis Humann, David Hodge, Zachary Mimlitz, and Robin Coleman. The authors wish to thank LeVar Burton, Fred Rogers, Gene Roddenberry, Carl Sagan, and their individual middle school and high school science and technical arts teachers who inspired them to pursue careers in the sciences and engineering, and instilled in them a sense of purpose and compassion.

## Abstract

This paper presents a method to conceptually model sacrificing non-critical subsystems, or components, in a failure scenario to protect critical system functionality through a functional failure modeling technique. Understanding the potential benefits and drawbacks of choosing how a failure is directed in a system away from critical subsystems and toward subsystems that can be sacrificed to maintain core functionality can help system designers to design systems that are more likely to complete primary mission objectives despite failure events. Functional modelling techniques are often used during the early stage of conceptual design for complex systems to provide a better understanding of system architecture. A family of methods exists that focuses on the modelling of failure initiation and propagation within a functional model of a system. Modelling failure flow provides an opportunity to understand system failure propagation and inform system design iteration for improved survivability and robustness. Currently, the ability to model failure flow decision making is missing from the family of function failure and flow methodologies. The Failure Flow Decision Function (FFDF) methodology presented in this paper enables system designers to model failure flow decision making problems where functions and flows that are critical to system operation are protected through the sacrifice of less critical functions and flow exports. The sacrifice of less critical system functions and flows allows for mission critical functionality to be preserved, leading to a higher rate of mission objective completion. An example FFDF application in a physical design is a non-critical peripheral piece of electrical hardware being sacrificed during an electrical surge condition to protect critical electronics necessary for the core functionality of the system. In this paper, a case study of the FFDF method is presented based on a Sojourner class Mars Exploration Rover (MER) platform.

Key Words: Functional Modelling, Failure Modelling, Failure Flow Decision-Making, Design Methodology

## 1 Introduction

Complex systems such as Mars Exploration Rovers (MERs) or airborne, aquatic, or terrestrial autonomous vehicles often operate in hazardous environments in which servicing or repairing damaged or broken hardware is infeasible or impossible due to high costs, isolation of the environment, or extreme conditions leading to loss before opportunity for repair. Low-cost systems such as hobby aerial drones or aquatic Remote Operated Vehicles (ROVs) can be replaced easily, but unique and expensive systems such as a MER are difficult and time- and resource-consuming to replace when the system fails. Therefore, it is critical to deploy systems capable of surviving in their operating environments and accomplishing the mission over the intended duration. Significant attention is often given to understanding system failure risk and mitigation of potential failures during system design, but risk analysis often occurs after major system architecture decisions have been made and the system design has already progressed to a point in which system design changes are prohibitively costly and time-consuming. Performance of risk analysis in the early conceptual stage of the design process before major architectural decisions are made can contribute to system designs that have a higher probability of surviving the intended mission without failure, without costly design changes, and without compromising other aspects of the design or mission.

In the past two decades, a variety of work has been performed to analyze the risk of system failure during operation at the conceptual stage of design. NASA’s Team-X (“JPL Team X” 2016) and other groups have developed methods for assessing risk of system failure at the conceptual stage of mission design through trade-off studies for risk analysis. Other groups have focused on the development of functional system modeling to assess risk of failure (Kurtoglu and Tumer 2008). Functional modelling has become a useful tool for conceptual system design and has been used to simulate a system’s response to an environment (Short and Van Bossuyt 2015b). Multiple methods have been developed in the last decade to analyze risk of system failure using functional models (Kurtoglu and Tumer 2007; Jensen, Tumer, and Kurtoglu 2009; Kurtoglu and Tumer 2008).

This paper presents the Failure Flow Decision Function (FFDF) method that provides insight into decisions when designing or operating systems facing failure. FFDF uses the Failure Flow Identification Propagation (FFIP) (Kurtoglu and Tumer 2008) method and Flow State Logic (FSL) (Jensen, Tumer, and Kurtoglu 2009) to perform system failure analysis on a functional model, allowing FFDF to determine what sub-system failures are more likely to cause critical system failure. FFDF builds upon the foundation of the Functional Basis for Engineering Design (FBED) (Stone and Wood 2000) as a convention for the representation of functional models of systems and proposes the addition of the Direct Failure function to the FBED lexicon. The Direct Failure function is representative of systems that allow for an action to be taken by a system that mitigates critical system failure through the sacrifice of sub-systems that are less critical to total mission success. An example of a Direct Failure function realized in physical components is a circuit that directs electrical surges to particular systems, a flood routing system downstream of a reservoir, or components that allow for a system to reorient itself to incur a physical impact in the safest way possible. A second example is a humanoid robot designed to carry a payload that is deemed critical to the robot’s function up and down stairs; if the robot falls it may stick out an arm to protect the payload. Acting increases the probability of damaging an arm, but reduces the probability of damaging the payload. Directed Failure functions can describe a wide variety of systems used in autonomy to mitigate failure. FFDF has applications in the design of more robust systems for operation in hazardous environments as well as the ability to inform control and operation of autonomous systems (Short and Van Bossuyt 2015b; Short, Mimlitz, and Van Bossuyt 2016; Short, Mimlitz, and Van Bossuyt 2016; Mimlitz, Short, and Van Bossuyt 2016).

### 1.1 Specific Contributions

This paper proposes a method for the analysis of system survivability informed by functional modelling. FFDF allows for rapid analysis of a system for survivability at the conceptual stage prior to making large architectural decisions. The result of FFDF is a set of values representing the criticality of individual components to system survivability that is used to inform the sacrifice of sub-systems when necessary during a failure scenario for the improved survivability of the system. When applied to autonomous systems such as exploratory rovers, FFDF can improve survivability by informing which functions can be sacrificed if necessary for system survival. The work also lays the groundwork to building more complex autonomous systems by providing an informed basis for decision making when facing failure.

## 2 Background

The FFDF method builds upon foundational work from several related fields including decision theory, failure analysis methods, and system and functional modelling. This section discusses topics pertinent to the understanding and implementation of FFDF.

### 2.1 System and Functional Modeling

System and functional modeling refers to a suite of tools and methods for conceptualizing complex systems and their functionality with a wide variety of applications for analysis and system design. Several established methods perform system modelling through the definition of functions and flows in a system, where a function represents a process performed by the system or sub-systems and flows represent energy, materials, or information passed between and worked on by the functions; additionally, there are import and export flows that enter or exit the system boundary. Several common techniques used in systems engineering applications include Functional Flow Block Diagrams (FFDB) (Lightsey 2001; Blanchard and Fabrycky 1990), Enhanced Function Flow Block Diagrams (EFFBDs) (Long 2002; David, Idasiak, and Kratz 2010), Integrated Computer Aided Manufacturing (ICAM) DEFinition for Function Modeling (IDEF0) (Force 1981), Systems Modeling Language (SysML) (Huang, Ramamurthy, and McGinnis 2007), and Functional Basis for Engineering Design (FBED) (Hirtz et al. 2002; Stone and Wood 2000). The FFBD method is useful for modeling systems in which there are direct linear flows passed between functions and a clear input and output exists for the system. However, FFBD is limited in its ability to model systems with more complex flow paths, which makes it difficult to accurately represent the system. EFFBD takes an alternative approach that instead models the behavior and physical properties of a system via the passing of states and information. EFFBD implements this approach through modelling mechanical and control system behavior. While EFFBD can offer benefits for system design and development of controls, EFFBD is limited as a platform for performing analysis of system health resulting from the lack of inclusion of sub-systems for monitoring system health. SysML is a flexible form of Unified Modeling Language (UML) (Rumbaugh, Jacobson, and Booch 2004) that is specific to system engineering. While SysML is used extensively for certain aerospace applications, its rigid rules set provides a significant barrier to adoption and little work has been done to analyze risk in the conceptual stage of design. FBED uses an established lexicon (Hirtz et al. 2002) to represent functions and flows, can be used to develop models of most engineered electrical and mechanical systems, and is particularly well-suited to modeling systems in the conceptual stage of design. For example using the FBED lexicon a battery could be represented by a Store Energy function with an Electrical Energy flow connecting the battery to powered systems. A more complex example is a drive train for an electrical vehicle, in which a motor control board delivers power to electrical motors, which go into gear boxes, which in-turn drives wheels. This system could be represented by a Control Magnitude Electrical function importing Digital Signal and Electrical Energy flows and exporting Electrical Energy flows into Convert Electrical-to-Rotational Energy functions representing the motors. A Rotational Work flow leaving the Convert Electrical-to-Rotational Energy function enters a Transmit Rotation function representing the gearbox, which exports a modified Rotational Work flow. Finally the Rotational Work flow enters a Convert Rotation-to-Translation representing a wheel and then exports Translational Work.

The work presented in this paper uses FBED for functional modelling which was selected over other methods because of its ability to represent a wide variety of complex systems, with a myriad of available failure analysis methods, the well-developed conventions and practices, existing functional component design repositories, and the support of FBED by the National Institute of Standards and Technology (NIST) (Bohm, Stone, and Szykman 2005; Materese 2002). The FBED lexicon was established after extensive study by Stone and Wood and consolidated previous functional model lexicons into a single standard (Hirtz et al. 2002; Stone and Wood 2000).

Surveying the existing work on the generation of functional models, two common approaches exist, the first of which involves the development of a model to represent an existing system through functional decomposition (van Eck, McAdams, and Vermaas 2007). In functional decomposition, the system is decomposed into component sub-systems and the functionality of each sub-system is then determined. Sub-system functionality can be represented as function blocks and flow paths, and a model can be developed. Functional decomposition is common when developing models of existing systems for the purpose of analysis. The second common approach generates functional models from the desired functionality of a system with no existing physical system design. Under this approach, desired system import and export flows are often determined and modelled as passing through an unknown “black box”. Necessary sub-systems are then defined to represent processes necessary to achieve the desired export flow and functionality, and functions can then be selected to represent the sub-systems. The developed functional model is then used to design the physical system by selecting components that satisfy functions (Bohm, Stone, and Szykman 2005; Stone and Wood 2000; Hirtz et al. 2002). Under both approaches to functional modelling, analysis of the system can be performed through various methods, and physical components can be selected in order to serve a defined functional purpose in the system. This paper presents a case study of a functional model of a sojourner class MER that is analyzed through FFDF for improved system survivability. While this work studied a model developed through functional decomposition, FFDF can also be used on functional models that were not based on existing systems.

### 2.2 Failure Analysis Methods

A variety of failure analysis methods are currently used in industry and academia. Common methods include: Probabilistic Risk Assessment (PRA) (Kumamoto, Henley, and J 1996; Stone and Wood 2000) which is heavily used in aerospace, defense, and civilian nuclear power; Reliability Block Diagrams (RBD) (Distefano and Puliafito 2007) which determine system reliability by modelling parallel and serial failure flow paths through blocks containing statistical reliability data; Failure Modes and Effects Analysis (FMEA) (Mohr 2002) which has been used across a broad spectrum of industries since its development in the 1950s for military applications; and Fault Tree Analysis (FTA) (Ericson 1999) which was developed by Bell Labs in the 1960s for analysis of aerospace systems. While PRA, RBD, FMEA, and FTA provide a good foundation for analysis, they are limited in their ability to model failure, generate practical models, and be implemented. Several methods have been developed for failure analysis of FBED models (Stone and Wood 2000). Function-based Analysis of Critical Events (FACE) (Hutcheson et al. 2006) enables designers to modify functional models informed by critical events during a complex system’s lifecycle. The Function Failure Design Method (FFDM) (Stone, Tumer, and Van Wie 2005) melds FMEA and FBED in order to create a method for selecting functions and component solutions informed by past, collected, or historical failure data. Function Failure Identification Propagation (FFIP) (Kurtoglu and Tumer 2008; Kurtoglu and Tumer 2007) models the flow of failure through FBED-based functional models as it passes between functions along flow paths. Function Failure Reasoning (FFR) (Kurtoglu, Tumer, and Jensen 2010) provides a simulation tool for modelling FFIP in a complex system. The Flow State Logic (FSL) (Jensen, Tumer, and Kurtoglu 2009) method refines FFIP and provides a representation of the analyzed system’s complete failure state. The Uncoupled Failure Flow State Reasoner (UFFSR) (O’Halloran, Papakonstantinou, and Van Bossuyt 2015) method improves on FFIP by including analysis of failures that do not propagate along nominal flow paths but rather through physical space between functions. Similarly to FFIP, Functional Dependency Network Analysis (FDNA) expands upon FMEA to anaylze complex systems, systems of systems, and enterprise systems (Garvey and Pinto 2009; Garvey, Pinto, and Santos 2014). FDNA builds on Inoperability Input-Output Models (IIM) (Haimes et al. 2005), Design Structure Matrices (DSM) (Browning 2001), Leontief Systems, and FMEA, and uses directed graphs consisting of nodes representing capabilities (similar to functions described above) and edges representing dependencies between modes. While FDNA analyzes unidirectional systems such supply chains or other enterprise systems, FDNA is limited in its ability to analyze systems in which failures can propagate against the flow of input-output relationships, a restriction not shared by FFIP. System Operational Dependency Analysis (SODA) (Guariniello and DeLaurentis 2017) improves upon FDNA by considering the internal status of systems and better accounts for stochasticity. Despite its benefits in efficiency of analysis and ability to support decision making, SODA takes a similar approach to FDNA regarding failure propagation; neither method has an ability to account for failures that propagate backwards against the direction of dependency. An advantage of SODA and FDNA over FFIP-based methods is that they consider non-binary failure states and can express operability of nodes on a scale from 0 to 100. The ability to express nuanced levels of operability is very helpful for systems that are repairable or can undergo scheduled maintenance, but becomes trivial in the case of remote systems that cannot be repaired such as those considered in this paper. While significant progress has been made in the past decade, many areas still need to be addressed and further development of existing methods is needed.

### 2.3 Hazard Rate Estimation and Success Assessment

A common method for analyzing the likelihood of a system surviving a desired mission is calculating the system’s hazard rate. A hazard rate is a probabilistic representation of expected system failures over a unit of time. Estimation of risk using a hazard rate is commonly performed using an exponential distribution (Wertz, Everett, and Puschell 2011a) to evaluate the expected survivability rate at time , as seen in Equation 1.

(1)

From the expected survival rate in Equation 1, the expected failure rate can be found through subtraction of from 1 as shown in Equation 2.

(2)

Under this convention, the survivability rate is the probability that a system with hazard rate will still be functional at time . Similarly, the failure rate describes the probability that a system will have failed by time . Modelling survival and failure rates is critical to the construction of a mathematical representation of failure in a functional model.

The FFDF method builds upon a foundation of system and functional modelling, failure analysis, hazard rate estimation, and decision theory. FFDF applies these concepts to enable informed decision making when facing failure, and enhances the effectiveness of system physical and control design.

## 3 Methodology

The Failure Flow Decision Function (FFDF) method presented in this paper is a tool that performs failure analysis on a functional model in order to inform failure flow routing decisions in the operation of a system for reduced probability of system failure over the duration of a mission. FFDF consists of eight steps separated into three distinct phases: Phase 1) Generation of the functional model, Phase 2) Performance of failure flow analysis, and Phase 3) Interpretation of results. Figure 1 shows the eight steps of FFDF. The FFDF method starts with the creation of a functional representation of a system of interest using the FBED lexicon (Steps 1-4 in Phase 1). Analysis is then performed on the developed functional model in order to determine the probability that the failure of an individual function leads to failure of a critical function (Steps 5-6 in Phase 2). The results of the analysis can then be interpreted and a Direct Failure function can be inserted into the functional model in the optimal position for system survivability (Steps 7-8 in Phase 3). FFDF is aimed towards the early conceptual stage of system design and allows for the design and operation of systems that are particularly adept at operation in environments where repairs are impractical or impossible.

### Phase 1: Generate Functional Model

#### Step 1: Develop Functional Model

The first step of FFDF is the generation of a functional model of the system of interest. The FBED lexicon can be used (Hirtz et al. 2002). The process begins by defining high-level functions to represent all major functionality of the system being modelled. The functions are then connected with flows, as shown in Figure 2, where thick solid lines represent the passing of physical material, thin solid lines represent the passing of energy such as electricity or heat, and dotted lines represent the passing of information or signals between functions. Figure 2 represents a simplified functional model of a Sojourner class MER. A more complex functional model of a Sojourner class MER used in the case study is shown in Appendix 1.



Figure 1: Process Flow of FFDF as 8 Steps



Figure 2: Example of a simplified functional model of a Sojourner-based Mars Exploration Rover. Boxes represent functions, thick lines represent material flows, thin lines represent energy flows, and dotted lines represent signals or information.

#### Step 2: Define Critical Functions and Flows

The second step of the FFDF method is the identification of critical flows and functions in the system of interest. Previous works define critical functions as a set of design functions that significantly represent the functionality of a system of interest (Lucero et al. 2014). FFDF uses a modified definition in which critical functions are elements of the functional model that must perform their intended function. There are two major classes of critical functions allowing for numerical and probability analysis of the functional model: 1) independent critical functions (ICFs), and 2) critical functions (kNCFs).

ICFs are individual functions that, in a failure state, lead to loss of functionality in the entire system. An example of an ICF for the case of a MER is the communications antenna represented by a Transmit Data function. The loss of the Transmit Data function leads to the MER losing the ability to communicate with Earth-based human operators, leading to the MER no longer being able to operate, even if all other MER functions have not experienced failure. Thus, Transmit Data is defined as a critical function of the MER.

kNCFs, the second class of critical functions, are sets of functions of size in which any subset of at least size must be functioning nominally for a system to not be in a critical failure state. An example of kNCFs in a MER are the three batteries represented by Store Energy functions. The MER functional model has *N=3* Store Energy functions, and it requires at least Store Energy functions to be operating nominally in order for the system not to be in a failure state. Typically, the set of critical functions for a functional model of a system will consist of multiple kNCFs and ICFs.

#### Step 3: Assign Failure Probabilities

The third step in FFDF is assigning failure probabilities for each function and flow in the functional model. This includes the probability of a function failing, of the failure passing along a flow path, and that a function on the other end of the flow path will accept the failure. The use of separate passing and accepting probabilities allow for the model to account for failures to occur and attempt to propagate, but allow for scenarios where the failure propagation does not affect the next function. For example a failure originating at a Distribute Electrical function and propagating along an Electrical Energy flow will always have the same probability of passing along that flow, but different functions have varying probabilities of accepting that failure. A component level example of this would be the system response in to a momentary voltage drop in an autonomous system. Driving motors are unlikely to be affected, but on board computers may reset. The calculation of the probability of failure propagating along a specific flow path using these values is described in Step 6.

Selecting appropriate probability values is one of the most critical steps in the development of a system model that is representative of reality. Generic failure probabilities can be sourced from texts and data sets appropriate to the system being modelled. For example, the probabilities used in the MER case study presented in this paper are based on values derived from (Wertz, Everett, and Puschell 2011b), which contains data on Mars rovers and other space systems, gathered experimentally from testing, or interpolated from similar systems already in operation, such as from PRA analysis of conceptual designs of nuclear power facilities (Gosselin 2006).

The failure probabilities are then stored. This paper advocates storing failure probability for efficient machine-searchability and expandability. Appendix 2 lists failure probabilities for system flows and functions used in the case study presented in this paper.

#### Step 4: Convert Functional Model into a Mathematic Representation

Before the functional model can be analyzed, it needs to be converted from a graphical representation that is human-readable to a machine-readable mathematical representation (Short and Van Bossuyt 2015a; Papakonstantinou et al. 2012; Sen, Summers, and Mocko 2013). In FFDF, functional models are represented as a cell array containing a string and two matrices. The string lists the class of functions represented by the cell. The first matrix contains flows into the function in the first column and the function at which the flow terminates in the second column. The second matrix has the same structure as the first, but instead tracks flows out of the function. This structure allows for the tracking of failure flow during the analysis phase of FFDF. The cell array contains the entire functional model in a format that is machine-readable. An example subsection of a functional model of a MER is displayed below in Model 1. In the model *f*0-*f8* refer to flow types, and the numbers refer to function block index values. A complete list of the values can be found in Appendix 3.

|  |  |  |
| --- | --- | --- |
| **Function** | **Flows In** | **Flows Out** |

Model 1: Sample Functional Model

### Phase 2: Design Iteration

#### Step 5: Define All Flow Paths for Machine Readability

One property of functional model analysis based in FFIP is that flow paths can propagate failure along parallel paths as well as in the opposite direction to a flow’s nominal path. A component-level example of a parallel failure is an over-voltage failure travelling along multiple parallel wires to several electrical devices. A component-level example of failure flows propagating in the reverse direction of a nominal flow path is a water hammer traveling through plumbing. Properly modelling this behavior in a functional model is important to accurately analyze system failure probabilities. For the purpose of FFDF, a functional model is best represented as a specialized network of nodes. While this structure allows for the accurate representation of a functional model, models with a small number of nodes can become very complex due to the large number of paths between nodes. An interesting computational problem arises from the increased complexity of the network representing the functional modelling as many traditional methods for solving failure flow paths are impractical due to the large computational resources needed.

The case study performed in this paper adopts a modified biased wall-following, maze-solving algorithm (Yadav, Verma, and Mahanta 2012) to navigate the functional model, while recording the path as it is traversed. The method is limited to taking only legal paths which is defined as paths that are acyclic and stay within the system model boundary, without passing through the operating environment in which the system is placed. In this implementation of FFDF, the functions and all connecting paths are put into increasing numerical order corresponding to their position in the functional model (see Step 4, Model 1 above). The lowest-ordered legal function in the model is navigated to, along the lowest-valued path. A function can be legally navigated to if it has not already been “visited” along the failure path, making all legal paths acyclic. Only acyclic paths are considered for FFDF because the method applies to unrepairable systems, where the first failure of a critical function is considered enough to fail the entire system. Additionally, legal paths for most cases will be constricted to not include travelling through the operating environment as this would involve the failure propagating through space in a potentially uncoupled manner such as in the case of UFFSR analysis (O’Halloran, Papakonstantinou, and Van Bossuyt 2015). This process is continued until no more legal paths are available, at which point the whole path is recorded and the algorithm returns to the last function with a legal path that has not yet been navigated. This is repeated until no more legal paths exist and all failure paths have been discovered.

The functional model flow path solving algorithm used here is computationally simple, relatively fast, and enumerates all possible paths. Other techniques were attempted, such as using a Monte Carlo method to generate paths through Bayesian sampling, but this was computationally expensive in order to reach a comparable level of resolution (Short and Van Bossuyt 2015a). Topological sorting was inapplicable due to inclusion of multidirectional flows (Kalvin and Varol 1983). A brute force method which generates a list of all potential paths, ignoring flow logic, and then checking them against the model for legality was feasible for small models but required prohibitive quantities of computer memory for larger models due to factorial growth of the memory required. Future work will explore quantifying and improving the efficiency of the flow path solving algorithm. One potential drawback of this method is that it is restricted to only acyclic failure paths and does not consider magnitude of failure. As a result, the model does not address the potential for non-failure inducing loops developing in the system that eventually lead to failure. While these cyclic paths are inherently less likely to occur than acyclic paths, future work will attempt to identify and address them.

#### Step 6: Calculate System Critical Failure Probability from Initiating Events

Calculating the system critical failure probability from initiating events determines how the failure of each individual function in a system’s functional model can lead to critical system failure through propagation of failure flows. For the purpose of the case study presented in this paper, it is assumed that the initiating failure event has occurred and therefore the probability of an initiating event failure is 1, representing a 100% chance of failure for ease of understanding the FFDF method. In practice, initiating event probabilities are selected to reflect real-world operating conditions of the system. The simplification allows for the paper to focus on the FFDF method without digressing into the minutiae of an environmental model.

An algorithm, shown in the formulation section below, calculates system failure probability for each failure flow path, starting with the selected initiating event. If a failure flow path does not contain any critical functions, it is skipped and the next path is selected. For cases in which the selected path contains an independent critical function, or ICF, the probability that the ICF will experience failure is calculated by taking the product of all probabilities of occurrence for all previous events in the path. Each event along the path is defined by two probabilities: , which describes the probability of a function in position in the flow path passes failure to the next function in the path, and which describes the probability that the function in position of the path accepts a failure passed to it from the previous function in the path. This calculation is performed for all ICFs in the failure flow path and can be seen in Equation 3 of Formulation 1.

If there is a kNCF in the failure flow path, a different calculation is used. The algorithm first calculates the probability of failure for every kNCF function in the path using the same method for finding probability of an ICF occurring and records the value, noting the initiating event. This value is then used to calculate the probability of failure for all kNCF sets for the initiating event . A mathematic representation of this process is shown in Equation 4 of Formulation 1.

The probability for every ICF failure and kNCF failure is recorded for the flow path. When all failure flow paths originating from an initiating event has been examined, the algorithm takes the sum of all critical failure probabilities for the initiating event. This allows the algorithm to account for situations leading to critical failure and determine a total probability of critical failure, as shown in Equation 5. The algorithm then selects the a new initiating event to analyze by determining the next function that can fail independently of other failure events; this process repeats until all initiating events are accounted for. A formulation for calculating the probability of total critical system failure for an initiating event is shown below in Formulation 1.

Formulation 1: Formulation of Critical System Failure for initiating failure event

Here we provide the mathematical formulation for **,** the probability of critical system failure for a path.

Formulation 1.1: Sets

Set of all types of functions

: Set of all function types in the functional model

: Set of independent critical functions

: Set of all function in critical functions set of type

: Set of all flow types in the functional model

: Set of all possible paths in the model, containing flows and denoting class. The first row contains all functions in order along the length of the path, and the second row contains all flows in order.

Set of classes of elements in a path, containing functions and flows

: Set of all unique paths originating at function and ending at an or

: A set of all paths originating from a function of interest for receiving directed failure flows

Formulation 1.2: Parameters

The probability that the function *f* will pass a failure along path

The probability that function *f* will accept a failure passed along path

Formulation 1.3: Calculation

If a flow path originating from a function contains no ICFs of kNCFs, the probability of critical system failure is given by.

If a flow path originating from a function contains an ICF, then the probability of critical system failure is given by

|  |  |
| --- | --- |
|  | (3) |

The probability of failure from kNCFs must be calculated for all paths containing kNCFs originating from function at once, and is given by,

|  |  |
| --- | --- |
|  | (4) |

The combined probability of failure for all paths originating from a function is given by

|  |  |
| --- | --- |
|  | (5) |

The value of is in terms of Probability of Survival on Demand (PSD), which is the probability that a system will survive a failure event.

Steps 5 and 6 currently present the best practice as determined by the authors at this time. However, alternative methods exist for implementation of these steps including those discussed in Step 5 above.

### Phase 3: Interpret and Report Results

#### Step 7: Analyze and Interpret Results

It is important to properly interpret the results of the failure flow analysis in order to choose the most effective Direct Failure functions to improve system survivability. From the analysis performed in Step 6, functions that are less critical to system survivability have the lowest probability of critical system failure . These functions are where failure should be directed; however, it is important to verify that the values make sense for the system. Due to the complexity of large functional models, errors could be made in either the initial functional model developed in Step 1, the selection of critical functions for the system in Step 2, or the assignment of failure probabilities in Step 3. Some potential errors to be mindful of are ICFs that do not have a value of 1, seemingly unimportant functions with very high probabilities (which can occur in many systems and may not be erroneous), and identical functions that have different probabilities of failure. Additionally, it is important to understand that in large complex systems, emergent behaviors are likely to surface, and determining why these behaviors arise is very important to improved survivability.

In addition to determining the probability of critical failure associated with a function, it is important to consider what classes of failure flows can be directed to a particular function. For example, an impact from a falling rock is difficult or impossible to direct towards an internal system component.

#### Step 8: Place FFDF Direct Failure Function

Using the insight gained in Step 7, Direct Failure functions can now be inserted into the functional model as desired. The Direct Failure function intercepts a failure flowing through the system and directs it towards the desired sacrificial subsystem of interest, as shown in Appendix 4. Similarly to the initial development of the functional model, it is important to consider the required model fidelity of the Direct Failure function, the interaction between the system and the environment, and the limitations of directing failure to certain functions (e.g., a pressure vessel cannot easily receive an electrical failure flow).

The FFDF method can be applied to a wide variety of systems to improve survival by informing failure flow routing decisions. Survival of systems is often highly dependent on the health of a few functions that are more closely linked to critical system functionality, and by using FFDF, sub-systems can be found that are sacrificial with minimal effect on system survival.

## 4 Case Study

In this section, a case study is performed in which a functional model of a Sojourner class MER (“Sojourner Rover Home Page” 2015) is modelled and analyzed using the FFDF method.

#### *Step 1: Develop Functional Model*

A functional model consisting of 50 functions and based on a Sojourner class MER is developed. A total of 14 types of functions are used in the model. Table 1 shows a list of the sub-systems represented in the functional model and representative functions. Appendix 1 shows the complete functional model.

The case study functional model is a simplified version of a Sojourner class Mars Rover and is complex enough to demonstrate FFDF while being simple enough to be easily understandable. The case study model is a more complex version of a MER model used in previous works (Short and Van Bossuyt 2015b; Short, Mimlitz, and Van Bossuyt 2016; Short and Van Bossuyt 2015a).

Table 1: Functions Used in Construction of the MER Functional Model

|  |  |  |
| --- | --- | --- |
| **Sub-System** | **Representative Function** | **Quantity Used** |
| Accumulate Energy | Solar Arrays | 13 |
| Store Energy | Batteries | 3 |
| Distribute Energy | Power Control Board | 1 |
| Direct Command | Primary CPU | 1 |
| Process Signal | Primary CPU | 1 |
| Store Data | Computer Memory | 3 |
| Record Position | Positional Sensors | 1 |
| Record Visual | Cameras | 3 |
| Process Signal | Communication Systems | 1 |
| Transmit Data | Communication Systems | 1 |
| Control Magnitude Electrical | Motor Control Board | 1 |
| Convert Electrical to Rotation | Driven Motors in Rocker Bogie Suspension | 10 |
| Transmit Rotation | Steering Columns on Wheels | 4 |
| Convert Rotation to Translation | Wheels | 6 |

The current 50-function model was analyzed in approximately 10 minutes on a standard laptop, but the scalability of the analysis was determined to be less dependent on the number of functions than on the number of possible paths through the model. For example, a long chain of functions in straight line would not significantly increase the power needed to solve the model, but the same number of functions added with flows interconnecting each function could significantly increase the power needed.

#### Step 2: Define Critical Functions and Flows

The functional model includes 7 critical functions (kNCFs) and 2 independent critical functions (ICFs). The kNCFs include the Store Energy functions representing batteries, Accumulate Energy functions representing the solar arrays, Record Visual functions representing cameras, Store Data functions representing on-board memory, and Convert Rotation to Translation functions representing wheels. These functions are selected to be kNCFs because they represent capabilities of the MER that are necessary to perform its mission but possess sufficient redundancies to allow for continued functionality with loss of some individual functions. The two ICFs are the Process Signal function, representing the CPU, and the Transmit Data function, representing the communication systems. These functions are chosen because if either of them is lost, the rover will cease to operate or communicate with human controllers on Earth.

#### Step 3: Assign Failure Probabilities

Failure probabilities for this case study are sourced from *Risk and Reliability in Space Mission Engineering, The New SMAD* (Wertz, Everett, and Puschell 2011b) and are representative but were intentionally modified to differ from true failure data. Further, we explicitly state that the failure probabilities and results of the analysis presented here, while being useful for demonstration of FFDF, are not intended for use in any real systems. Practitioners must develop their own system models and failure probability data for their specific applications and cannot rely upon the data presented here. A complete list of the failure probabilities used in this case study is shown in Appendix 2.

#### Step 4: Convert Functional Model into Mathematic Representation

The functional model is transformed from a graphical human-readable representation to a mathematical machine-readable representation using the format described in Step 4 in the Methodology section. Converting a graphical functional model to a mathematical functional model is currently achieved through a manual process and is labor intensive, but manual conversion still allows for the relatively rapid generation of functional models compared to previous work (Short and Van Bossuyt 2015a). In future work, an automated tool for conversion back and forth between graphical and mathematical representations of functional models will be developed.

#### Step 5: Define All Flow Paths for Machine Readability

All legal flow paths in the functional model are found using the functional model flow path solving algorithm described in Step 5 of the methodology above. This algorithm is computationally inexpensive and the software implementation used in this case study is able to execute rapidly on large, complex functional models. In the MER model used for this case study, 1371 paths were found and analyzed.

#### Step 6: Calculate System Critical Failure Probability from Initiating Events

Critical system failures are calculated using the method described in Step 6 of the methodology section. This is performed by first determining if a path contains an ICF or at least *k* kNCFs of the same set. If a flow path meets either of these conditions, the probability of a critical failure occurring is calculated using Equations 3 through 5. Using a relatively short path generated in Step 5, a sample calculation is performed, as shown in Eq. 8. In this case, the failure initiates at a Store Energy function representing a battery. Potential causes of battery failure could include overheating, a short across the battery, or mechanical failure due to physical impact. The Store Energy function failure propagates to the Distribute Electricity function representing a power control board, which accepts the failure, which is then passed to a Direct Command function representing a part of the primary computer. Finally, the failure is passed to the Process Signal function representing the computer’s processor and the rover experiences critical failure. Using Eq. 3 values shown in Appendix 2, the calculation is performed and shown as a result of Eq. 8.

|  |  |
| --- | --- |
|  | (8) |

This individual path has a low probability of occurrence, but when combined with the other 1371 paths, the probability of system failure can become significant. In the example shown in Eq. 8, only downstream failure flow paths were used; however, failure can propagate both downstream and upstream. Appendix 2 shows the associated probabilities for upstream and downstream failure propagation.

#### Step 7: Analyze and Interpret Results

After the functional model has been analyzed to determine which functions are likely to lead to critical system failure, the results are then checked and interpreted for design insights. For this case study, spot checks were performed in order to ensure that the analysis was properly carried out.

#### Step 8: Place FFDF Direct Failure Function

The Direct Failure functions are then placed in locations that most improve the probability of system survival when informing failure flow routing decisions. Additionally, it is important that the incoming failure flow be directed to a function that can accept the failure flow. For the case study presented in this paper, an electrical surge is considered and can be directed to systems that have electrical energy flows.

## 5 Results

The case study presented above demonstrates the capability of FFDF to improve the survivability of systems. Table 2 presents the Probability of Survival on Demand (PSD) for each function and the quantity of the function available. Some function types are listed multiple times, but with varying quantities and different PSD values, such as Convert Electrical to Rotational. In these cases, the same type of function appears in multiple contexts within the functional model.

For the case of an electrical surge in the system that must be routed into a sacrificial function, the optimal function to direct failure toward was found to be one of the two Convert Rotation-to-Translation functions representing the center unsteered wheels in the rocker-bogie suspension of the MER. Over the course of the mission, directing failure to one of the two Convert Rotation to Translation functions results in a PSD of 0.1520, or approximately 15%. If no failure is directed, it can be assumed that the failure is randomly assigned to a function uniformly; under this class of undirected failure, the PSD is 0.1151. The worst-case scenario is directing the failure to one of the independent critical functions (ICFs), such as the Process Signal function, resulting in a PSD of 0. It is therefore recommended that a Direct Failure function be placed to intercept the failure flow representing an electrical surge and direct it to one of the previously described Convert Rotation to Translation functions. This results in an approximately 4% increase in system survivability over the length of the mission.

Another function towards which it is advantageous to direct failure is one of the thirteen Collect Energy Solar functions each of which have a PSD of 0.1498. While these functions have a slightly lower probability of system survival when sacrificed, there are significantly more available in the system signifying that several can be sacrificed over the length of the mission if necessary. One drawback of this is that the Collect Energy Solar functions are kNCFs, so if too many fail, the MER will no longer be able to function. This is why the Collect Energy Solar function’s PSD value is as low as it is despite the high level of system redundancy.

Table 2: Results of FFDF Analysis through FFIP shown in Probability of Survival on Demand (PSD), with the highest PSD sacrificial function bolded.



The Convert Rotation to Translation functions representing the steered wheels in the rocker-bogie suspension has a PSD of 0.0990, which is almost half the PSD value of the unsteered wheels. This appears to be a result of the fact that there are two flows connecting the steered wheels to the rest of the system versus the singular flow for the unsteered wheels.

Another observable behavior is that the more isolated a function is (in terms of both number of flows connecting it to its neighbors and proximity to critical functions), the less likely it is to lead to critical system failure. This holds logically, because if failure is propagating along paths, failure flows that must take fewer and longer paths to propagate to a critical function will be less likely to cause critical system failure. Related to this behavior is the seemingly counter-intuitive behavior that a function with lots of parallel flows leading to it is more likely to fail. Analysts might surmise that failure should be less likely to occur due to the apparent redundancy; however, the opposite is true because the function is a bottleneck in the system with a large number of potential paths for accepting failure, but with no redundancy of its own. Additionally, if this system were to fail, the failure could propagate along many more flows, leading to higher probability of critical function failure.

As a result of the heightened probability of system survivability from directed failure to the Convert Rotation to Translation and Accumulate Energy Solar functions, it is recommended the Direct Failure functions be placed into the system in order to direct failure to these functions. Appendix 4 shows a graphical representation of this. As the mission length increases, survival probability diminishes, but the disparity between directed and undirected failure increases due to their different initial hazard rates.

## 6 Discussion

In the case study, FFDF is shown to be effective for improving the survival of a Sojourner class Mars Exploration Rover (MER) by directing a failure flow representative of an electrical surge into a Convert Rotation-to-Translation function representing an unsteered wheel. As stated in the Results section above, the probability of survival can be increased from PSD 0.11 to 0.15 by directing failure to one of the two Convert Rotation-to-Translation functions or to one of the thirteen Collect Energy Solar functions. While the Convert Rotation-to-Translation function has a marginally better survivability at PSD 0.1520 compared to the 0.1498 of the Collect Energy Solar function, it may be advantageous to direct the failure to the Collect Energy Solar functions because there is more redundancy. While it will result in a slightly higher probability of failure, 0.22%, there are more Collect Energy functions that can be sacrificed before critical system failure occurs. One potential avenue of future work is the analysis of FFDF where multiple failures occur over time and multiple Direct Failure functions are inserted in an effort to determine how the loss of a function and related functions can affect the loss of another function and related functions later in the mission. Further discussion of this can be found in Section 7.1.

The FFDF method can have positive impacts in many system design efforts in which system failure or downtime is highly undesirable, including space exploration, transportation, and power generation. For example, a power generation system design approached with FFDF will be more reliable and have higher system up-time, which is important for power generation contracts and grid load balancing. FFDF can also allow for rapid automated design and analysis of an entire power grid, including functions for power generation, distribution, and regulation with a focus on continuous and uninterrupted power.

## 7 Conclusion and Future Work

The FFDF method is effective for improving system survivability at the conceptual phase of system design. FFDF presents a method that analyzes a conceptual system design to determine the most desirable locations at which to place Direct Failure functions that route failure flows to sacrificial subsystems or functions and away from critical functions. FFDF is performed through the completion of three phases. The first phase encompasses the generation of a functional model to represent a system of interest. This phase consists of actions that build on existing functional modelling techniques, and proposes a convention for the development of machine interpretable functional models for analysis. The second phase consists of analysis of the functional model using FFIP and similar methods. The final phase consists of interpreting the results of the analysis to determine where it is best to deliver system failure. The end result of FFDF is deeper insight into how system failure can be prevented through protecting critical system functions in the early conceptual phase of system design.

### 7.1 Future work

Future development of FFDF will involve improvement of the analysis software for efficiency and usability. Specifically, the development of a user-friendly Graphical User Interface (GUI) for the creation of functional models. In addition to making FFDF more approachable to new users, the GUI will allow for the generation of a wide variety of functional models through crowd sourcing. The shared models may then provide a basis for the development and analysis of increasingly complex systems through FFDF and other related methods. Additionally, potential problems with scalability of highly interconnected functional models could be addressed through a pre-solving technique that identifies sections that can be simplified and reducing the effective size and complexity of the functional model prior to analysis.

One interesting potential application of FFDF is the design and analysis of individuals in a swarm of robots designed to explore an unknown space (Truszkowski et al. 2004; Navarro et al. 2012). For this mission, the swarm could be modelled as a single system-of-systems. The system-of-systems model could then be analyzed to determine how control and management decisions of the swarm affect swarm health and capability. In this case, a relatively small increase in individual survivability can lead to greater survivability of the swarm in its attempt to complete a mission.

Finally, the implementation of user friendly and streamlined methods for inclusion of FFDF into risk analysis techniques for decisions making such as Active Mission Success Estimation is currently being pursued (Short and Van Bossuyt 2016). This will enable functional models generated and analyzed through FFDF to be applied to the analysis of large and complex mission scenarios such as space mission design or as a basis for development of autonomous system behavior when facing risk.

## Appendix 1



## Appendix 2

|  |  |  |
| --- | --- | --- |
| **Flow Type** | **Probability of Passing Failure Downstream** | **Probability of Passing Failure Upstream** |
| Collectable Energy | 0.10 | 0.00 |
| Electrical Energy | 0.40 | 0.02 |
| Digital Signal | 0.50 | 0.02 |
| Control Signal | 0.50 | 0.02 |
| Positional Information | 0.47 | 0.00 |
| Visual Information | 0.47 | 0.00 |
| Rotational Work | 0.50 | 0.15 |
| Translational Work | 0.50 | 0.15 |
| Alignment Work | 0.25 | 0.15 |
|  |  |  |
| **Function** | **Probability of Accepting Failure Flow** |  |
| Accumulate Energy | 0.50 |  |
| Store Energy | 0.12 |  |
| Distribute Electrical | 0.24 |  |
| Control Magnitude Electrical | 0.20 |  |
| Convert Electrical to Rotation | 0.16 |  |
| Transmit Rotation | 0.16 |  |
| Convert Rotation to Translation | 0.16 |  |
| Direct Command | 0.44 |  |
| Process Signal | 0.01 |  |
| Store Data | 0.01 |  |
| Record Position | 0.22 |  |
| Record Visual | 0.22 |  |
| Transmit Data | 0.44 |  |

## Appendix 3

|  |  |  |  |
| --- | --- | --- | --- |
| **Index** | **Flow Type** |  |  |
| *f1* | Collectable Energy |  |  |
| *f2* | Electrical Energy |  |  |
| *f3* | Digital Signal |  |  |
| *f4* | Position Information |  |  |
| *f5* | Visual Information |  |  |
| *f6* | Rotational Work |  |  |
| *f7* | Translation Work |  |  |
| *f8* | Steering Work |  |  |
|  |  |  |  |
|  |  |  |  |
| **Index** | **Function Type** | **Index** | **Function Type** |
| 1 | Operating Environment | 26 | Convert Electric-to-Rotation 7 |
| 2 | Accumulate Energy 1 | 27 | Convert Electric-to-Rotation 8 |
| 3 | Accumulate Energy 2 | 28 | Convert Electric-to-Rotation 9 |
| 4 | Accumulate Energy 3 | 29 | Convert Electric-to-Rotation 10 |
| 5 | Accumulate Energy 4 | 30 | Transmit Rotation 1 |
| 6 | Accumulate Energy 5 | 31 | Transmit Rotation 2 |
| 7 | Accumulate Energy 6 | 32 | Transmit Rotation 3 |
| 8 | Accumulate Energy 7 | 33 | Transmit Rotation 4 |
| 9 | Accumulate Energy 8 | 34 | Convert Rotation-to-Translation 1 |
| 10 | Accumulate Energy 9 | 35 | Convert Rotation-to-Translation 2 |
| 11 | Accumulate Energy 10 | 36 | Convert Rotation-to-Translation 3 |
| 12 | Accumulate Energy 11 | 37 | Convert Rotation-to-Translation 4 |
| 13 | Accumulate Energy 12 | 38 | Convert Rotation-to-Translation 5 |
| 14 | Accumulate Energy 13 | 39 | Convert Rotation-to-Translation 6 |
| 15 | Store Energy 1 | 40 | Direct Command |
| 16 | Store Energy 2 | 41 | Process Signal |
| 17 | Store Energy 3 | 42 | Process Signal (digital) |
| 18 | Distribute Electricity | 43 | Store Data 1 |
| 19 | Control Magnitude Electrical | 44 | Store Data 2 |
| 20 | Convert Electric-to-Rotation 1 | 45 | Store Data 3 |
| 21 | Convert Electric-to-Rotation 2 | 46 | Record Position |
| 22 | Convert Electric-to-Rotation 3 | 47 | Record Visual 1 |
| 23 | Convert Electric-to-Rotation 4 | 48 | Record Visual 2 |
| 24 | Convert Electric-to-Rotation 5 | 49 | Record Visual 3 |
| 25 | Convert Electric-to-Rotation 6 | 50 | Transmit Data (analogue) |

## Appendix 4



## References

Blanchard, Benjamin S., and Walter J. Fabrycky. 1990. *Systems Engineering and Analysis*. Vol. 4. Prentice Hall Englewood Cliffs, New Jersey. http://sutlib2.sut.ac.th/sut\_contents/H104135.pdf.

Bohm, Matt R., Robert B. Stone, and Simon Szykman. 2005. “Enhancing Virtual Product Representations for Advanced Design Repository Systems.” *Journal of Computing and Information Science in Engineering* 5 (4): 360–72.

Browning, Tyson R. 2001. “Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions.” *IEEE TRANSACTIONS ON ENGINEERING MANAGEMENT* 48 (3). http://axiod.com/technology/papers/4DSMs.pdf.

David, Pierre, Vincent Idasiak, and Frederic Kratz. 2010. “Reliability Study of Complex Physical Systems Using SysML.” *Reliability Engineering & System Safety* 95 (4): 431–50.

Distefano, Salvatore, and Antonio Puliafito. 2007. “Dynamic Reliability Block Diagrams: Overview of a Methodology.” In *ESREL*, 7:1059–68. https://www.researchgate.net/profile/Antonio\_Puliafito/publication/254229063\_Dynamic\_reliability\_block\_diagrams\_Overview\_of\_a\_methodology/links/0f31753c575b50cf6d000000.pdf.

Eck, Dingmar van, Daniel A. McAdams, and Pieter E. Vermaas. 2007. “Functional Decomposition in Engineering: A Survey.” In *ASME 2007 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 227–36. American Society of Mechanical Engineers. http://proceedings.asmedigitalcollection.asme.org/proceeding.aspx?articleid=1604183.

Ericson, C. 1999. *Fault Tree Analysis–A History from the Proceeding of the 17th International System Safety Conference*. Orlando.

Force, US Air. 1981. “ICAM Architecture Part II, Vol. IV., Function Modelling Manual (IDEF0).” AFWAL-TR-81-4023, OH: Wright-Patterson Air Force Base, USA.

Garvey, Paul R., and C. Ariel Pinto. 2009. “Introduction to Functional Dependency Network Analysis.” In *The MITRE Corporation and Old Dominion, Second International Symposium on Engineering Systems, MIT, Cambridge, Massachusetts*. https://pdfs.semanticscholar.org/865c/27f6870ead4fddc7ab0af3248f89f1875dc7.pdf.

Garvey, Paul R., C. Ariel Pinto, and Joost Reyes Santos. 2014. “Modelling and Measuring the Operability of Interdependent Systems and Systems of Systems: Advances in Methods and Applications.” *International Journal of System of Systems Engineering* 5 (1): 1–24.

Gosselin, S. R. 2006. *Probabilities of Failure and Uncertainty Estimate Information for Passive Components: A Literature Review*. Division of Fuel, Engineering, and Radiological Research, Office of Nuclear Regulatory Research, US Nuclear Regulatory Commission.

Guariniello, Cesare, and Daniel DeLaurentis. 2017. “Supporting Design via the System Operational Dependency Analysis Methodology.” *Research in Engineering Design* 28 (1): 53–69.

Haimes, Yacov Y., Barry M. Horowitz, James H. Lambert, Joost R. Santos, Chenyang Lian, and Kenneth G. Crowther. 2005. “Inoperability Input-Output Model for Interdependent Infrastructure Sectors. I: Theory and Methodology.” *Journal of Infrastructure Systems* 11 (2): 67–79.

Hirtz, Julie, Robert B. Stone, Daniel A. McAdams, Simon Szykman, and Kristin L. Wood. 2002. “A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts.” *Research in Engineering Design* 13 (2): 65–82.

Huang, Edward, Randeep Ramamurthy, and Leon F. McGinnis. 2007. “System and Simulation Modeling Using SysML.” In *Proceedings of the 39th Conference on Winter Simulation: 40 Years! The Best Is yet to Come*, 796–803. IEEE Press. http://dl.acm.org/citation.cfm?id=1351687.

Hutcheson, Ryan S., Daniel A. McAdams, Robert B. Stone, and Irem Y. Tumer. 2006. “A Function-Based Methodology for Analyzing Critical Events.” In *ASME 2006 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 1193–1204. American Society of Mechanical Engineers. http://proceedings.asmedigitalcollection.asme.org/proceeding.aspx?articleid=1588322.

Jensen, David, Irem Y. Tumer, and Tolga Kurtoglu. 2009. “Flow State Logic (FSL) for Analysis of Failure Propagation in Early Design.” In *ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 1033–43. American Society of Mechanical Engineers. http://proceedings.asmedigitalcollection.asme.org/proceeding.aspx?articleid=1650614.

“JPL Team X.” 2016. Accessed April 1. http://jplteamx.jpl.nasa.gov/.

Kalvin, Alan D, and Yaakov L Varol. 1983. “On the Generation of All Topological Sortings.” *Journal of Algorithms* 4 (2): 150–62. doi:10.1016/0196-6774(83)90042-1.

Kumamoto, Hiromitsu, Henley, and Ernest J. 1996. *Probabilistic Risk Assessment and Management for Engineers and Scientists*. Institute of Electrical & Electronics Engineers (IEEE Press). http://www.bcin.ca/Interface/openbcin.cgi?submit=submit&Chinkey=231931.

Kurtoglu, Tolga, and Irem Y. Tumer. 2007. “Ffip: A Framework for Early Assessment of Functional Failures in Complex Systems.” In *The International Conference on Engineering Design, ICED*. Vol. 7. http://www.designsociety.org/download-publication/25348/ffip\_a\_framework\_for\_early\_assessment\_of\_functional\_failures\_in\_complex\_systems.

———. 2008. “A Graph-Based Fault Identification and Propagation Framework for Functional Design of Complex Systems.” *Journal of Mechanical Design* 130 (5): 051401.

Kurtoglu, Tolga, Irem Y. Tumer, and David C. Jensen. 2010. “A Functional Failure Reasoning Methodology for Evaluation of Conceptual System Architectures.” *Research in Engineering Design* 21 (4): 209–34.

Lightsey, Bob. 2001. “Systems Engineering Fundamentals.” DTIC Document. http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA387507.

Long, Jim. 2002. “Relationships between Common Graphical Representations in Systems Engineering.” *Vitech White Paper, Vitech Corporation, Vienna, VA*, 70.

Lucero, Briana, Vimal K. Viswanathan, Julie S. Linsey, and Cameron J. Turner. 2014. “Identifying Critical Functions for Use across Engineering Design Domains.” *Journal of Mechanical Design* 136 (12): 121101.

Materese, Robin. 2002. “A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts.” Text. *NIST*. February 1. https://www.nist.gov/node/742436.

Mimlitz, Short, and Van Bossuyt. 2016. “Towards Risk-Informed Operation of Autonomous Vehicles to Increase Resilience in Unknown and Dangerous Environments.” In *ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*.

Mohr, R. R. 2002. “Failure Modes and Effects Analysis.” *JE Jacobs Sverdrup,*. http://icecube.wisc.edu/~kitamura/NK/Flasher\_Board/Useful/FMEA.pdf.

Navarro, I&#xf1, aki, Mat&#xed, Fernando A, I&#xf1 Navarro, aki, Mat&#xed, and Fernando A. 2012. “An Introduction to Swarm Robotics, An Introduction to Swarm Robotics.” *International Scholarly Research Notices, International Scholarly Research Notices* 2013, 2013 (September): e608164. doi:10.5402/2013/608164, 10.5402/2013/608164.

O’Halloran, Bryan M., Nikolaos Papakonstantinou, and Douglas L. Van Bossuyt. 2015. “Modeling of Function Failure Propagation across Uncoupled Systems.” In *Reliability and Maintainability Symposium (RAMS), 2015 Annual*, 1–6. IEEE. http://ieeexplore.ieee.org/xpls/abs\_all.jsp?arnumber=7105107.

Papakonstantinou, Nikolaos, Seppo Sierla, David C. Jensen, and Irem Y. Tumer. 2012. “Simulation of Interactions and Emergent Failure Behavior during Complex System Design.” *Journal of Computing and Information Science in Engineering* 12 (3): 031007.

Rumbaugh, James, Ivar Jacobson, and Grady Booch. 2004. *Unified Modeling Language Reference Manual, The*. Pearson Higher Education. http://dl.acm.org/citation.cfm?id=993859.

Sen, Chiradeep, Joshua D. Summers, and Gregory M. Mocko. 2013. “Physics-Based Reasoning in Conceptual Design Using a Formal Representation of Function Structure Graphs.” *Journal of Computing and Information Science in Engineering* 13 (1): 011008.

Short, Adam R., and Douglas L. Van Bossuyt. 2015a. “Rerouting Failure Flows Using Logic Blocks in Functional Models for Improved System Robustness: Failure Flow Decision Functions.” In *International Conference on Engineering Design 2015*.

Short, Mimlitz, and Van Bossuyt. 2016. “Autonomous System Design and Controls Design for Operations in High Risk Environments.” In *ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*.

Short, and Douglas L. Van Bossuyt. 2015b. “Risk Attitude Informed Route Planning in a Simulated Planetary Rover.” In *ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, V01BT02A048–V01BT02A048. American Society of Mechanical Engineers. http://proceedings.asmedigitalcollection.asme.org/proceeding.aspx?articleid=2483254.

———. 2016. “Active Mission Success Estimation through PHM-Informed Probabilistic Modelling.” Accessed March 4. https://www.phmsociety.org/sites/phmsociety.org/files/phm\_submission/2015/phmc\_15\_051.pdf.

“Sojourner Rover Home Page.” 2015. Accessed December 15. http://mars.nasa.gov/MPF/rover/sojourner.html.

Stone, Robert B., Irem Y. Tumer, and Michael Van Wie. 2005. “The Function-Failure Design Method.” *Journal of Mechanical Design* 127 (3): 397–407.

Stone, Robert B., and Kristin L. Wood. 2000. “Development of a Functional Basis for Design.” *Journal of Mechanical Design* 122 (4): 359–70.

Truszkowski, Walt, Mike Hinchey, James Rash, and Christopher Rouff. 2004. “NASA’s Swarm Missions: The Challenge of Building Autonomous Software.” *IT Professional* 6 (5): 47–52.

Wertz, James Richard, David F. Everett, and Jeffery John Puschell. 2011a. “Risk and Reliability.” In *Space Mission Engineering: The New SMAD*. Microcosm Press.

———. 2011b. *Space Mission Engineering: The New SMAD*. Microcosm Press.

Yadav, Sandeep, Kamal Kumar Verma, and Swetamadhab Mahanta. 2012. “The Maze Problem Solved by Micro Mouse.” *International Journal of Engineering and Advanced Technology (IJEAT) ISSN*, 2249–8958.

1. A version of this paper appeared in the ICED2015 conference where it was recognized as a Reviewers’ Favourite. [↑](#footnote-ref-1)
2. Department of Mechanical Engineering, Colorado School of Mines, Golden, Colorado, United States of America [↑](#footnote-ref-2)