DESIGN ANALYSIS USING FUNCTION-BASED FAILURE PROPAGATION IN FAILED SYSTEM STATES

AN ABSTRACT OF THE THESIS OF

David Charles Jensen for the degree of Master of Science in Mechanical Engineering presented on June 1, 2009.

Title: Design Analysis Using Function-Based Failure Propagation in Failed System States.

Abstract approved:

_____

Irem Y. Tumer

For safety critical complex systems, reliability and risk analysis are important design steps. Implementing these analyses early in the design stage can reduce costs associated with redesign and provide important information on design viability. In the past several years, various research methods have been presented in the design community to move reliability analysis into the early conceptual design stages. Concept stage reliability analysis faces several challenges. First, in complex systems, the integration of software and hardware into a single reliability framework is hampered by the different risk factors and behaviors of these two types of subsystems. Secondly, quantifying risk in order to make design decision is challenging when minimal technical information of the system and components is known. Finally, the current methods of conceptual design reliability analysis use a functional representation as the basis for reliability analysis. However, in non-nominal system states, the functional representation can have serious limitations. This limitation is evident when failures are modeled to propagate along energy, material, and signal (EMS) flows. A function-based reliability method must consider all potential flows, and not be limited to the function structure of the nominal state. This work introduces the Flow State Logic (FSL) method as a means for reasoning on the state of EMS flows. This method provides a means to assess

failure propagation over potential flows that were not considered in a functional representation of a “nominally functioning” design. The impact with respect to the operating state of functional elements can be calculated for these possible flow failures as well as other failure events. The results from this type of analysis can then be used to make design decisions for integrated software-hardware, complex systems.

Design Analysis Using Function-Based Failure Propagation in Failed System States

by
David Charles Jensen

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented June 1, 2009
Commencement June 2009

Master of Science thesis of David Charles Jensen presented on June 1, 2009

APPROVED:

Major Professor, representing Mechanical Engineering

Head of the School of Mechanical, Industrial and Manufacturing Engineering

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

David Charles Jensen, Author

ACKNOWLEDGEMENTS

I am grateful to many people for their assistance in refining this research and assisting me in producing this work. First, the direct guidance and detailed editing from Dr. Irem Y. Tumer has been immensely helpful. Her dedication to both my research and me personally has made all the difference. Considerable thanks is also due to Dr. Tolga Kurtoglu for the direction of the beginning of this work, his guidance for the Summer of 2008, and the vision he provides for its future. I am also grateful to the many individuals at NASA Ames who answered my innumerable questions. My peers in the Complex Systems Design Lab have provided both encouragement and technical criticism in equal measure.

The foundation for all of this has been laid by the sacrifices my parents and my Grandma Rose Marie have made to provide me with my undergraduate education. Thank you for letting me tear apart every broken household gadget. Graduate school would not have been possible for me if not for the faithful care of my beautiful wife, Heidi. Thank you for being willing to follow me no matter what!

CONTRIBUTION OF OTHER AUTHORS

Much of the work presented in the thesis has been published or has been submitted to publishing. This author has worked closely with Dr. Tolga Kurtoglu and Dr. Irem Y. Tumer to produce the following three works:

1. Jensen, D., Tumer, I.Y., Kurtoglu, T. *Modeling the Propagation of Failures in Software Driven Hardware Systems to Enable Risk-Informed Design*. in *ASME International Mechanical Engineering Congress and Exposition*. 2008. Boston, Massachusetts.

2. Jensen, D., Tumer, I.Y., and T. Kurtoglu. *Flow State Logic (FSL) for analysis of failure propagation in early design*. in *International Design Theory and Methodology Conference, IDETC/CIE*. 2009. San Diego, CA.

3. Kurtoglu, T., Tumer, I.Y., Jensen, D. *A Function Failure Reasoning Method for Evaluation of Conceptual System Architectures.* Submitted to J. of Research in Engineering Design. April, 2009.

In Chapter 4 the methodology outlined in sections 4.1 through 4.4 was created by Dr. Kurtoglu and Dr. Tumer. It is presented as part of this thesis to provide context to the application of the method to the electrical power system in section 4.5 and 4.6. These later sections represents a significant contribution by this author.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

Page

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF TABLES

LIST OF APPENDIX FIGURES

LIST OF APPENDIX TABLES

DEDICATION

Soli Deo Gloria.

Design Analysis Using Function-Base Failure Propagation in Failed System States

1 Introduction

The fundamental complexity of a complex system stems from the integration of multiple technical disciplines to form one successful design. A systems engineering approach to design requires information management of inputs from these different contributing disciplines. For example, the electrical, mechanical and control software subsystems of a satellite must work together to extend a solar energy panel. In the design stage of this satellite technical information must be shared between subsystem experts to create a successful design. Further, consider an anti-lock brake system. The hydraulic, electrical control, and sensor suite all contribute to the successful operation of the system. The failure of one subsystem directly affects other connected subsystems and the performance of the system as a whole. For safety critical systems it is therefore necessary to view reliability from at a system-wide perspective.

Numerous products currently being designed or manufactured incorporate hardware and software components. For safety critical systems simple reliability analysis approaches that ignore the complex interactions of these components are insufficient. What is needed instead is a single model that can incorporate software and hardware as well as their interactions. The basic building blocks are clearly different for software and hardware, indicating that a combination of lines of code and nuts and bolts will not produce a viable model. The fundamental differences between hardware and software and their associated failures often lead to the use of different reliability methods and metrics for these two components of complex systems. An integrated approach to reliability would thus have to incorporate software and hardware in a similar way in order to accurately represent the reliability of the total system.

There are other significant advantages to conducting reliability analysis during product design, including decreased redesign time, improved quality and safety, and decreased manufacturing time [2]. Most reliability analysis techniques, however, require mature designs to provide meaningful information. Several recent research efforts have

focused on moving risk and reliability into the conceptual design phase. Different approaches have attempted to do this in numerous ways. From research focusing on automated design, reasoners have been developed to assess system reliability as a metric for design decision-making. The use of autonomous control for aircraft and other applications has led to a need for fault detection and mitigation software. Some research has focused on providing fault information to software engineers as early in the conceptual design as possible to facilitate the creation of these health management systems. Finally, design theory research has focused on improving product quality by addressing component and system reliability using statistical and qualitative measures. Fundamental to each of these different methods for reliability analysis is the means of system representation.

Conceptual reliability analysis has made extensive use of functional modeling for system representation. A functional model is a block diagram composed of function blocks identified as acting on energy, material and signal (EMS) flows, which are arcs between the blocks. Functional models of systems represent how the design is expected to transform EMS flows to satisfy the objective or requirements for that system. Some methods of conceptual reliability analysis consider only single fault, independent failures. This limited set of failure types is insufficient for complex systems analysis where failures can be expected to propagate. For failure propagation analysis, failures are generally modeled to follow EMS paths. For example, a software fault in the controlled release of a solar power array from a satellite might propagate along the path of the control signal to cause a failure in a mechanical component. Further, a failure in the speed sensor from the anti-lock braking system would propagate to the software controller along a signal path and then to the brake calipers along the hydraulic liquid path. The previous two examples use component names instead of functions however the underlying logic is the same. Because functions (and components) interact by means of EMS flows, failures must propagate along these flow paths.

This research has identified the designed system representation as being insufficient for failure propagation analysis for numerous types of failure scenarios. In conceptual reliability analysis the system representation is based on the designed or nominal mode. In many possible failure scenarios there are EMS flows that are not part of the designed system. Explosions, impacts, and backflow all represent possible failures with EMS flows that would not be accounted for in the nominal system representation. The motivation of this work was to address how a reliability method might analyze failures that propagate along pathways that are not accounted for in the nominal system representation.

The contribution of this work to the field of conceptual design reliability analysis is a presentation of a general approach to failure analysis leading to a specific methodology developed to augment previous research. The novel approach to failure propagation presented here is to categorize failures paths based on the state of EMS flows in a system. In this work, EMS (and data) flow states are categorized for use in failure propagation reasoning. The application of this concept is the following Flow State Logic (FSL) method. This method is an augmentation of a previous research in function-based failure reasoning method, specifically, the Function Failure Identification and Propagation (FFIP) framework. The contribution of this research can clearly be seen through a comparison of results and scope of analysis of the original FFIP and the FSL augmented method.

This research is based on the rich field of fault detection and mitigation and in early design reliability analysis. Chapter 2 provides a background and state of the art related to this area as well as identifies the specific method chosen for this research, namely the Function Failure Identification and Propagation (FFIP) framework. In Chapter 3 it is shown how the FFIP method is capable of addressing the issue of integrating software and hardware systems and presents a means of adapting that method to evaluate software system monitoring. Chapter 4 presents an extension of FFIP for quantifying risk in design for conceptual design decision-making. This method is illustrated with a by an analytic analysis of redundancy for a set of

conceptual electrical power system designs. In Chapter 5 the FSL method and the concept of failure paths following designed and potentials EMS flows is presented. Chapter 6 provides a proof of concept case study where both the FFIP method and the FSL augmented methods are applied to the conceptual design of a liquid-fueled rocket engine. While the result of applying both methods to the case study can be found in Chapter 6, Chapter 7 presents a detailed discussion of the results, their implications and resulting future work. This work is concluded in Chapter 8 by overview the goal, result and impact of this research.

2 Current State and Supporting Work

2.1 Reliability for Existing Designs

2.1.1 Fault Mitigation in Operation

The first step in operational fault mitigation is fault detection. Detection is largely done though system monitoring using sensors and a reasoning agent. The creation of these reasoning agents is founded upon the research in model-based diagnosis (MBD). MBD makes use of early work in qualitative physics and qualitative reasoning [3-6]. MBD shares a common process in which a system is monitored and a comparison is performed of observed and expected behavior of the system to detect anomalous conditions usually with the goal of run-time repair [7]. The artificial intelligence community for MBD employed system configuration and qualitative behavior models for diagnosis tasks [8, 9]. Livingstone [10] and its extension L2 [11] are two of the most notable examples from NASA that utilize algorithms adapted from qualitative model-based diagnosis. In control engineering communities, transfer functions and state space equations are used as system models [12]. Finally, expert systems are extensively used in diagnosis, where knowledge acquired from human experts is formulated in different ways such as “if-then” rules or decision trees [13], and statistical and probabilistic classification methods are applied where physical behavioral is difficult to model in analytical form [14, 15].

2.1.2 Risk Analysis

The area of complex software driven system risk analysis can be seen as the convergence of the two well-established fields of software and hardware reliability. This generally leads to analyzing the reliability of software and hardware components separately then combining and modeling the interactions between the two different systems. The fundamental differences between these two domains of engineered systems and their associated expertise provide reason for this separate analysis. Such

differences include the type and occurrence of failures experienced by each system and can be easily illustrated in a stochastic approach. Where hardware components may experience independent failures, software failures are not independent [16]. Secondly, the probabilities of failure are markedly different. As a hardware system ages, physical degradation leads to predictable and repeatable failures in contrast to software, which tends to become more reliable as the random failures are removed through testing and updating while in operation. In combined software/hardware systems there are additional sources of failure from the interaction of these two systems. Hardware can operate outside of software design leading to failure and alternatively software can operate outside the feasible physical reality of hardware components [17].

One exception to the approach to dealing with hardware systems and software systems separately is found in the literature of computer and communications engineering[18, 19]. Computer hardware components tend to be operated at their technological maximum operating capacities increasing the awareness of hardware faults on software systems and their subsequent inclusion into reliability analysis. For this latter reasoning there is a rich breadth of published studies on reliability of software/hardware interactions analyzing computer systems [18-21]. However, this area is limited to failure of hardware that the software is operating over and does not adequately reflect the failures of mechanical hardware found outside of the computer.

The prediction of failures in a software system is generally approached with three different analysis methods: error seeding and tagging, data/input domain analysis, and time domain analysis. Error seeding is a testing stage approach where software faults and data errors are injected into a system. The resulting system performance is measured and errors are tracked. This data can be used to quantify the expected reliability of the system in operation to a reasonable degree of certainty. However, error seeding is a time intensive process and requires a fully written code to be applicable [22]. The more common approach is a time domain reliability model.

Using this method, many different software reliability growth models (SRGMs) have been developed and have been shown to be useful for different applications. These are in essence probabilistic models used to predict failure over the useful life of the software. Examples such as Wiebull and Gamma failure time class models, infinite failure category models and several others can be found in [23]. The previous reliability methods require some form of actual software code to evaluate. A design stage approach is considered in [24] which describes a method of analyzing reliability based on software failure modes and the probability of those failures.

Numerous methods exist for system level risk and reliability analysis of mature hardware designs. These methods use specific component and configuration information to derive the effect and likelihood of failures to provide designers with risk information. In particular, the three main reliability methods currently accepted in practice are FMEA, FTA and PRA. Failure modes and effects analysis (FMEA) is a tool for analyzing risks of a system down to a desired component level fidelity. By linking the likely failure modes and the resulting system effect for each component, FMEA offers a designer a means to evaluate the overall reliability of the system [25]. Using an FMEA approach for risk analysis involves determining component or subsystem level failures and evaluating what effect those failures would have on the rest of the system using expert judgment. By incorporating fault likelihood and severity an overall risk profile can be created based on the whole-system evaluation. Fault tree analysis (FTA) is an incident focused method that starts with an undesirable system event and then works backwards to define the contributing events that would lead to that higher-level event. FTA uses Boolean logic descriptors to combine all the possible events that may lead to a failed state and is represented in a tree structure [26]. Probabilistic risk assessment (PRA) combines event sequencing diagrams and fault trees to develop a stochastic model of the overall system reliability [27, 28]. PRA method defines risk mathematically as multiplicative result of the probability of a specific failure occurring and the cost of that failure. Event trees are used as analysis tools for identifying critical events and the possible effect of responsive measures.

Based on these responses the impact of various paths for failure propagation can be estimated. Fault trees identify all the necessary steps to cause a specified failure for a system. Because of this level of component (either physical or functional) analysis fault trees are generally suited to design analysis while event trees are more suited to procedure analysis.

2.1.3 Quality Approach for Risk Reduction

Several other methods for implementing quality into design have been presented as a mean for improving system reliability. For example Suh [29] presented functional independence and information content of a system as axioms that serve as goals for the highest quality design. While Pahl and Beitz [30] used a general approach to design by starting with system and subsystem functions leading ultimately to the refinement of a specific embodiment for that system. This is the general approach used in this paper. Alternatively quality in design has been addressed by limiting inherent variance and uncertainty as in the Kececioglu [31] and Taguchi [32] methods. More recent research has focused on “robust design”, or designs that have minimal loss of functionality various operating conditions [33]. The robust design approach has been used to address design decision making as related to noise factors [34].

2.2 Function-Based Reliability Methods in Conceptual Design

The majority of design stage reliability methods use functional modeling as a basis for system representation and analysis. Functional modeling is a system representation method that was developed as a means to enhance the concept generation stage of product design [30]. This method identifies specific functions that a product must accomplish and connects these functions in a block diagram with the EMS flows that are then transformed by the functions. Functional models are created by decomposing high-level functions, which are system or customer requirements, into sub functions. Functions structures are composed of functions and flows as verb-noun pairs and can be dissected to a fidelity level where components can be identified to embody functions. Additionally, functional modeling had been used in reverse engineering

methods such as the Force Flow Analysis [35, 36]. This method dissects products into components and the forces acting on and between components and finally into the functions that components embody. The Functional Basis was developed by Hirtz et al [37, 38] in order to avoid the use of designer specific function and flow descriptions, providing a standard taxonomy concept design. Because of the usefulness of functional modeling for system representation in early design, it has also been used as part of the system representation in some early design reliability methods.

2.3 Function-Based Failure Analysis

The first method that proposed a function-based approach to failure analysis was the Function Failure Design Method (FFDM) [39, 40]. In FFDM, the functional model is developed to represent the system design, which serves as a basis for generating configuration concepts of component implementations of function. Based on historical failure data for these types of components, it is then possible to establish likely failure modes for a given function. While useful, when using historical failure data for component information, configuration-specific failures are difficult to incorporate into the analysis, leading to failures that are limited to single independent faults.

Several other methods built upon the FFDM methodology. Hutcheson et al. presented a methodology to enable the design of health monitoring modules concurrently with system conceptual design in order to reveal, model, and eliminate associated risks and failures [41]. Additionally, Grantham et al. presented the Risk in Early Design (RED) method of formulating functional-failure likelihood and consequence based risk assessment classified high-risk to low-risk function-failure combinations [42, 43].

Connecting component failure and risk to a functional model for design stage reliability was also shown by Meshkat et al. using a commercial systems engineering tool, CORE [44]. Using this method, functional models are related to dynamic fault trees to correlate historic risk and failure mode data of components to implemented

functions. As with pervious empirical approaches, model accuracy in this work is directly related to depth and applicability of historical data.

2.4 Conceptual Failure Propagation Analysis

A limitation of these methods is the static nature of the failure analysis results. To overcome this limitation, other work has focused on including the propagation of failure in the analysis. As a direct extension, the Function-Based Failure Propagation method [43] builds upon the FFDM and RED[42] methods to develop failure propagation mapping based on historical data using a functional model for system representation. This method adapted the element of "common interfaces" from change prediction [45] to apply to the functional level. This method explicitly defines failures as propagating along the designed flow paths.

In addition, Wang and Jin [46] present a Bayesian network analysis tool for evaluating the properties of function structures based on dependencies between flows and functions. In this method, the causation relationship is identified between a flow and every functional failure for each identified high-level function. Failure propagation is analyzed using a Function Event Network of all possible causation relationships in the function structure. This type of approach allows for a probabilistic analysis by applying a statistical reliability to the failure of each function in the function structure. An extension of this work is the Conceptual Stress and Conceptual Strength Interference Theory (CSCSIT) method [47], where the conceptual strength of a function is the ability of a function to continue to operate while under normal energy, material and signal (EMS) flows. Conceptual stresses are the EMS flows in the function structure. The application of interference theory is used to define functional faults as when output flow from a function is out of a specified normal range.

The Function Failure Identification and Propagation (FFIP) framework was introduced by Kurtoglu and Tumer [1, 48] as a design stage method for reasoning about failures based on the mapping between components, functions, and nominal and off-nominal behavior. The goal of the FFIP method is also to identify failure propagation paths,

mapping component failure states to function health. However, this approach does not strictly rely on a historical database but instead uses a failure simulation method for determining fault propagation and risk, and provides the designer with the possibility of analyzing functional and component failures and reasoning about their effects downstream in a design based on their nominal and failed state behavior.

3 FFIP for Software/ Hardware Systems

This chapter focuses on the research extending the FFIP framework to analyze failure propagation in a software-driven hardware system. Incorporation of software into FFIP is an advancement in the field of complex system reliability analysis and this work demonstrates how failure propagation identification can be used to evaluate the effectiveness of software control of a system. The use of the FFIP framework allows for a design stage analysis and design comparison. The specific results identify how analyzing failure propagation behavior provides information on the effectiveness of system monitors and key components that act as a nexus for the propagation of failures.

3.1 FFIP Method

FFIP follows a basic set of steps summarized as follows.

1. Create functional model representation of system
2. Create configuration flow graph (CFG) of generalized components
3. Compile generalized component behavioral models into a system model
4. Create a function failure logic (FFL) reasoner to evaluate function health
5. Apply critical fault simulations to system model to identify function failures and the propagation of failure in the system.

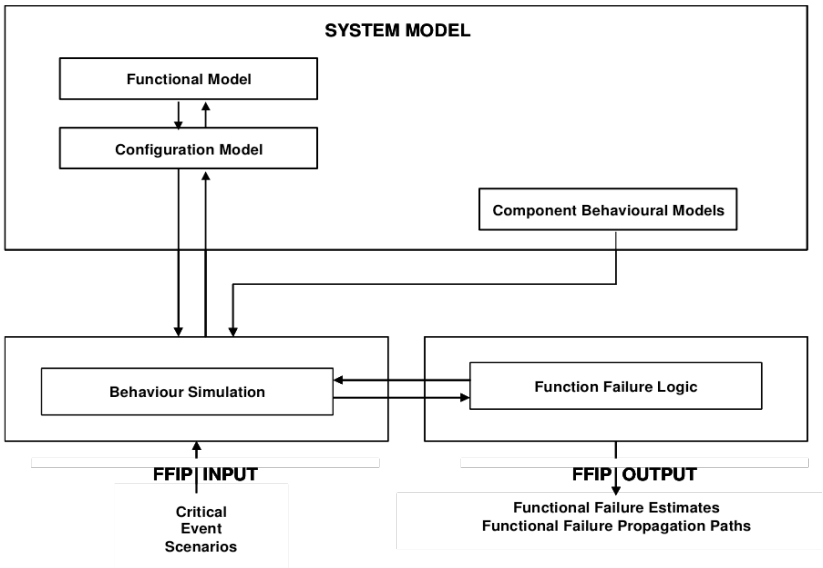
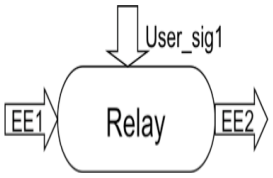


***Figure 1*. The architecture of the FFIP framework [1].**

Figure 1 graphically illustrates the relationship between the system models and reasoning that provides failure propagation information. The objective of the FFIP framework is to relate critical failure events to the health state of functional elements in order to provide risk information to designers. This mapping is done by applying the critical event in the form of component mode changes at discrete time steps. The mode changes determine the behavior of generalized components. A set of component modes would include the operating and identified failed states, e.g., "On", "Off", "Failed On" and "Failed Off." The behavioral models for the generalized components identified in the CFG are created by relating the component modes and EMS flow inputs of a component to the output EMS flows. The configuration flow graph represents the actual design being considered. From the design development stage it is known which components implement each function by creating the configuration flow graph from the functional model. The Function Failure Logic (FFL) reasoner is used to map the EMS flow values entering and exiting components and the health of the functions that are embodied in those components. Figure 2 illustrates part of the FFL reasoner for the "actuate electrical energy" function that is embodied by a generalized relay component.




```
If  (EE1=="Zero")
    {Actuate EE=={Operative}}
Else
    If  (User_sig1=="Closed"
        If (EE2==EE1)
            {Actuate EE=={Operative}}
        Else
            {Actuate EE=={Lost}}
        End
    Else
        If (EE2=="Zero")
            {Actuate EE=={Operative}}
        Else
            {Actuate EE=={Lost}}
        End
End
```

Figure 2. **Example Function Failure Logic FFL for the "actuate electrical energy" function embodied by a relay component.**

3.2 Augmentation of FFIP for Software/Hardware Systems

The proposed methodology is applied by: 1) Creating the component and functional model for the system to be analyzed; 2) Identifying the distinct failed states for each component based on specified input flow and output flows; 3) Identifying system monitors and components that actively affect the propagation of failures; 4) Establishing scenarios of one or more initiating failures; 5) Finally, comparing multiple scenarios to identify effective design changes that mitigate failure. Steps 1 and 2 are the FFIP framework while steps 3-5 reflect the propagation analysis being advanced in current research.

3.2.1 Creating the Configuration and Functional Models

The configuration model represents the actual design being considered and is developed from the functional model. The functional model will be the combination of all the functions that the system must contain. Functional modeling for the hardware components of the system is straight forward and well established [49]. The software components of the system can be functionally modeled and modularized with an object orientated programming or a general systems engineering approach [50, 51].

At this point the flows of energy, material, signal and data are identified and mapped through the system.

3.2.2 Identifying Component Operating States

Because this design paradigm uses abstract functional representations, identifying exact operating states can be challenging. For some mechanical applications the distinct failed states are apparent, for a simple valve, the states would be: failed open, failed closed, or nominal operating. Identifying failed states for functional software components is limited to the knowledge of how that component is structured [50]. For this reason there are two levels of identifying failed states. Components that have unknown failed states can be analyzed as nominal or failed based on their ability to operate on the incoming flow. A nominal state is one where a component is operating on a flow with the exact functionality intended by the designer. A failed state is when a component acts on the flow in a way that it was not designed to, including not acting on the flow at all or only limited action. The distinction between nominal and failed states provides a way for analyzing how the failure affects the component it is propagating through. This allows for linear analysis of failure propagation even with non-linear failures.

3.2.3 Identifying System Monitors

A system monitor is any subcomponent that, by way of a signal, can alter the range of functionality of the parent component. A software example is an “exception handler”, which acts as a functional subcomponent that receives a signal and alters the functional range of a parent computational component. In the hardware world an example can be seen in pressure relief valves which change the functional range of the storage tanks to which they are physically attached. Without the monitor the parent component would lose functionality based on certain input flows. For the software example, that means the parent mathematical operation could not have operated on the

incoming data or signal flow. In the mechanical example the storage tank could not store the incoming material flow based on capacity. However, with the action of the monitor the parent functional component can operate on the flow without loss of functionality. This step is critical for determining how the system can alter the propagation failure.

3.2.4 Establishing Scenarios

Once the model is established, a scenario can be analyzed using the knowledge of the system component behavior. A scenario is evaluated according to the rules of the FFL reasoner over discrete time steps. A fault or error is induced at any functional component and allowed to naturally propagate through the system. Using FFIP it is possible to have multiple fault injections points. This can be used to analyze system reliability in scenarios of multiple simultaneous and independent component failures.

3.2.5 Comparing Multiple Scenarios

Finally, by a comparison of multiple component failures, a pattern of fault propagation paths can be mapped. This mapping will reveal key components that require monitors for early fault detection. Additionally, the mapping can be used to compare different system configurations.

3.3 Case Study: RCS Jet-Failed Leak Monitor

For an example system this method is applied to the redundancy management system of the reaction control system (RCS) jet. This example has been explained and developed in [52, 53]. These RCS jets are responsible for the maneuvering of the space shuttle as well as other space vehicles. These jets maneuver the vehicle through a controlled combustion of fuel and an oxidizer. The redundancy management software is a subset of the larger data processing software and is involved in jet

selection (there are 44 on the shuttle), warning systems, and pilot control. There are several parts of the redundancy management software that deal with monitoring RCS jet operation. The software component that monitors for leaks does so by evaluating temperature data from the fuel and oxidizer injectors and flags a jet as having a failure by leak if the temperature data is out of bounds for three or more cycles. This software component is called the "Jet-Failed Leak monitor." There are other monitors for detecting jet failures based on firing commands and reaction chamber pressure.

3.3.1 System Representation

Figure 3 demonstrates a possible early stage design model for the RCS system. In this model, blocks represent system components and lines connecting the blocks represent the material, energy, or information between components. (For clarity, solid lines represent material and dashed lines represent signals). It is important to note that most of the component blocks could be further broken down into smaller subcomponents and flows. There is no minimum level of model depth or complexity for FFIP, making it a useful tool in the design stage when there is limited system knowledge. However, as with all models greater fidelity provides more system information.

Under designed operation, a leak in either the fuel or the oxidizer lines is monitored with the use of temperature sensors on the injectors and in the jet exhaust. The temperature signals, along with chamber pressure and valve manifold position, are bundled in the multiplexer and sent to the shuttle computers and to ground computers. The redundancy management software receives the signals from the RCS of temperature, pressure, and valve position for all 44 jets and also receives a signal from their reaction jet drivers (RJDs) indicating the command that was sent to the fuel and oxidizer valves. The failed leak monitor will flag a leak after three clock cycles. The flag becomes a warning that is sent to the operator console. The flight control system can send a command to the RCS redundancy management software that inhibits the software and allows the digital autopilot (DAP) to ignore the warnings from the redundancy monitors. Under manual flight control the DAP collects the jet

functionality information from all of the monitors in the redundancy management and selects the jets to be fired based on jet availability and the maneuvering information from the operator. The DAP sends fire commands to the individual RJDs which in turn open or close valves as mentioned previously. Due to the complexity of this system, only the components involved with the jet failed leak monitor are developed in the later failure scenarios.

Applying the FFIP framework to this system requires the development of a functional model and a behavior model. The functional model for this system can be seen in Figures 4 and 5. Figure 5 is simply an expansion of the failed leak monitor to illustrate that the software and hardware components of the system can be evaluated at similar fidelity levels. In this model blocks represent functional components of the system that are directly linked to the configuration model. Some functions are not represented such as transport for the signals and material flows. The O-ring failure from the Challenger shuttle tragedy clearly illustrates that these lesser implied functions should certainly be considered in a more thorough analysis. For simplicity, however, this example lumps these functions into other components that are represented in the model. For example a leak in the fuel transport lines can be seen as a leak in the fuel tank, either would have the same effect on the functionality of the RCS control while exact leak location would have greater importance for other failures. The flow of material and signal in this model is represented by the lines connecting the functional blocks. For clarity the material flow of fuel and oxidizer are represented with solid lines while the signals are separated between dashed and dotted lines. The dashed lines are generally assumed to be analog signals or voltages and the lighter colored lines represent command signals. The dotted lines represent data and are the primary flow within the software components of this system. Again, there are flows not explicitly represented in this example that are instead represented by broader functional blocks.

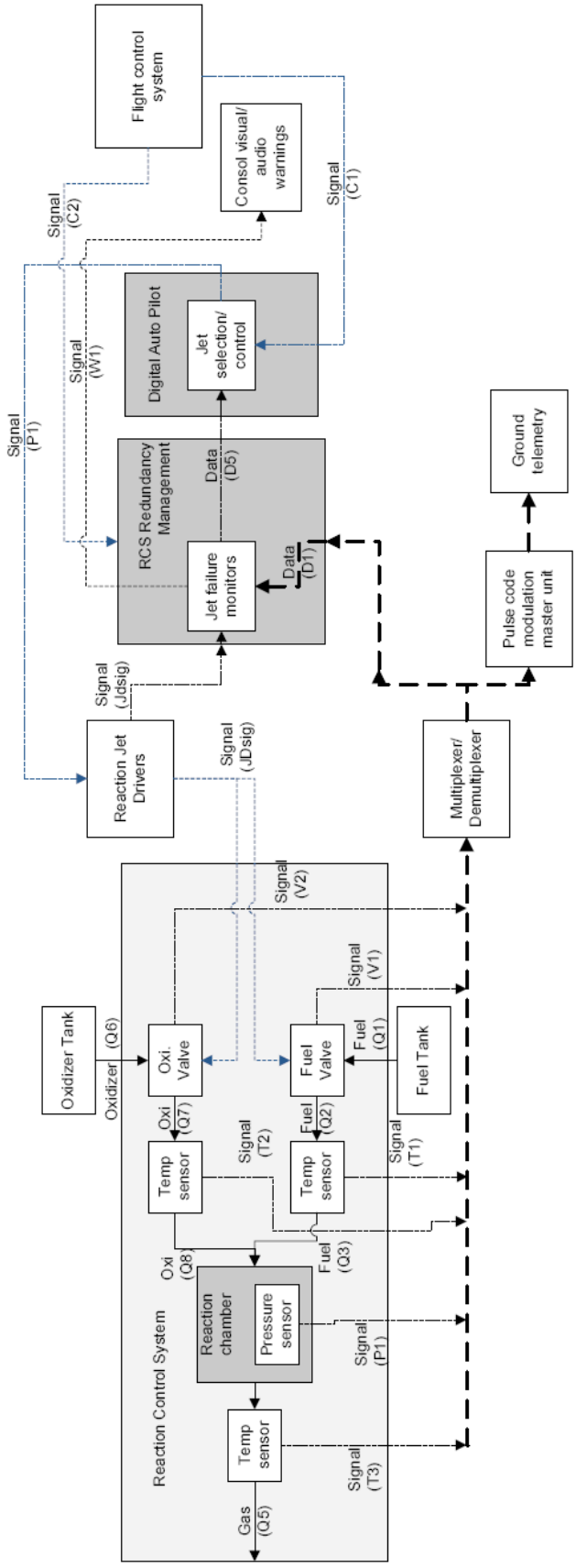


Figure 3. **RCS jet configuration flow graph.**

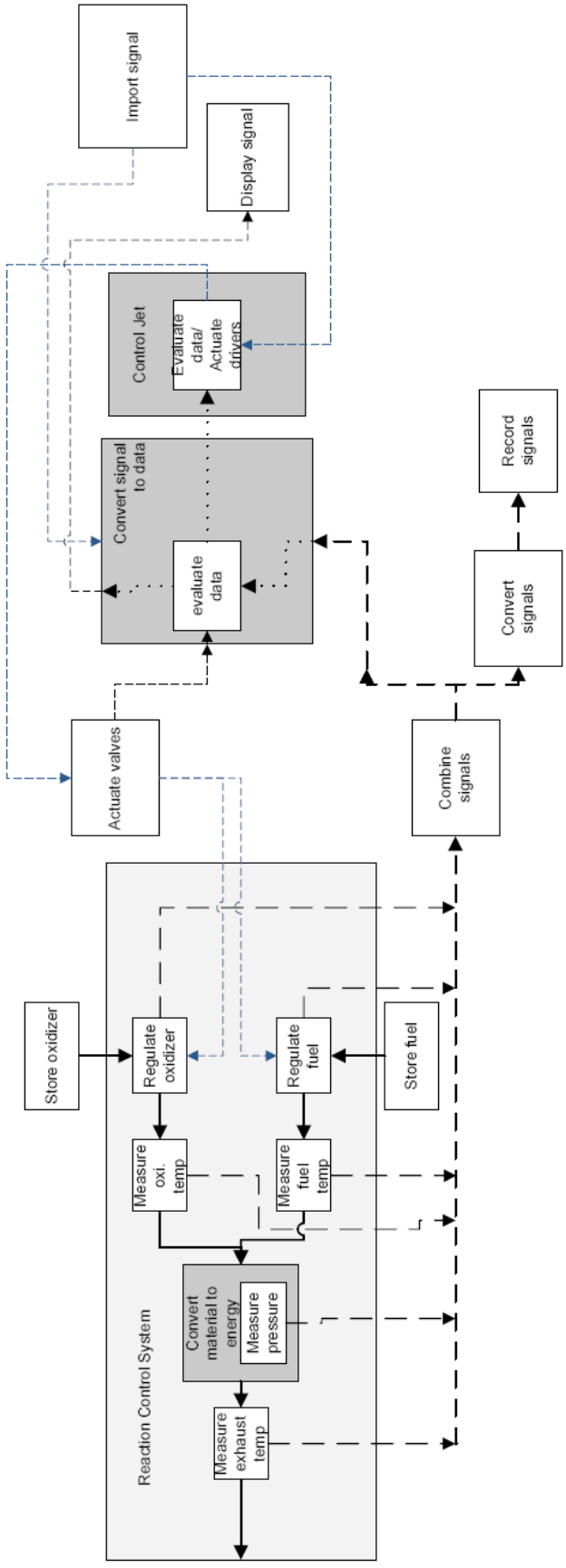


***Figure 4*. RCS jet functional model.**

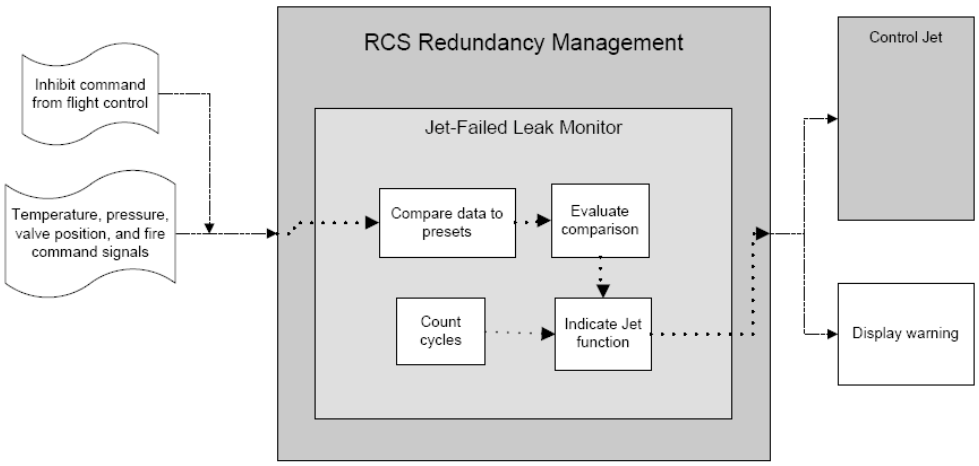


Figure 5. **Functional model of RCS Jet-Failed Leak monitor.**

3.3.2 Behavioral Model

The component behavior models are the known states of each component, both operating and in failure. Several different repositories could be used to determine the possible failed states for the functional representations of hardware components. The behavior models of valves and sensor are fairly generic across systems but the system specific components indicate the benefit of system specific repositories for thorough model analysis. Because the software components are represented in a functional model and not in architectural form, component behavior is generally limited to "functioning" and not "functioning."

3.33 System Monitor Identification

The system monitors are easily identified by name. The failed leak monitor, through the digital auto pilot, alters the functionality of the RCS so that functionality of the RCS is not lost. The other redundancy management monitors behave likewise. An additional system monitor is found in the pilot controls and the RCS redundancy management inhibit command. Flight control can alter the functionality of the redundancy software with the inhibit command, preserving the functionality of the system as a whole.

3.3.4 Failure Scenarios

The failures within a scenario are reasoned through the FFL rules. Two example FFL rule sets can be seen in Figure 6. Using FFL logic rules two failure scenarios and the propagation of failure through the system are represented in Figures 7 through 10. In the first failure scenario, illustrated in Figure 7 and Figure 8, the initiating failure is the temperature sensor sending an inaccurate temperature signal. The signal sent from this sensor is not indicative of the actual temperature of the incoming fuel making this the propagation path of failure in the system. This failure propagates nominally through the system without affecting the functionality of any system components. The failed-leak monitor operates on the incoming signal and flags that jet as failed. The DAP automatically deselects that jet, finally leading to the loss of functionality of the jet. At this point the flight controller, based on other sensor data, inhibits the RCS redundancy management software allowing the DAP to return functionality to the previously deselected RCS jet.

The second scenario, shown in Figure 9 and Figure 10, presents both a software and a hardware failure. The two initiating failures are a leak in the fuel containment system and a fault in the comparison function of the failed-leak monitor. The leak in the fuel storage decreases the flow that the fuel valve operates on causing failure to propagate through that component in such a way as to lose the designed functionality. Failure then propagates through the temperature sensor and to the jet where the functionality is again lost due to insufficient flow. The branch of failure that went through the temperature sensor propagates through nominally. The fault in the software comparison function causes the loss of functionality of the evaluator component. The functional failure of the evaluator component means that a failed leak flag is not triggered and no fault information is delivered to the DAP. If these two faults are simultaneous or the software function fails first the operator may invoke the inhibit command to bypass the RCS redundancy software. This is shown with the upper propagation path. The faulty command from the flight controller propagates through the system nominally and ends in the lose of function of the RCS jet.

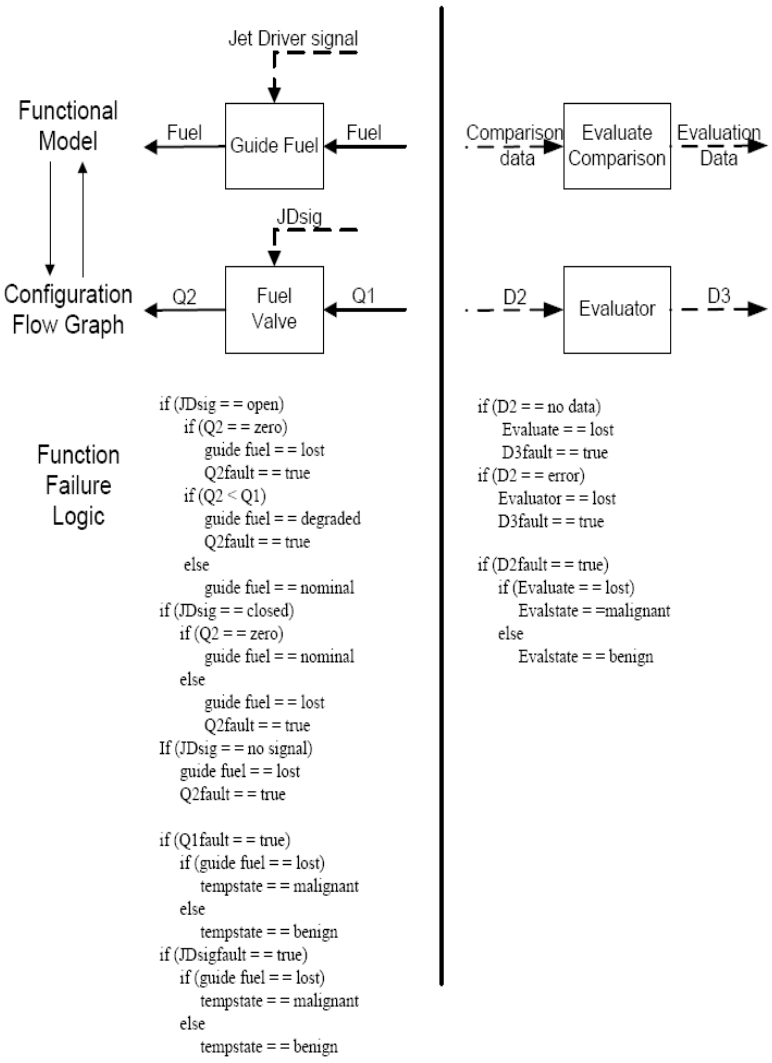


Figure 6. **Function failure logic for guide liquid and evaluate comparison.**

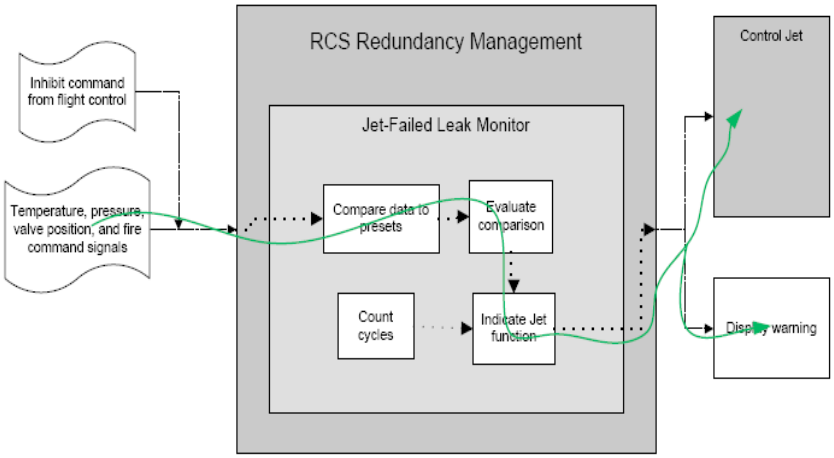


Figure 8**. Software section for scenario 1, Temperature sensor failure.**

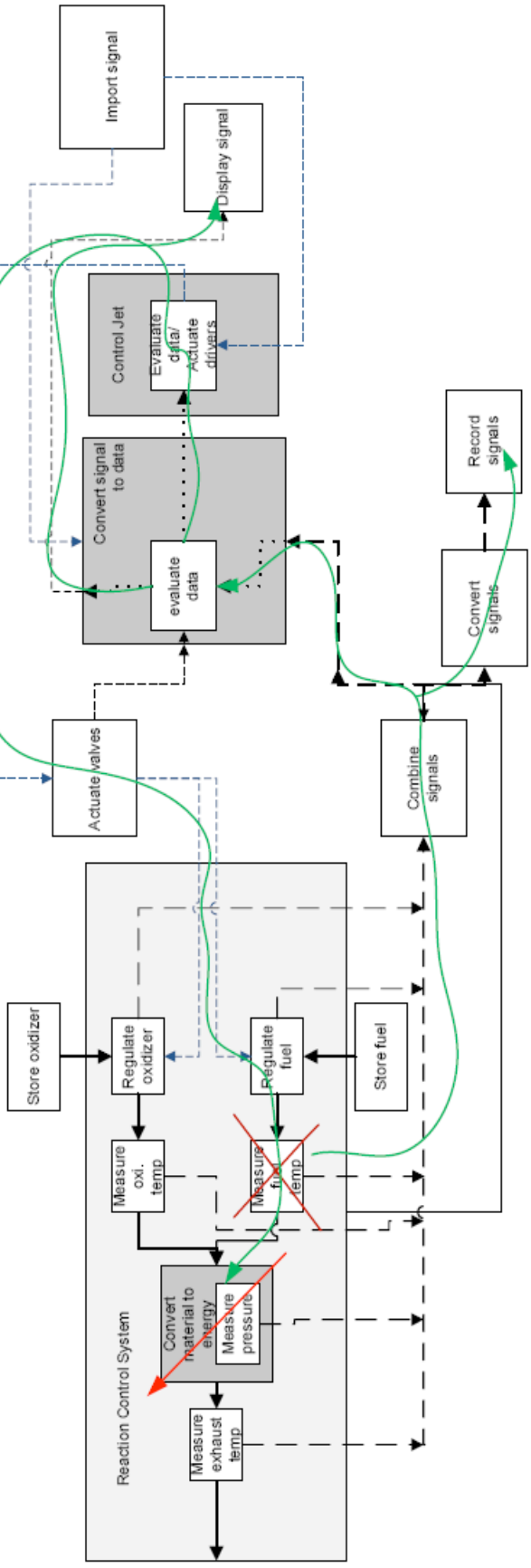


Figure 7. **Scenario 1, Temperature sensor failure.**

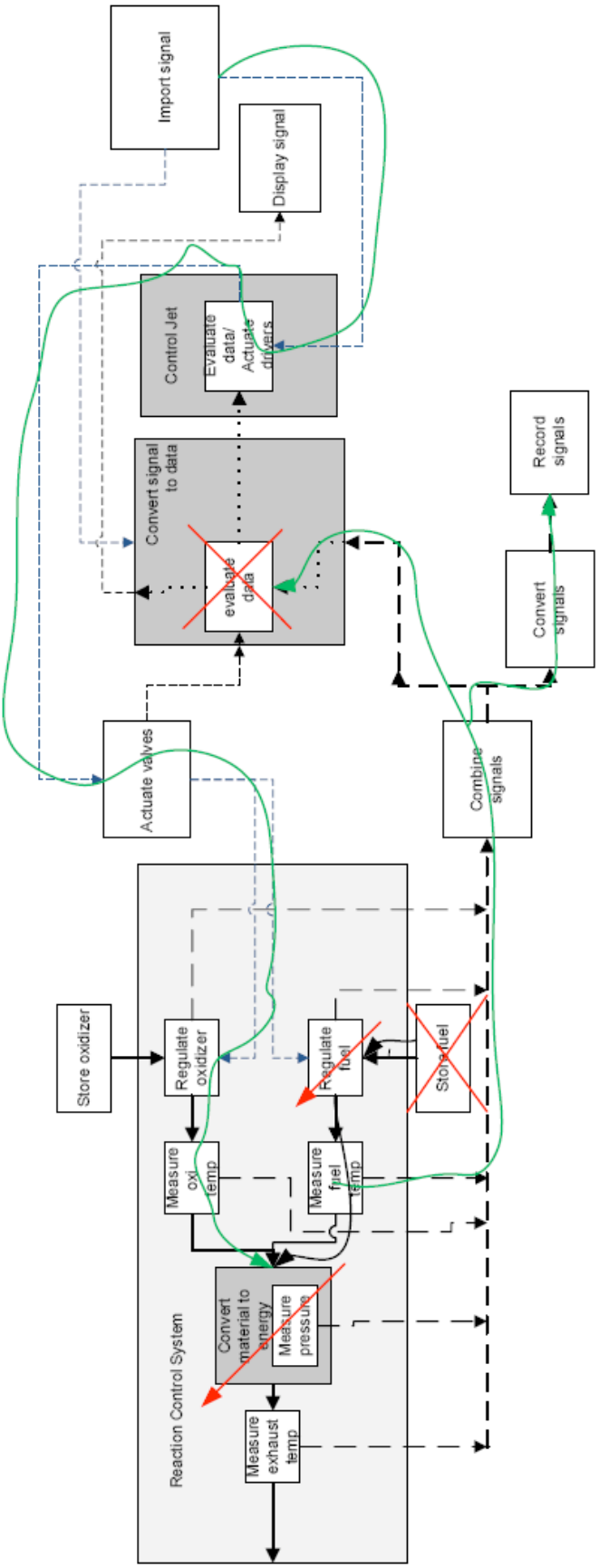


***Figure 9*. Scenario 2, Tank leak and software failure.**

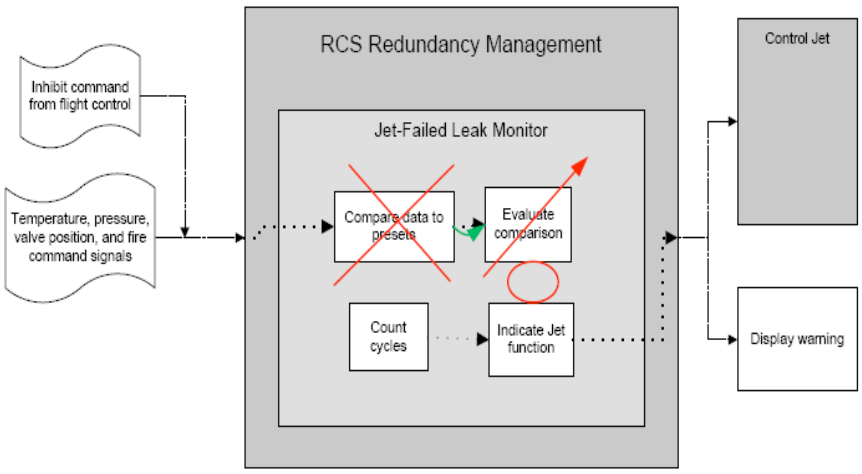


***Figure 10*. Scenario 2, Software fault.**

3.3.5 Scenario Comparison

The result of comparing these two scenarios reveals important design safety information. The failed-leak monitor and the flight controller both act as monitors for this system. In the first scenario if only the failed-leak monitor acted on the system then the RCS jet function would be lost due to inaccurate temperature data. However, the presence of a second monitor can restore the functionality of the jet. In the second scenario the failed-leak monitor is disabled and the flight control monitor fails to identify the leak failure. These two scenarios illustrate a possible insufficiency in system monitoring by the failed-leak monitor and the flight controller monitor. Both monitors must be in operation for safe control of the RCS jets. Depending on the criticality and probabilities of these failure modes a designer might consider redundant monitoring through pressure or flow sensors would be one way to improve monitor reliability. Alternatively a different system configuration that provides more pertinent information to all the monitors would improve the effectiveness of each monitor. For example if the flight controller also received the injector temperature information then the leak in the second scenario could have been identified. If the failed-leak monitor also evaluated pressure or flow information then a failed temperature sensor would not propagate through the system as in the first scenario.

3.4 Discussion of SW-HW FFIP

Evaluating the above two example failure scenarios provides insight into how the FFIP framework can be extended to analyze software-driven hardware systems. Where previous work focused on highlighting component failures, the proposed methodology highlights the propagation path of failure in a system. This additional information for the above example highlighted the strong dependency of the two system monitors and the path of that dependency. Although it is possible that the dependency of the monitors could be inferred through previous FFIP evaluation, the key components of that dependency would not be readily apparent. Secondly, this paper illustrates how the FFIP framework can be expanded to integrate software and hardware components and evaluate simultaneous, independent failures.

In addition to the above analysis, the way that propagation of failure affects system components could be used for design decision-making. In the first example scenario, the failure initiated by the temperature sensor propagates through the system without affecting the functionality of any other component until it reaches the actual jet. Failures that propagate through a component can be defined as benign when they do not affect the functionality of that component and malignant when they do affect the functionality. This analysis shows that the first scenario has a predominantly benign failure path, which as a result, goes unnoticed by the system monitors. In contrast, in the second scenario multiple components change functional states as a result of failure propagating through them. This latter propagation path would be malignant through these components. The biological analogy applies to the component level and ends at the system level. Because of multiple component failures, system monitors are provided with more accurate information regarding system state. In the second scenario, valve failure information would also indicate a fuel leak to the redundancy software. In practice, this type of failure propagation analysis could be used to design systems to have key component failures to provide early warning information to system monitors. By designing systems with low cost or easily repaired/replaced components that consistently fail and can be monitored, more important components

can be saved by informed system monitors. This is already done in simple safety critical hardware such as automotive hydraulic jacks. The functionality of the lever handle is usually designed to be lost before the functionality of the hydraulic piston, effectively mitigating the propagation of failure from the initiating fault of too high a force on the jack.

For complex systems it is necessary to establish what type of components, both hardware and software can act as key failure points. Hardware components can be somewhat straight-forward and may be of assistance in establishing the kinds of software components being sought. Key hardware failure points are simple and inexpensive, such as pipe-valve combinations that automatically limit flow, redundant sensors or signal evaluators. All of these hardware components are only key points of failure propagation based on specific system configuration. It is reasonable then, that software specific key components for failure propagation would be dependent on software architecture. Therefore, it is clear that a form of software architecture must be included in the FFIP configuration model. Although the proposed methodology, within the FFIP framework, modeled both the software and hardware components in one integrated model, the software side still represents an area that requires further research.

As well as research into the identification of what could act as key components for failure propagation in software, mechanisms for handling the functional flows for software should also be developed. In the example scenarios it was shown how failure flowed from physical components along the material, energy and signal flows. However, in a software fault, as described in the second scenario, the failure meant that there was no fault information passed between components yet successive components did fail for lack of that fault information. It is yet to be established if this is consistently the case for software faults or if there are alternative fault behaviors.

3.5 Summary - Necessity for Analytical Failure Reasoning

This chapter presented the extension of the FFIP framework, developed by Kurtoglu and Tumer in [1, 54] to include software by evaluating a software-driven hardware system. In addition, the FFIP framework is expanded to include failure path identification and characterization with respect to function. The results of this latter addition allow for the evaluation of system monitors and the comparison of system monitor designs. Through evaluating two failure scenarios with the proposed methodology, the dependency and inadequacy of the two software monitors was found and design changes became readily apparent from the results of the analysis. However, all though critical scenarios were used in this analysis, the impact of those scenarios was left to interpretation. It would be possible for two designers to apply the same scenario and interpret the results or the impact to the system differently. This interpretation would be largely based on expert knowledge and therefore provides minimal design decision-making support. The results from this stage of the research indicate a need for an analytical evaluation of the impact of failure. To meet this need the method of Function Failure Reasoning was developed and is presented in the following chapter.

4 Decision-Making Using FFIP and Function Failure Reasoning

The functional failure reasoning (FFR) method, introduced in this chapter, uses the FFIP analysis framework to perform a simulation-based analysis of functional failure propagation as a first step, and then associates that analysis with the criticality of functional losses to enable tradeoffs between competing conceptual system architectures [55]. The FFIP framework uses a simulation-based analysis of functional failure propagation. The main novelty of the FFR method introduced in this chapter is that it presents a conceptual design tool that enables robust and reliable system design and development of complex systems during the stages of design where only functionality and basic (generic) configuration information is available. Note that, only redundancy decisions are targeted in this chapter, however, the method is applicable to other design decisions governing the configuration of a system. The FFR method offers two immediate advantages by:

1) Accounting for the individual impact of failure of basic functional elements in a system as well as the combined system-level impact resulting from the propagation of functional failures.

2) Helping to determine the level of risk mitigation that can be achieved by alternative system architectures by computing the effect of architectural changes on functional failures and the impact on the overall system safety.

The basis for these analyses is a four-step process, which is explained next, followed by an application to the Electrical Power System (EPS) testbed.

4.1 System-Level Modeling

The first step in the methodology is to apply the system-level modeling module of the FFIP framework to represent system function, architecture, and behavior by an interrelated array of graph-based, elemental component models [1]. The graph-based modeling approach provides a coherent, consistent, and formal schema to capture function-configuration-behavior architecture of a system at an abstract level.

4.1.1 Functional representation

System function is represented using function structures [30, 56, 57], establishing a formal function-based design paradigm based on the concept of functional modeling [37, 57, 58]. Functions and flows are represented as verbs and nouns respectively (e.g., transfer gas, mix liquid, open gate, display warning, record data, etc.). The flows are broken down into three categories: energy, material, and signal. The Functional Basis (FB) taxonomy, with its hierarchical set of flows and functions [37, 58, 59], and the functional modeling processes proposed in the literature [30, 56] are used to develop the functional models for the systems under study.

4.1.2 Architectural (Configuration) Representation

The architecture, on the other hand, is captured using configuration flow graphs (CFGs) [49]. A CFG strictly follows the functional topology of a system and maps the desired functionality into the component configuration domain. In a CFG, nodes of the graph represent system components, whereas arcs represent energy, material or signal flows between them. For flow naming, the Functional Basis terminology is adopted, while the components of the graph are named using a taxonomy of standard components [60]. The component types in a CFG can be thought of as generic abstractions of common component concepts.

4.1.3 Behavioral Representation

Finally, the behavior of the system is represented using a component-oriented modeling approach. The approach involves the development of high-level, qualitative behavior models of system components in various discrete nominal and faulty modes. The transitions between these discrete modes are defined by mode transitions. The component behavior in each mode is derived from input-output relations and underlying first principles. These modular, reusable component behavior models follow the form of configuration flow graphs. Accordingly, state variables critical to the system behavior are incorporated into the representation by associating them with their respective (CFG) flows [1].

4.2 Computing the Function Criticality Rating (FCR)

The objective of Step 2 in the methodology is to determine how critical each system function is for the system's operations. The method accomplishes this task by estimating the distribution of criticality over the functional elements of a system. Accordingly, the function criticality rating (FCR) for each system sub-function is determined by comparing the criticality of individual system functions and by converting the criticality ratings into a coefficient that is normalized based on the combined criticality of all system functions.

This is accomplished by means of functional decomposition [30], where the overall function of a system (i.e., the black-box representation) is decomposed into smaller lower level functions in a hierarchical manner. The top-level functions are those that are implemented by all alternative conceptual system architectures. Each of these functions are then further decomposed into lower level, elemental sub-functions depending on the specific implementation of the architectural design of the system. This process is schematically illustrated in Figure 11 where two system architectures derived from the same overall function are shown. In Figure 11a, top-level functions 1 and 3 are implemented by three elemental sub-functions, whereas top-level function 2 is an elemental sub-function itself, and in Figure 11b functions 1 and 2 are implemented by two elemental sub-functions and function 3 is implemented using three sub-functions.

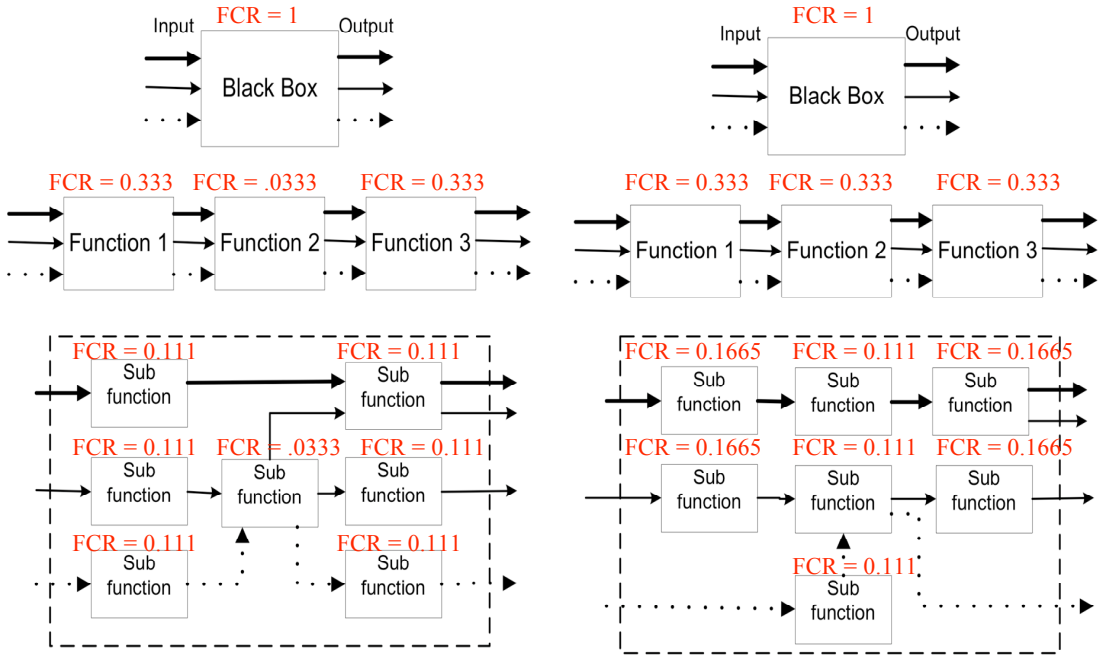


***Figure 11*. Illustration of functional decomposition and the estimation of functional criticality ratings: (a) FCR values for system architecture A estimated using equal criticality distribution, (b) FCR values for system architecture B estimated using skewed criticality distribution.**

The function criticality rating (FCR) of each elemental sub-function is then estimated by following the functional decomposition scheme introduced. Accordingly, the overall or the black-box function of the system is assigned an FCR value of 1, which is then distributed over the lower level functional elements in the system. In order to accomplish this, the functional criticality of the top-level functions is assigned first, followed by further projection of FCR values to lower level functions as shown in Figure 11. The FCR ratings for lower-level functions can be assigned by soliciting expert opinion on functional importance, or alternatively by progressively projecting higher-level FCR values on to lower-level functions based on a distribution scheme. In Figure 11a, an equal criticality distribution is assumed for the projection of FCR values, whereas in Figure 11b, a skewed distribution is used which assigns a higher criticality to the top-level Function 1.

At the end, the individual functional criticality ratings constitute the relative weight of each system function based on overall system functionality and provide an expected distribution of risk over functional elements. In other words, the higher the FCR of a function, the more valuable is maintaining that functionality during system operations.

4.3 Computing the Functional Failure Impact (FFI)

The objective of Step 3 in the methodology is to quantify the overall impact of functional failures and their propagation on system functionality. The "consequential cost" of functional failures is calculated in this step by following 4 different sub-steps, described next.

4.3.1 Selecting a Set of Scenarios of Interest

Critical scenarios are determined based on the concept of operations of a particular system. The FFIP framework is then used to analyze the consequences of these what-if scenarios in a system governed by the occurrence of specific component failures [1].

4.3.2 Running the FFIP Behavioral Simulation

The FFIP behavioral simulator then determines the system behavior under certain specified conditions for the critical scenarios. These conditions are represented by the occurrence of events that cause specific component mode transitions. During the simulation, both the discrete component modes and the system state variables are tracked.

During conceptual design, the system state variables are not known quantitatively. To deal with this constraint, these continuous variables are discretized into a set of qualitative values. For example, an electrical current variable may take on values from the set of {zero, low, nominal, high}. Similarly, a status signal variable indicating the position of a circuit breaker may have values of {open, closed}.

The state of the system is then simulated by solving the continuous-time system in the intervals between discrete events. When an event occurs, the continuous-time simulation is stopped, and the corresponding component mode transition is executed.

Using this scheme, critical events, consequences of which are investigated, can be inserted into the simulation at any time step [1]. Examples are answers to questions such as: “How does the system behave if a relay fails to open?” or, “What is the impact of an AC/DC inverter breakdown on overall system behavior?”

4.3.3 Running the FFIP Function Failure Logic

Next, the function-failure logic (FFL) module of the FFIP framework uses its reasoner to determine the state of each system function (i.e., whether it is operational, degraded, or lost.) The simulation feeds the state of the system to the FFL reasoner at the end of each time step and the state of each system function gets evaluated at these discrete points. The FFL reasoner translates the dynamics of the system into functional failure identifiers and facilitates the assessment of potential functional failures and resulting fault propagation paths.

Note that, FFL allows the assessment of the operability of a function to be made based on the values of the input and output state variables of the CFG that corresponds to the component by which the function is realized. Therefore, capturing the mapping between the functional model (function) and the configuration flow graph (behavior) is fundamental to the employment of the function failure logic. The reasoner uses a set of form-independent system function models that describe conditions under which functions deviate from their intended operation [1]. Accordingly, system functions are classified as ‘operating’, ‘degraded’, ‘lost recoverably’ or ‘lost’, defined as follows:

Operating: Function operates on a flow as designed

Degraded: Function operates on a flow not as designed

Lost Recoverably: Function has no flow to operate on because of a different functional failure

Lost: Function does not operate on flow

Using the simulation scheme of FFIP, functions that can potentially be degraded, lost, or lost recoverably can be computed for particular scenarios of interest.

4.3.4 Calculating the Functional Failure Impact (FFI)

Finally, after the FFIP analysis is run, the Functional Failure Impact of a selected scenario can be calculated by simply summing over the "Functional Criticality Ratings (FCR)" of all functions that are classified as "degraded", "lost", or "lost recoverably" during the simulation using:

$$\textit{Functional Failure Impact: } FFI = \Sigma\, C_i \, x \, FCR_i \qquad (1)$$

where C_i is the consequential cost factor determined by the functional state of function i (Table 1), and FCR_i is the functional criticality rating of function i as determined by the behavioral simulation.

Table 1. **Function states and consequential cost factors**

Functional State	**Consequential Cost Factor, C_i**
Operating	0
Lost Recoverably	1
Degraded	2
Lost	4

The consequential cost factors shown in Table 1 can be defined based on the requirements of a particular application. As defined by (1), the consequential cost factor multiplied by the FCR provides a function failure impact (FFI) for an elemental sub-function in the system. The sum of all sub-function impacts is the system level FFI.

Naturally, the estimated loss of functions with higher criticality ratings will result in higher functional failure impact for the system. As stated earlier, this process allows the system designers and risk analysts to quantify the overall impact of functional failures and their propagation on the functional operability of the system. This quantification of the functional failure impact is crucial for exploring alternative system architectures, as it constitutes a formal basis for making design decisions relevant to risk management in general and for risk mitigation in particular. The way

in which these alternative system architectures are explored is summarized in the next step.

4.4 Computing the Reduction in Risk (RIR)

The proposed design methodology is based on the assumption that one can reduce the severity of consequences of failures by making architectural changes to mitigate risks associated with certain functional elements in the system. This can be done, for example, by placing more sensors in a sub-system, designing in more redundancy, changing the configuration of the sub-system by the addition or removal of certain components, or by introducing new technologies.

The objective of Step 4 is to quantify the level of mitigation a designer can achieve by making such architectural changes. The FFR method accomplishes this by first calculating the consequential cost of functional failures for a modified design under the same set of critical scenarios used in Step 3. The "Functional Failure Impact of the modified design (FFIm)" is then computed by making the necessary modeling changes, and by running the FFIP analysis under the same set of scenarios for the modified design. Finally, the "Reduction in Risk (RIR)" is calculated, expressed in percentage by using Eqn. 2:

$$RIR = \% (FFIm\text{-}FFI)/FFI * 100 \quad (2)$$

The RIR value formally quantifies the amount in risk reduction based on a specific architectural change. The RIR value can be used to determine the decisions that most efficiently mitigate risks associated with functional elements in a design. Moreover, it allows system designers to assess system safety beginning from the very early stages of design, and to explore various conceptual design alternatives guided by safety and reliability requirements. The next section demonstrates this by applying the proposed approach to the design of the previously introduced electrical power system. In this analysis, three alternative conceptual system architectures are compared and evaluated to the baseline design for an electrical power system.

4.5 Case Study of Electrical Power System (EPS)

Motivated by the critical role electrical power systems (EPS) play in most complex systems, the Advanced Diagnostic and Prognostic Testbed (ADAPT) at NASA Ames Research Center provides a representative aerospace vehicle electrical power system that enables automated diagnosis of faults in a physical software-hardware testbed. The EPS testbed (Figure 12) is designed to deliver power to select loads, which in an aerospace vehicle would include subsystems such as the avionics, propulsion, life support, and thermal management systems. The EPS is required to provide basic functionality common to many aerospace applications: power storage, power distribution, and operation of loads [61].

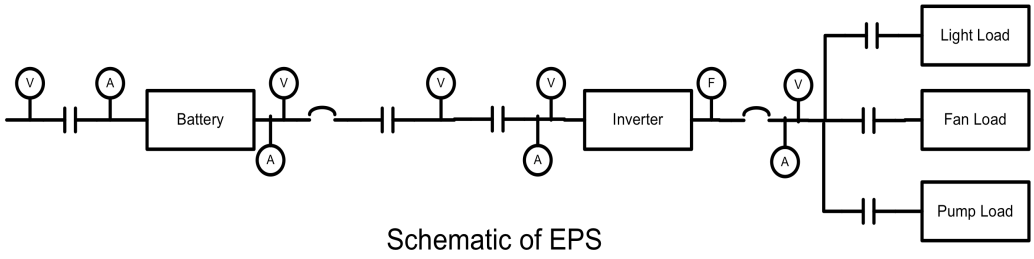


***Figure 12*. The schematic of the basic electrical power system (EPS) design. The system is used to provide power to various components including pumps, fans, etc.**

4.5.1 EPS Testbed Baseline Modeling Using the FFIP Framework

The EPS testbed was originally designed using the Function-Failure Based Design (FFDM) methodology described in Section 2 at the early concept design phase [62]. Using the function-based design approach, several critical elements were identified and incorporated into the final design and realization of the testbed. In the current realization of the testbed, the power storage can consist of one or multiple battery modules, which are used to store energy for the operation of the loads. Any of the battery modules can be used to power any number of loads in the system. This

requires the EPS testbed to have basic redundancy and reconfiguration capability. Electromechanical relays or other electrical actuators can be used to route the power from the batteries to the loads. In addition, circuit breakers are needed at various points in the distribution network to prevent over currents from causing unintended damage to the system components. Moreover, a sensor suite is required to allow monitoring of voltages, currents, temperatures, switch positions, etc. and to provide an integrated health management functionality. (More information on the existing ADAPT testbed can be found in [61].

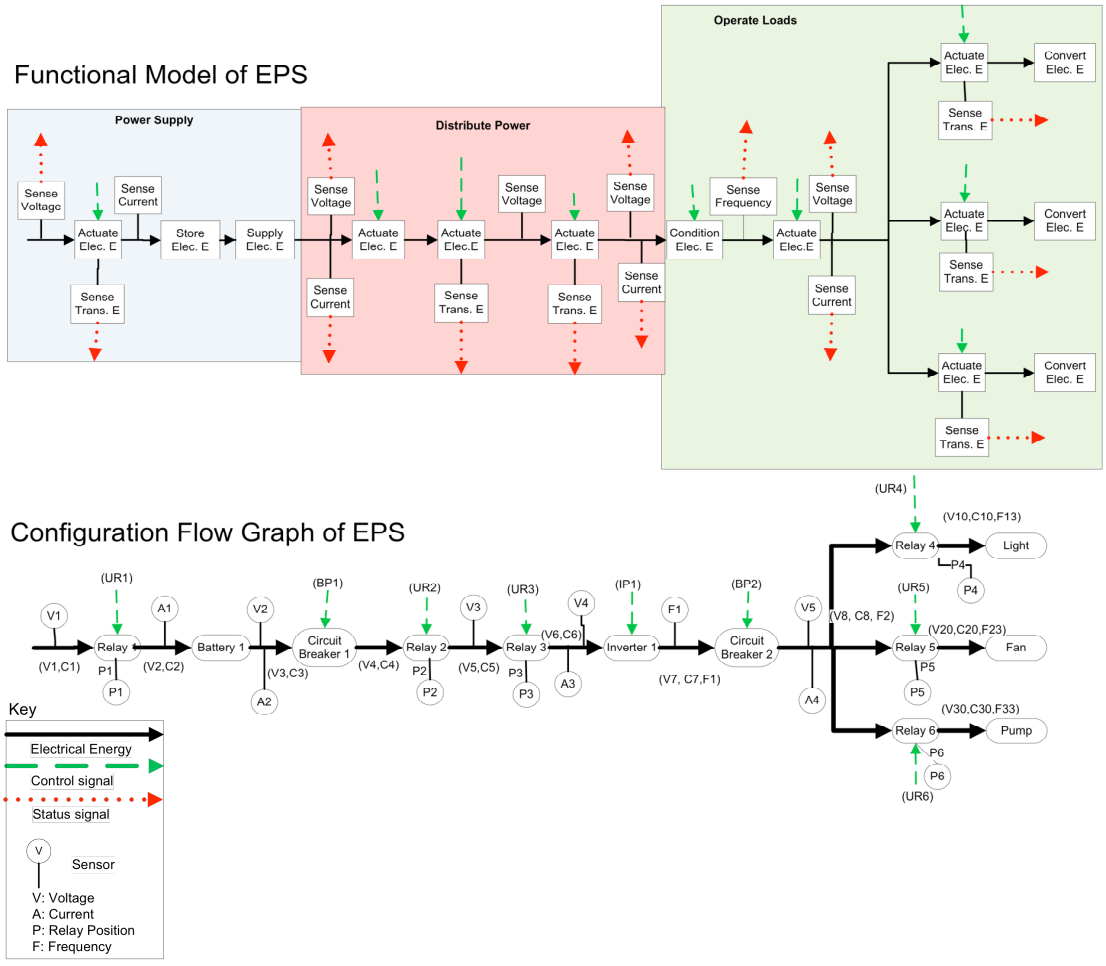


***Figure 13*. A high-level functional and architectural model of the electrical power system baseline design. The functional model illustrates the three top-level functions (i.e. supply power, distribute power, and operate loads) of the system and their decomposition.**

Figure 13 shows a functional and a configuration model of the electrical power system (EPS), which is used as the baseline architecture in this paper. The construction of a functional model (FM) and the corresponding configuration flow graph (CFG) captures a direct mapping between the functional and the structural architecture of a system. Each mapping represents a transformation that shows how a functional requirement was addressed in the actual design by the use of a specific component concept. In the electrical power system example of Figure 13, the component "battery 1" addresses functions "store electrical energy", and "supply electrical energy". Similarly, the component "inverter 1" provides "condition electrical energy" function in the system. Capturing this mapping between functionality and component configuration of a system is crucial for accurately reasoning about failures at a functional level.

For the system configuration flow graph shown in Figure 13, there are 42 state variables (attached to the arcs of the CFG). Also the 29 components of the system have a total of 42 distinct behavioral modes. These state variables and component modes are identified in Tables 2 and 3.

***Table 2*. Components modes for the baseline EPS design.**

Component	**Modes**					
Relay	Nominal Closed	Nominal Open	Stuck Closed	Stuck Open		
Breaker	Nominal Closed	Nominal Tripped	Failed Open	No Trip		
Battery	Nominal	Level Loss	Terminal Short			
Inverter	Nominal On	Nominal Off	Failed Off			
Load-Light Bank	Nominal On	Nominal Off	Failed Off			
Load-Fan	Nominal	Over speed	Under speed			
Load-Pump/valve	Nominal	Blocked Flow				
Relay position sensor	Nominal	No Signal	Stuck at Open	Stuck at Closed		
Voltage Sensor	Nominal	No Signal	Stuck at Zero	Stuck at Low	Stuck at low, nominal, or high	Drift up or down
Current Sensor	Nominal	No Signal	Stuck at Zero	Stuck at Low	Stuck at low, nominal, or high	Drift up or down
Frequency Sensor	Nominal	No Signal	Stuck at Zero	Stuck at Low	Stuck at low, nominal, or high	Drift up or down

***Table 3*. State variables for the baseline EPS design.**

Flows	**States**				
V#	Electrical Energy, Voltage	Zero	Low	Nominal	High
C#	Electrical Energy, Current	Zero	Low	Nominal	High
F#	Electrical Energy, Frequency	Zero	Low	Nominal	High
P#	Translational Energy	Zero	Low	Nominal	High
UR#	Control Signal, Relay Position	Open	Closed		
BP#	Control Signal, Breaker Switch	Open	Closed		
IP#	Control Signal, Inverter Power	Open	Closed		
VS#	Status Signal, Voltage	Zero	Low	Nominal	High
CS1#	Status Signal, Current	Zero	Low	Nominal	High
FS#	Status Signal, Frequency	Zero	Low	Nominal	High
RS#	Status Signal, Relay Position	Open	Closed		

To illustrate the component-oriented behavioral modeling approach of FFIP, Figure 14 shows behavioral models for two generic components ("relay" and "inverter") from the baseline EPS system of Figure 13. A relay can transition from a "nominal open" (or "nominal closed") mode to "stuck open" or "stuck closed" modes as a result of a fault event. Similarly, an inverter can operate nominally, or failed to operate. ("failed off"). The dynamic behavior of the component in each of these discrete modes is governed by a different set of physical laws and mathematical relations, and is therefore defined separately.

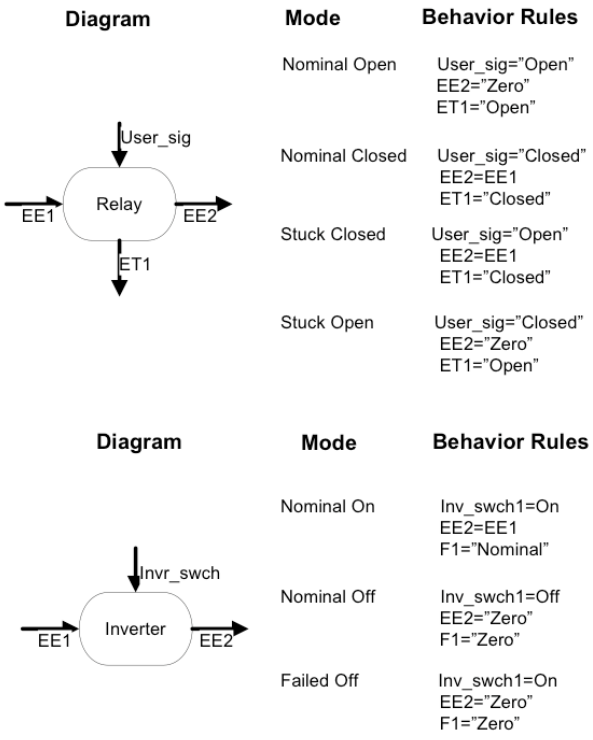


Figure 14. **Behavioral models for generic "relay" and "inverter" components.**

Finally, Figure 15 presents three rules for the electrical power system example. The first rule defines the failure logic for the "actuate electrical energy" function that is addressed by a generic "relay" component. The state of this function is classified depending on the values of the input control signal (User_sig), and the output current (EE2) of the relay. This rule basically states that, the function "actuate electrical energy" will be lost if there is no outflow from the relay when it is commanded closed, (2) there is an outflow from the relay when it is commanded open. In all other cases, the function is considered to be operating normally. Similar models are shown for condition electrical energy – inverter and sense electrical energy – voltage sensor function-to-component mappings.

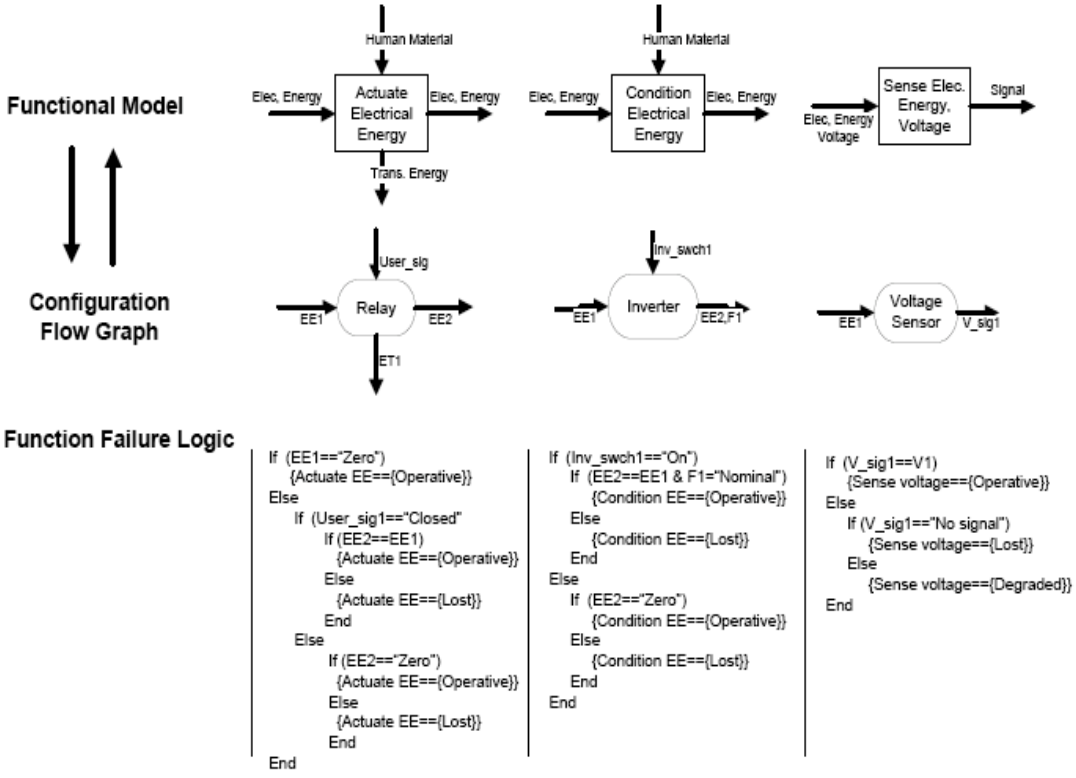


Figure 15. **Function Failure Logic relates the behavioral model to specific functional health.**

4.5.2 EPS Testbed Alternative System Architectures

The baseline design, shown in Figure 3, represents an early stage design implementation of the three top-level functions that the EPS testbed is required to perform. In the baseline system, the normal operating condition assumes that all three loads receive and operate a nominal voltage and current. Any change in load or power supply and distribution will have some effect on the system; in other words, the initial design is not designed to be fault tolerant. However, the same set of functional design requirements can be met by designing different system architectures representing fault tolerant behavior by employing different levels of redundancy and reconfigurability. In addition, the sensor allocation related to the integrated health management functionality can be made a number of different ways.

In this case study, we analyze the basic baseline design and compare that to three alternative conceptual system architectures. These alternative designs are developed from the same black-box representation ("provide power") and top-level functions ("supply power", "distribute power", and "operate loads") and demonstrate different levels of risk mitigation in different functional areas of the system. The functional models and configuration flow graphs for each design alternative can be found in the Appendix. The first modification of the baseline design includes a redundant power supply such that, if no electrical power is coming from the primary power source, a secondary source can supply the required power. The power to operate the loads comes from one power source or the other, that is, the system is not designed to take partial power from both. The second modification of the baseline design is an identical system with redundant loads configuration. This system would operate from a single power source and is designed to operate all six attached loads concurrently in a nominal state. Finally, the third modification of the baseline design incorporates the redundancies of the previous two modifications. This system is designed to operate six loads concurrently from either the primary or secondary power source.

4.6 FFR Method Applied to the Electrical Power System Design

Three alternative system design architectures (shown in Appendix) and the baseline design of Figure 13 will be used in the remainder of this chapter to demonstrate the application of the FFR methodology to the design of electrical power system architectures by following the four-step process described in Section 3.

4.6.1 System Models for the EPS

This step is summarized in Section 4.1 for the baseline design. The modeling approach is built upon a modeling environment, which uses modular, reusable function-component-behavior models so that each alternative design can also be modeled with minimal effort.

4.6.2 Determination of Functional Criticality Ratings

The function criticality ratings (FCR) for the sub-functions of the baseline design (see Figure 13) and the alternative designs (see Appendix A) are estimated using the process summarized in Section 3.2. Accordingly, the overall functional criticality is projected onto the three highest-level functions (i.e., supply power, distribute power, and operate loads) by assuming an equal critically distribution for all designs. (This translates into an FCR value of 0.333 for each of the three top-level functions). Since different design alternatives implement these top-level functions using a different level of functional decomposition, the FCR values vary from this level on when projected onto the elemental sub-functions. For example, the baseline design of Figure 13 implements the "supply power" top-level function by decomposing it into six sub-functions (shown with the module box labeled "power supply" in Figure 13), whereas the alternative designs implement the same functionality by using twelve, six, and twelve sub-functions respectively. Therefore, assuming equal criticality again, the elemental sub-functions decomposed from "supply power" function are assigned an equal FCR value of 0.0556 (1/6th of 0.333) for the baseline design and alternative design #2, and an FCR value of 0.0278 (1/12th of 0.333) for alternative designs #1 and #3. Similarly, each lower level sub-function is assigned an FCR value based on the decomposition of higher-level functions. The higher the FCR value of a function, the more valuable is maintaining that functionality during system operations. For example, for the baseline design, loosing the "actuate electrical energy" sub-function decomposed from the "distribute power" top-level function carries a higher risk (0.0333) compared to the same sub-function in the "operate loads" function (0.0238). The final function criticality ratings for the baseline electrical power system and the three alternative designs are summarized as in Table 4.

Table 4. **Distribution of FCR values for the baseline EPS design and the three alternative designs.**

Function	FCR	Baseline Sub-function FCRs	Design 1 Sub-function FCRs	Design 2 Sub-function FCRs	Design 3 Sub-function FCRs
Supply Power	0.3333	0.0555	0.0277	0.0555	0.0277
Distribute Power	0.3333	0.0333	0.0238	0.0185	0.0138
Operate Loads	0.3333	0.0238	0.0238	0.0118	0.0118

4.6.3 Computing the Functional Failure Impact for the EPS Scenarios

To evaluate and compare the alternative conceptual system architectures, 30 critical fault scenarios are identified and summarized in Table 5. These scenarios are chosen based on the operating characteristics of the system and discussions with its operators as well as the researchers testing diagnostic algorithms using this system. All thirty scenarios are run for each design through the FFIP behavioral simulator and reasoner. All component failures that are simulated are applicable to the baseline design as well as each of the three alternative designs. That is, if a relay is simulated as failed for the baseline design, the corresponding failure can be initiated in the alternative designs as well. (The only exception to this is scenario #25, which involves the interaction between two batteries, which cannot be simulated for the baseline design and modified design #2. In this scenario modified design 1 is taken as the baseline.) In Table 5, the fault type that is analyzed is indicated in the second column, whereas the third and the fourth columns list the component type and the functional location of the injected faults respectively. Collectively, the 30 scenarios capture all distinct faulty behaviors of components compromising the electrical power system. Moreover, it includes critical multiple fault scenarios (Scenario #24 - #30), as the simulation and the framework supports the analysis of multiple failures that affects the system in parallel.

***Table 5*. Summary of scenarios selected as a basis of comparison.**

	Scenario #	Fault type	Component type	Functional Location
Single Fault	1	Drift Up	Relay Position Sensor	Distribute Power
	2	Drift Up	Current Sensor	Distribute Power
	3	Drift Down	Voltage Sensor	Distribute Power
	4	Drift Down	Frequency Sensor	Operate Loads
	5	Stuck at Low	Relay Position Sensor	Supply Power
	6	Stuck at Low	Current Sensor	Supply Power
	7	Stuck at High	Voltage Sensor	Supply Power
	8	Stuck at High	Frequency Sensor	Operate Loads
	9	No Signal	Relay Position Sensor	Distribute Power
	10	No Signal	Current Sensor	Operate Loads
	11	No Signal	Voltage Sensor	Operate Loads
	12	No Signal	Frequency Sensor	Operate Loads
	13	Failed Off	Fan Load	Operate Loads
	14	Underspeed	Fan Load	Operate Loads
	15	Overspeed	Fan Load	Operate Loads
	16	Blocked Flow	Pump Load	Operate Loads
	17	Failed Off	Light Load	Operate Loads
	18	Failed Off	Inverter	Operate Loads
	19	Level Loss	Battery	Supply Power
	20	Terminal Short	Battery	Supply Power
	21	Stuck Open	Relay	Distribute Power
	22	Stuck Closed	Relay	Supply Power
	23	Failed Open	Circuit Breaker	Operate Loads
Multiple Fault	24	Blocked Flow, No Trip	Pump, CB	Distribute, Operate
	25	Failed Closed, Terminal Short	Relay, Battery	Supply, Distribute
	26	Overspeed, Failed Off	Fan Load, Light Load	Operate Loads
	27	Blocked Flow, Drift Up	Pump, Current Sensor	Distribute, Operate
	28	No Trip, Blocked Flow	Breaker, Pump	Distribute, Operate
	29	Stuck Open, Stuck Open	Relay Sensor, Relay	Distribute Power
	30	Drift Up, Stuck Low	Voltage, Current Sensor	Distribute Power

Figure 16 shows the current simulation environment. To start the simulation, the modes of individual components (nodes) in the CFG are initialized along with the values of system state variables associated with input flows (arcs). All 30 scenarios are run for each design by injecting “fault events” as summarized by Table 5. Each time step propagates the values of certain state variables depending on the mode of components, the behavioral models in that particular mode, and the defined component constraint relations. Following this approach, the simulation may be run over a certain number of time steps, or until the system reaches a prescribed end state.

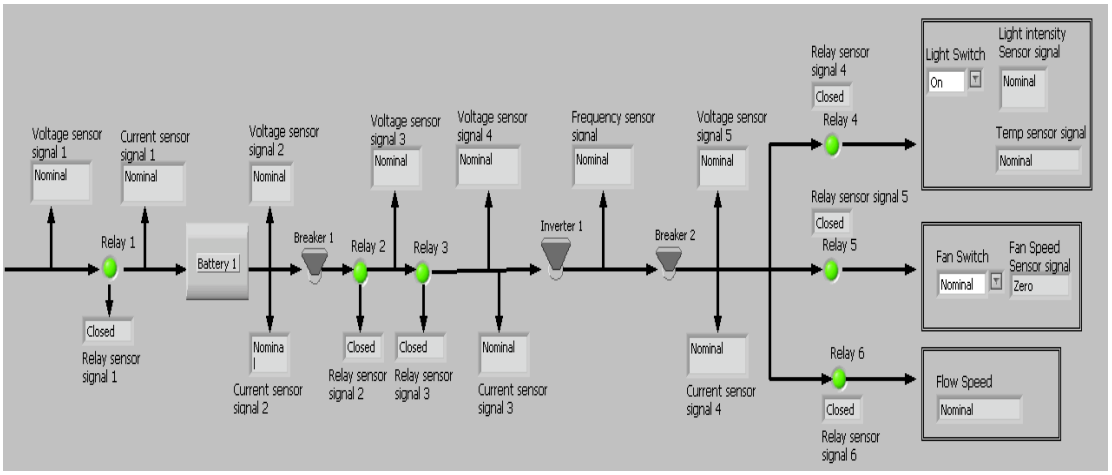


Figure 16. **LabView simulation environment shown for the electrical power system example.**

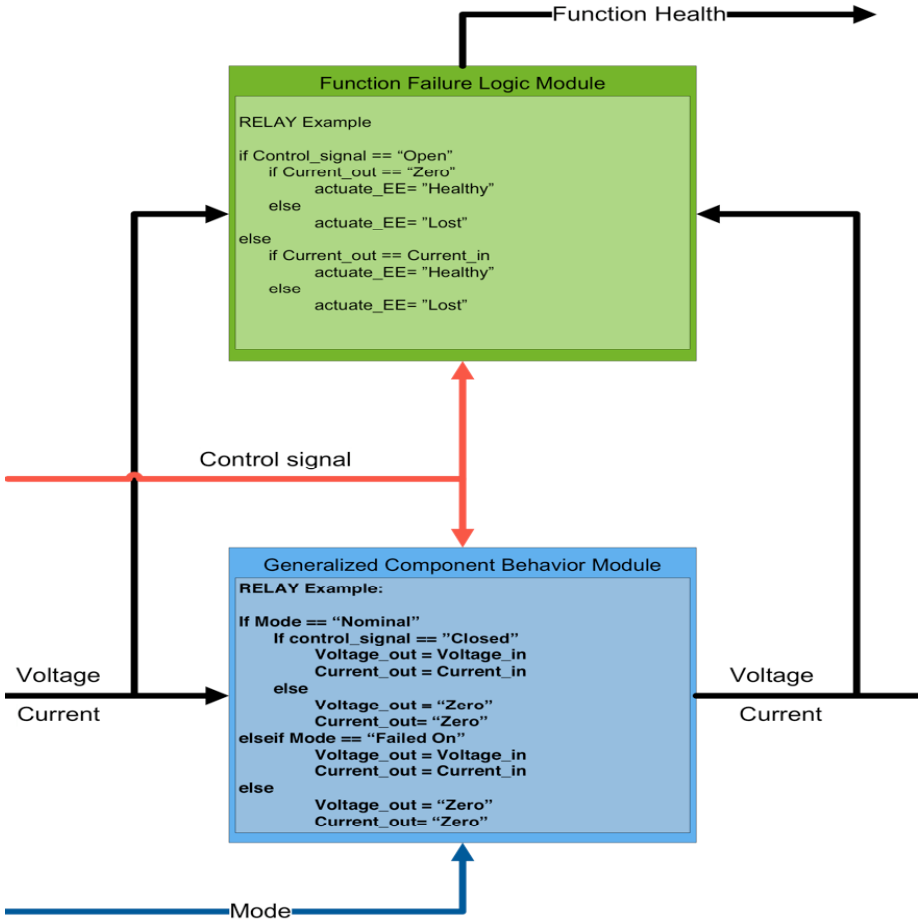


Figure 17. **The implementation of function failure logic.**

Using the simulation environment, component behavioral models and the corresponding function failure logic (FFL rules) are created as small, enclosed programs called virtual instruments. The virtual instruments are linked together by piping which carry the system variables, functional health and component modes as strings. This implementation is shown in Figure 17 for a relay component.

As described earlier, during the simulation, the functional state of each function is assessed through the FFL reasoner. This is illustrated in Figure 18 where the state of each function is shown at the end of a critical scenario for the baseline design. For this particular scenario, two functions are classified as "degraded", whereas all remaining functions are determined to be "operating". This step is also repeated for the baseline design and all alternative designs.

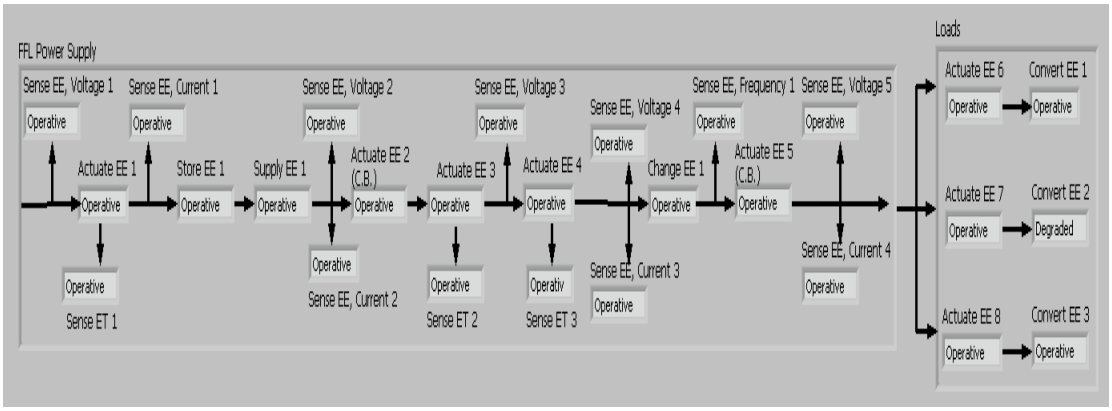


Figure 18. **The GUI for functional failure reasoning.**

Finally, the Functional Failure Impact of the selected scenarios is calculated using the functional failure reasoning employed by the FFIP FFL reasoner. In this step, first the functional failure impact of the eight critical scenarios is computed on the baseline design using Eqn. (1). This is followed by the calculation of the functional failure impact values for the alternative designs for the same set of scenarios. To illustrate this process, the impact of the two previously identified scenarios (Scenarios #21, and #28 in Table 5) on the baseline design are detailed next:

Scenario #21: In this scenario, the objective is to power all the loads in the system. To achieve that, the system needs to be configured by closing all associated relays and circuit breakers on the path from the battery to the loads. However, a relay failure is injected to the simulation at component "relay 2" in the form of "stuck-open", meaning that the relay cannot be commanded closed. This results in no power supply to the rest of the system components that are located downstream of "relay 2". Functionally, this means that the "actuate EE" function provided by "relay 2" is *lost* and all functions (except sensing functions) realized by the components downstream of "relay 2" are *lost recoverable*.

Scenario #28: This critical fault scenario begins with an initial state of all loads operating in a nominal state. Then, the "circuit breaker 2" fails in "no trip" mode – meaning that it will not trip as designed if its current level exceeds the threshold that triggers a trip. When this failure first happens, there is no observable functional level effect because the breaker is designed to be operating in a closed position at the time (in other words it is still performing its function). Later, another failure, in the form of "blocked flow" is injected into the component "pump". This failure causes a high current draw to the pump, which requires the breaker to trip. However, due to the "no trip" failure the breaker fails to trip, which causes less-than-nominal power to the light and fan loads. Functionally, this means that the "actuate electrical energy" function of the circuit breaker is *lost* and the "convert electrical energy" functions of all the three loads are *degraded*.

Using these results, the functional failure impact of these two scenarios are calculated to be:

FFI Baseline– Scenario #21: 0.3571

FFI Baseline– Scenario #28: 0.2380

4.6.4 Computing RIR for the EPS Alternative System Architectures

After the functional failure impact of the selected scenarios is established on the baseline and alternative designs, the reduction in risk (RIR) is computed for each alternative design (and each scenario) using Eqn. (2). This is shown in Table 6 for the two scenarios detailed in the previous section.

Table 6. **Function Failure Impact (FFI) and the subsequent Reduction In Risk (RIR).**

	Function Failure Impact Summary						
Scenario	Baseline	Alternative Design 1	RIR	Alternative Design 2	RIR	Alternative Design 3	RIR
21	0.3571	0.30949	13.33%	0.1878	47.41%	0.0555	84.44%
28	0.2380	0.2380	0.00%	0.119	50.00%	0.119	50.00%

For Scenario #21, the RIR values for the three alternative designs 1-3 are 13.33%, 47.41%, and 84.44% respectively. For the first alternative design, the redundant power supply configuration has marginal effect on the RIR value. This is expected because the faulty relay is downstream of the power supply units and having redundant batteries does not mitigate the effects of the relay failure. For the second alternative design, the RIR value improves. In this redundant load configuration, only one redundant path (from the faulty relay to the loads downstream) is functionally affected by the fault, and the system can be reconfigured to provide power to the second load bank. Finally, the RIR value is the best for the third alternative design, which employs an architecture including both a redundant power supply and a redundant load configuration. In this case, only a single function (the function provided by the faulty relay) is affected by the fault with no impact on the remaining functions. This is because the faulty relay can be bypassed by reconfiguring the system such that the redundant battery can provide power to either load bank.

For Scenario #28, the RIR values for the three alternative designs 1-3 are 0.00%, 50.00%, and 50.00% respectively. For the first alternative design, the redundant power supply configuration has no effect on the RIR value. This is expected because the

faulty components are both located at the “operate loads” module downstream of the batteries. For the second alternative design, the impact is similar to the baseline design where the “actuate electrical energy” function provided by the faulty circuit breaker and the “convert electrical energy” functions provided by the three loads are affected. However, due to the redundant load configuration, the redundant path from the power supply units to the second load bank remains intact and is functionally affected resulting in an overall RIR value of 50.00%. Finally, for the third alternative design, the RIR value is the same as the second alternative design. Here, the increased redundancy provided by the redundant power supply has no impact on mitigating the effects of the pump and circuit breaker faults.

4.7 Discussion of Results

As described earlier, the FFR methodology is implemented using thirty fault scenarios on three alternative designs. The reduction in risk (RIR) values for the three alternative designs are tabulated in Table 7 for these scenarios. Notable observations from the results are as follows:

1. The FFI values can be used as a *quantitative measure* of risk for each scenario. For example, the FFI results for the baseline design indicate Scenarios #20 (terminal short for battery), #21 (stuck open for relay), and #29 (relay sensor and a relay) to have the highest negative impact on the overall functionality of the system. This type of analysis helps designers formalize the information traditionally captured by an FMEA. These results can be used to identify risk-sensitive areas of the design and incorporated into design decision-making in order to develop necessary safeguards (such as redundancy, monitoring points, etc.) for the system.

2. Although the reduction in risk (RIR) values generally improves with increased redundancy, this is not the case for all fault types and locations. As was shown in Scenario #28, the increased redundancy provided by the redundant power supply in the third alternative design has no mitigation effect for the faults in

the “operate loads” module of the system. Moreover, in certain cases increased redundancy may reversely impact risk mitigation. This can be seen in Scenarios #15, #16, and #25 which all have negative RIR values for some of the alternative designs analyzed. Such results can be used for gaining insight regarding component arrangement decisions. Consider Scenario #15 as an example. In this scenario, a fan failure causes a circuit breaker to trip in the system (at one location for the baseline design and at two separate locations for the first alternative design.) For this particular case, the FFR analysis provides insight that if the redundant circuit breaker in the alternative design were to be located after the power splitting, the RIR value of the alternative design would not have been negative.

3. Although the previous two points illustrate how individual scenario results can be utilized to make certain design decision, the foremost benefit of the FFR methodology becomes evident when the combined effect of all critical scenarios are considered. This can be visualized by comparing the average RIR values (16.89%, 42.59%, and %53.53 – averaged over 30 scenarios) to the level of redundancy (21.62%, 39.58%, and 53.22% - computed based on the number of components) for the three alternative designs. The percentages show that – assuming equal likelihood of occurrence for all scenarios - the second design is the best alternative to the baseline design. This determination can be made by comparing the *RIR(%)/Redundancy(%)* values of the three alternative designs. This measure is analogous to the price/performance ratio, in that it indicates the level of risk mitigation that can be achieved by investing into a certain level of architectural changes (redundancy in this particular case) in a design. Using such combined results, designers can determine the level of risk mitigation that can be achieved by different system architectures and choose among competing design alternatives.

***Table 7*. Function Failure Impact (FFI) and the subsequent Reduction In Risk (RIR) values.**

		BASELINE	*ALTERNATIVE DESIGN 1*		*ALTERNATIVE DESIGN 2*		*ALTERNATIVE DESIGN 3*	
	Scenario #	**FFI**	**FFI**	**RIR**	**FFI**	**RIR**	**FFI**	**RIR**
Single Fault	1	0.133	0.095	28.57%	0.074	44.44%	0.056	58.33%
	2	0.133	0.095	28.57%	0.074	44.44%	0.056	58.33%
	3	0.133	0.095	28.57%	0.074	44.44%	0.056	58.33%
	4	0.095	0.095	0.00%	0.048	50.00%	0.048	50.00%
	5	0.222	0.111	50.00%	0.222	0.00%	0.111	50.00%
	6	0.222	0.111	50.00%	0.222	0.00%	0.111	50.00%
	7	0.222	0.111	50.00%	0.222	0.00%	0.111	50.00%
	8	0.095	0.095	0.00%	0.048	50.00%	0.048	50.00%
	9	0.133	0.095	28.57%	0.074	44.44%	0.056	58.33%
	10	0.095	0.095	0.00%	0.048	50.00%	0.048	50.00%
	11	0.095	0.095	0.00%	0.048	50.00%	0.048	50.00%
	12	0.095	0.095	0.00%	0.048	50.00%	0.048	50.00%
	13	0.095	0.095	0.00%	0.048	50.00%	0.048	50.00%
	14	0.095	0.095	0.00%	0.024	75.00%	0.024	75.00%
	15	0.257	0.286	-11.11%	0.078	69.86%	0.058	77.52%
	16	0.257	0.286	-11.11%	0.078	69.86%	0.058	77.52%
	17	0.095	0.095	0.00%	0.048	50.00%	0.048	50.00%
	18	0.262	0.262	0.00%	0.131	50.00%	0.131	50.00%
	19	0.111	0.056	50.05%	0.111	0.00%	0.056	50.00%
	20	0.513	0.135	73.68%	0.222	56.65%	0.139	72.90%
	21	0.357	0.309	13.33%	0.188	47.41%	0.056	84.44%
	22	0.222	0.111	50.00%	0.222	0.00%	0.111	50.00%
	23	0.238	0.238	0.00%	0.119	50.00%	0.119	50.00%
Multiple Fault	24	0.238	0.238	0.00%	0.119	50.01%	0.119	50.01%
	25	N/A	0.556		N/A		0.720	-29.64%
	26	0.143	0.143	0.00%	0.071	50.00%	0.071	50.00%
	27	0.300	0.262	12.70%	0.157	47.53%	0.155	48.41%
	28	0.238	0.238	0.00%	0.119	50.01%	0.119	50.01%
	29	0.490	0.405	17.48%	0.262	46.60%	0.206	57.93%
	30	0.267	0.190	28.57%	0.148	44.44%	0.111	58.33%

4.8 Summary – Necessity for Potential Failure Paths

In this chapter, a new methodology was introduced that can be used during early design of complex systems. The proposed functional failure reasoning (FFR) approach is based on the notion that a failure happens when a functional element in the system does not perform its intended task. Risk is defined depending on the role of functionality in accomplishing designed tasks.

A simulation-based failure analysis tool is used to analyze functional failures and their impact on overall system functionality. The analysis results are then integrated into a functional failure impact analysis framework that relates the impact of functional failures and their propagation to decision making in order to guide system level architectural design decisions. With the help of the proposed methodology, a multitude of failure scenarios can be quickly analyzed to determine the effects of decisions on overall system risk. Using this methodology, design teams can systematically explore risks and vulnerabilities during early, functional stage of system development prior to

the selection of specific components. Thus, the proposed method offers opportunities for significant reduction in cost, and increase in system safety and reliability by enabling early development of preventive measures that can effectively and efficiently guard against system failures.

In the process of applying the method presented in this chapter several insights were made into the limits of this approach. For ADAPT system represented an excellent opportunity to apply the FFIP framework to a system that was originally designed from a functional perspective. FFIP had not yet been applied to an electrical system prior to this stage of the research. One observation from modeling the behavior of components was that the electrical energy flows indicated in the system representation did not completely reflect actual system behavior. For example, the configuration and functional models portray electrical energy leaving the battery, that energy is measured by current (and other) sensors, and finally operate loads. This linear path works well for describing the designed operating behavior. However, when a load fails and draws high current there is no path that reflects the backward draw from the battery in the functional and configuration system representation. This path was added to the simulation to account for actual behavior. On further analysis, it was observed that numerous actual EMS flow paths are not accounted for in the designed system models. Faults that were directly related to these paths would not be identified by the current method. This stage of the research thus laid the foundation for the concept of failures propagating along paths not identified in the system representation. Further research revealed the implication that this limitation has not only for the FFIP method for failure analysis methods in general. To directly meet this limitation the Flow State Logic method was developed to augment the developing FFIP method and is presented in the following chapters.

5 Flow State Logic Methodology

5.1 Summary:

The following Flow State Logic (FSL) methodology is presented as an extension and modification of the Function Failure Identification and Propagation (FFIP) framework. This integrated approach shows how reasoning on flow state can provide system risk and failure propagation information. This section concludes with a brief summary on the difference between the presented method and the previous FFIP work. The method presented here is a means for system risk evaluation that incorporates failures that propagate along failure flow paths not identified in the original design. The method is summarized in the following seven steps.

System representation

1. Create Functional Model with weighted attributes
2. Create at least one component configuration flow graph

System simulation

3. Create propagation based behavioral models for components
4. Create Function Failure Reasoner
5. Create Flow State Reasoner

Risk analysis and mitigation

6. Simulate fault scenarios through component mode changes and new flow faults
7. Evaluate impact of scenario possibly redesign from component configuration graph

5.2 Approach

Failure events can often lead to unanticipated Energy, Material, and Signal (EMS) flows in a system. If failure propagation is assumed to follow EMS flow paths then a complete failure analysis of a design must include potential flows. From a failure standpoint, any flow between components and from a component is possible. Also any flow from the environment to the component is also possible. It is therefore necessary to distinguish between designed flows and non-designed, or potential flows. Figure 19 illustrates the difference between designed flows and potential flows for a simple system.

Non-designed flows are the cause and/or effect of certain failure events. In Figure 19 the arcs representing potential flows in the system are simplified to represent multiple EMS flow types.

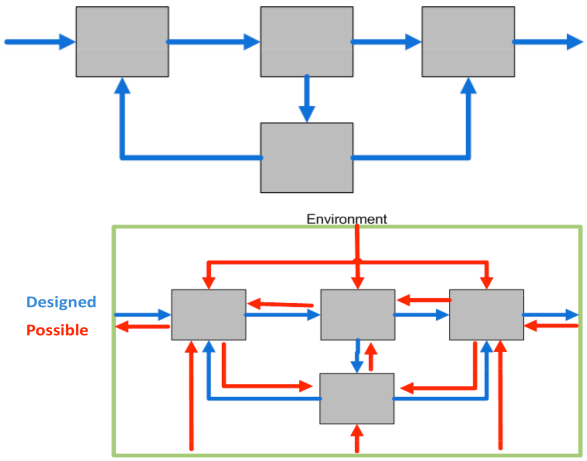


Figure 19. **Flows in a functional model of a designed system (top) vs. possible flows a system may actually experience (bottom)**

5.3 Method Steps

5.3.1 Create Functional Model

System functions are derived directly from the system or customer requirements and are represented using function structures [30, 37, 56, 63]. Functions structures are

composed of all the functions and flows of a system where functions are represented as verb-noun pairs (e.g., transfer gas, mix liquid, etc.). The flows are broken down into three categories: energy, material, and signal. A functional taxonomy has been developed to describe the function flow interactions for electro-mechanical and fluid systems [37, 57]. With the addition of software specific language for function and data flow relationships used in object oriented programming [2], a generic taxonomy of functions is developed. Using this taxonomy the high level functions are expanded to a useful level of fidelity. At this point a preliminary weighting factor is assigned as an attribute to each function within the system. These weighting factors sum to unity and are called Function Criticality Factors (FCR's). FCR's serve as means to link high-level system requirements to specific functional implementation [48].

5.3.2 Component Configuration Model

The system architecture design is captured using configuration flow graphs (CFGs) [60]. A CFG follows the functional topology of a system and maps functions to specific component implementations. In a CFG, nodes of the graph represent system components, whereas arcs represent energy, material or signal flows between them. For flow naming, the Functional Basis terminology is adopted, while the components of the graph are named using a taxonomy of standard components [49]. The component types in a CFG can be thought of as generic abstractions of common component concepts. In addition to the components an environment block is included which serves as a boundary for the analysis and source for system flows.

5.3.3. Propagation-Based Behavioral Models

Previous methods of defining component behavior identify the relationship between known input and output flows based on component mode [1, 48]. To be flexible enough to analyze failure scenarios with new unknown flow types the behavioral models must be approached differently. This method uses a systematic approach to flow propagation to create component behavioral models. This is done by establishing a relationship between the component behavior mode and the propagation

characteristics of a flow. The types of flows used in this method are the secondary level of flow as specified by the Functional Basis [37, 57] and summarized in Table 8.

Table 8. **Flow types considered for behavioral model propagation.**

Energy	Material	Signal
Generic	Human	Status
Human	Gas	Control
Acoustic	Liquid	
Biological	Solid	
Chemical	Plasma	
Electrical	Mixture	
Electromagnetic		

For a component, the behavioral model is created by defining the relationship between designed input and output EMS flows based on component mode. Then for each type of potential flow that is considered, a designer specifies the critical level at which a component mode would change. If a critical level exists, then the component mode change is specified. Finally, some components absorb flows and do not propagate to other system components. Therefore, the Boolean options of propagation or no propagation are specified for each potential flow type.

For example, a generic electric motor is nominally designed to have an input flow of electrical energy and an output flow of rotational energy. In developing the behavioral model of this component the potential flow of liquid material to this component should be considered. Based on designer knowledge, the behavioral model represents that the mode of this component will change to a failed state with a low level of liquid material. Also this component has no capacity to store a liquid material so the behavioral model would reflect that liquid material would propagate to other components. This systematic approach can be taken with each type of flow to determine component behavior to potential flows.

Both the designed behavior and the behavior associated with new flows the designer must answer the following three questions for each of the EMS flows identified in Table 8:

1) What qualitative level (zero, low, high) for each flow is necessary to change the mode of the component? (This is the critical level.)
2) Considering each flow individually, will that flow be propagated to nominally connected components? (Components are nominally connected through designed EMS flows.)
3) How will each flow at its critical level affect component mode?

The summary of answers to these three questions are used to derive the behavior model for each component as shown in Table 9.

***Table 9*. Example of propagation behavior definition for a generic liquid valve component.**

Flow	Critical Level	Propagates?	Mode change
Designed Flows			
Liquid Material	None	Yes	None
Control Signal	On/Off	No	On/Off
Potential Flows			
Generic E.	Low	Yes	Stuck closed
Human E.	Low	Yes	Stuck closed
Acoustic E.	High	Yes	Stuck closed
Biological E.	High	Yes	Stuck closed
Chemical E.	High	Yes	Stuck closed
Electrical E.	High	Yes	Stuck closed
Electromagnetic E.	Low	Yes	Stuck closed
Human Mat.	Low	Yes	Leak
Gas Mat.	High	Yes	Stuck closed
Liquid Mat.	High	Yes	Stuck closed
Solid Mat.	Low	Yes	Leak
Plasma Mat.	Low	Yes	Leak

5.3.4 Function Failure Logic Reasoner

The function-failure logic (FFL) module of the FFIP framework uses its reasoner to determine the state of each system function. At discrete time steps, the reasoner evaluates the state of a function by the comparing the actual and expected input and output flows of the behavioral models. The FFL reasoner translates the dynamics of

the system into functional failure identifiers and facilitates the assessment of potential functional failures and resulting fault propagation paths in the function domain.

Note that, FFL allows the assessment of the operability of a function to be made based on the values of the input and output state variables of the CFG that corresponds to the component by which the function is realized. Therefore, capturing the mapping between the functional model (function) and the configuration flow graph (behavior) is fundamental to the employment of the function failure logic. The reasoner uses a set of form-independent system function models that describe conditions under which functions deviate from their intended operation [1]. Accordingly, system functions are classified as 'operating', 'degraded', 'lost recoverably' or 'lost', defined as follows:

Operating: Function operates on a flow as designed

Degraded: Function operates on a flow not as designed

Lost Recoverably: Function has no flow to operate on because of a different functional failure

Lost: Function does not operate on flow

5.3.5 Flow State Logic

The Function Failure Logic (FFL) reasoner in the FFIP framework captures the health of functions embodied by components but not the state of flows between components. To map failure propagation along new system EMS flows the Flow State Logic (FSL) reasoner has been developed. The FFL reasoner identifies the health of functions embodied by components. The two reasoner roles are differentiated in Figure 20. The logic of both FFL and FSL operate on the inputs and outputs of component behavioral models called ports. The ports are alphabetically labeled in Figure 20. The use of ports can be found in other literature which have outlined the general types of designed ports [64, 65]. In addition these earlier defined port types, an input and output port for potential flows is included in this methodology. The FFL and FSL reasoners read these port values to determine function health and flow state respectively. The FSL

provides state information for both the designed and potential flows in a system. Flow states are separated as one normal state representing the designed flow and three states representing non-nominal flow. These states are:

Normal – Flow is consistent with the original design

New Flow– Flow exists but was not designed to exist

No Flow – Flow does not exist but was designed to exist

Reversed Flow – All aspects except direction of flow are as designed

The state of "No Flow" was captured by the Function-Based Failure Propagation Method [43] as a function failure but is included here for completeness. By defining all flows that appear in a system in the above way during a critical event simulation it is possible to identify the failures that propagate along both designed and potential EMS flow paths. Figure 21 illustrates nominal and potential flow states as a result of specific system faults.

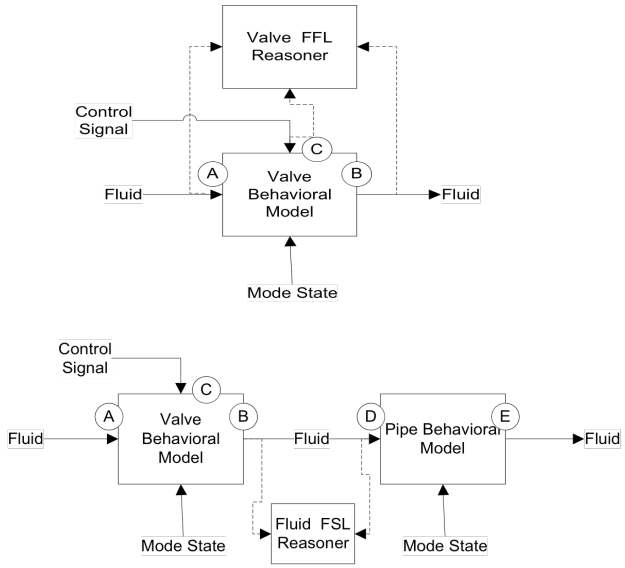


Figure 20. **FFL and Behavioral Model (BM) relationship (top) and FSL and BM relationship (bottom). Both read BM port values, here labeled alphabetically.**

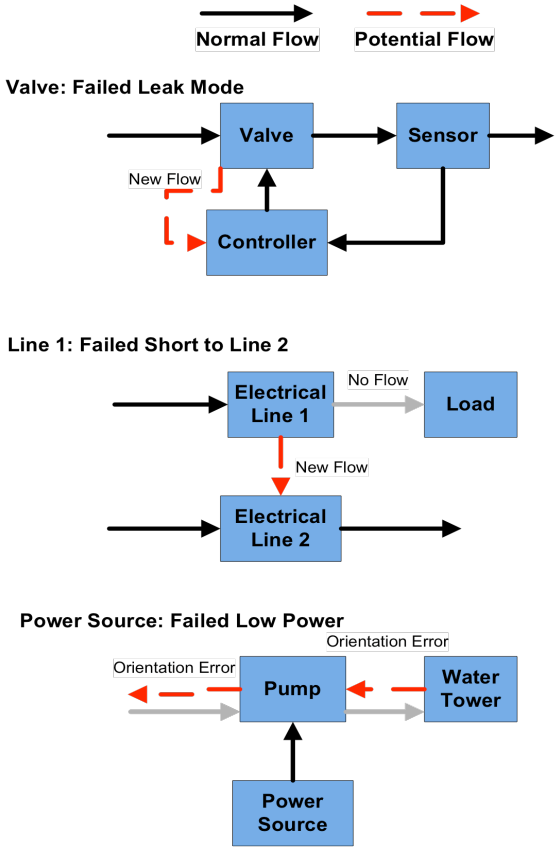


Figure 21. **Potential and designed flows in three example failure states.**

Figure 22 shows the logic used by the FSL for potential and designed flows.

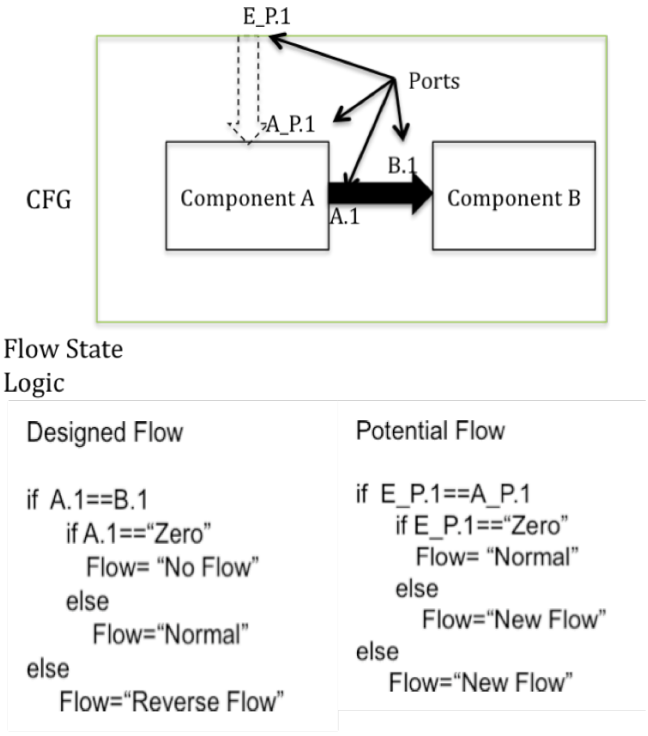


***Figure 22*. Example logic used in the Flow State Logic (FSL) Reasoner.**

5.3.6 Fault injection and Scenario Simulation

The component behavioral models, FFL and FSL modules are integrated to form a behavioral simulator that determines the system state under certain specified conditions for the critical scenarios. Critical scenarios are created by injecting faults into the simulation in the form of component mode changes and new system EMS flows. During the simulation, both the discrete component modes and the system state variables are tracked.

During conceptual design, the system state variables are not known quantitatively. To deal with this constraint, these continuous variables are discretized into a set of

qualitative values. For example, an electrical current variable may take on values from the set of {zero, low, nominal, high}. Similarly, a status signal variable indicating the position of a circuit breaker may have values of {open, closed}.

The scenario directly indicates the path of fault propagation in the system. The combined output of the FFL and FSL modules provides the complete functional state of the system as a result of the injected fault.

5.3.7 Evaluate the Impact of the Fault Scenario

The output of the FFL is the function state for each function in the system. A relative cost factor is assigned to each state. Detailed explanation of this step can be found in Chapter 4. An abbreviated version is presented here for clarity. When the cost factor of a functions state is multiplied by the FCR assigned when creating the functional model the direct impact to that function for that fault is quantified. In this way the Function Failure Impact (FFI) can be calculated for the whole system for each fault scenario as:

$$\textit{Functional Failure Impact: } FFI = \Sigma\, C_i \, x \, FCR_i \qquad (1)$$

The output of the FSL module is the state of designed and potential flows in the system as a result of a fault scenario. This is used to map the fault propagation of failure in the system on the functional model.

Based on the results of a series of critical scenarios the risk of a specific component implementation and configuration with respect to function is assessed. This information then guides redesign and design comparison.

5.4 Specific Advantage of Methodology

The FFIP framework is capable of providing function health, EMS flow values, and failure propagation mapping for specific failure scenarios. The FLS method presented here extends the scope of results that can be attained by FFIP through failure scenarios that contain new EMS flows. The new scenarios that can be evaluated with the

augmented FFIP method include failures that result from non-designed EMS flows entering the system and EMS flows that occur as a result of a component failure. Secondly, in the original FFIP method EMS flow values where either specified or reported based on the output of behavioral components. The FSL augmented FFIP method can identify flow state failures that occur when two or more behavioral models have conflicting values for an EMS flow. For example, this can occur with a failed mode of pipe. A blocked pipe would have zero output of liquid material and therefore would have zero input of liquid material. However, an upstream valve component could be operating nominally, resulting in a behavioral model defining the output of liquid material equal to the input. If these two components are connected with the same EMS flow it creates an inconsistency that can now be handled because of the FSL reasoner.

6 Proof of Concept Case Studies

6.1 Controlled Liquid-Fueled Rocket

To illustrate the effectiveness of this method to the field of complex system design an example system has been designed and analyzed with the presented methodology. The example system is of a liquid-fueled engine. The goal of this design is to combine fuel and oxidizer in a controlled fashion and provide thrust for a higher-level system. Liquid fueled rockets are used extensively for spacecraft applications. Physically this system stores a set amount of fuel and oxidizer in a liquid state. By controlling the combustion of these two fluids in a reaction chamber, thrust is provided for the larger system. To control and monitor the continuous chemical reaction a series of sensors and a software controller component must be included.

6.1.1 Functional Model and Criticality

The first step in the presented methodology is creating the functional model for the system. Clearly a degree of expert knowledge is necessary to create a design that is likely to be successful. Figure 23 illustrates the initial functional decomposition based on the system requirement with an equal distribution of function criticalities.

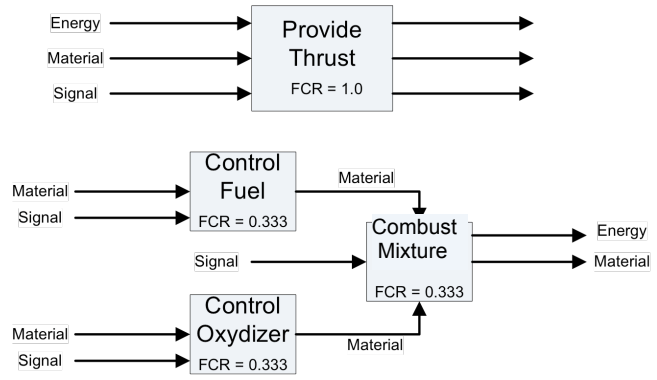


Figure 23. **Initial functional decomposition of liquid-fueled rocket engine.**

Figure 24 represents the functional model of the liquid fuel rocket and Table 10 shows how the FCR's are equally distributed. Note, that the sub-functions that satisfy multiple high-level functions, such as "Process Signal," have a higher FCR due to the additive factor from each high-level function.

6.1.2 Component Configuration

After the functional model is established a configuration flow graph (CFG) is created based on embodying functions into generic components. As can be seen from Figure 25, this design includes a software controller to regulate valves based on command signals. Included in the CFG is a block identified as the environment that this system operates within. This block represents an environmental behavioral model that can interact with components within the system through potential and designed EMS flows.

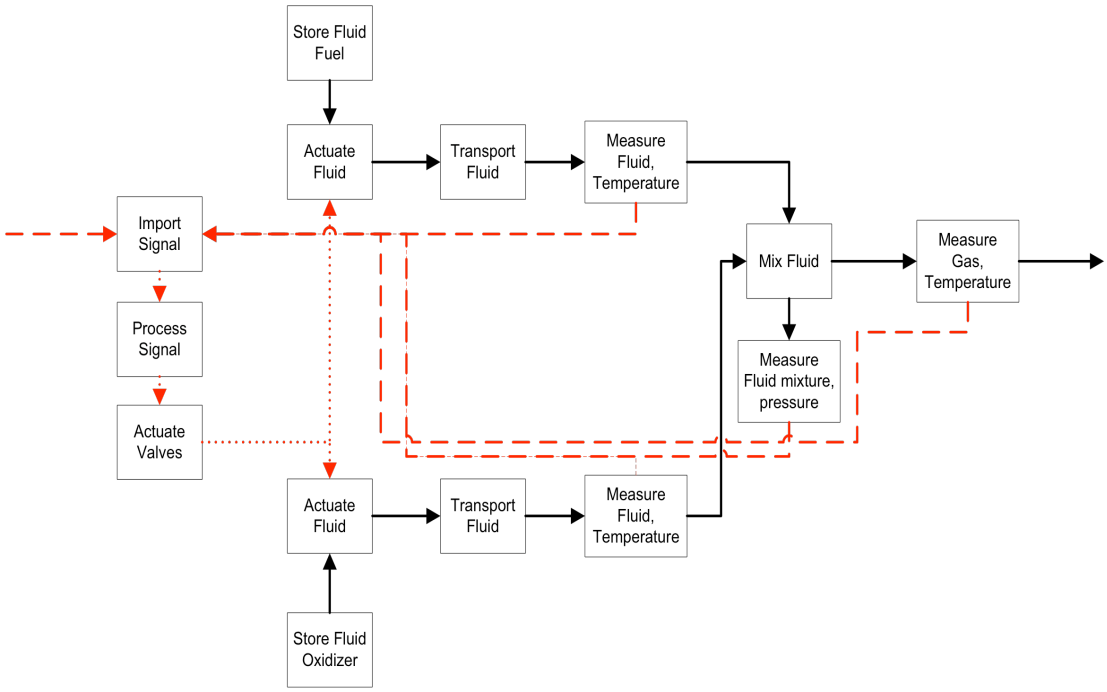


Figure 24. **Functional Model of a liquid-fueled rocket.**

Table 10. **Functional Criticality Rating (FCR) for of a liquid-fueled rocket modeled in Figure 24.**

Sub function	FCR
Store Fluid Fuel	0.047614286
Actuate Fluid Fuel	0.047614286
Transport Fluid Fuel	0.047614286
Measure Fluid Fuel	0.047614286
Store Fluid Oxidizer	0.047614286
Actuate Fluid Oxidizer	0.047614286
Transport Fluid Oxidizer	0.047614286
Measure Fluid Oxidizer	0.047614286
Combust mixture	0.05555
Measure mixture pressure	0.05555
Measure Mixture Temp	0.05555
Import Signal	0.150778571
Process Signal	0.150778571
Export Signal	0.150778571

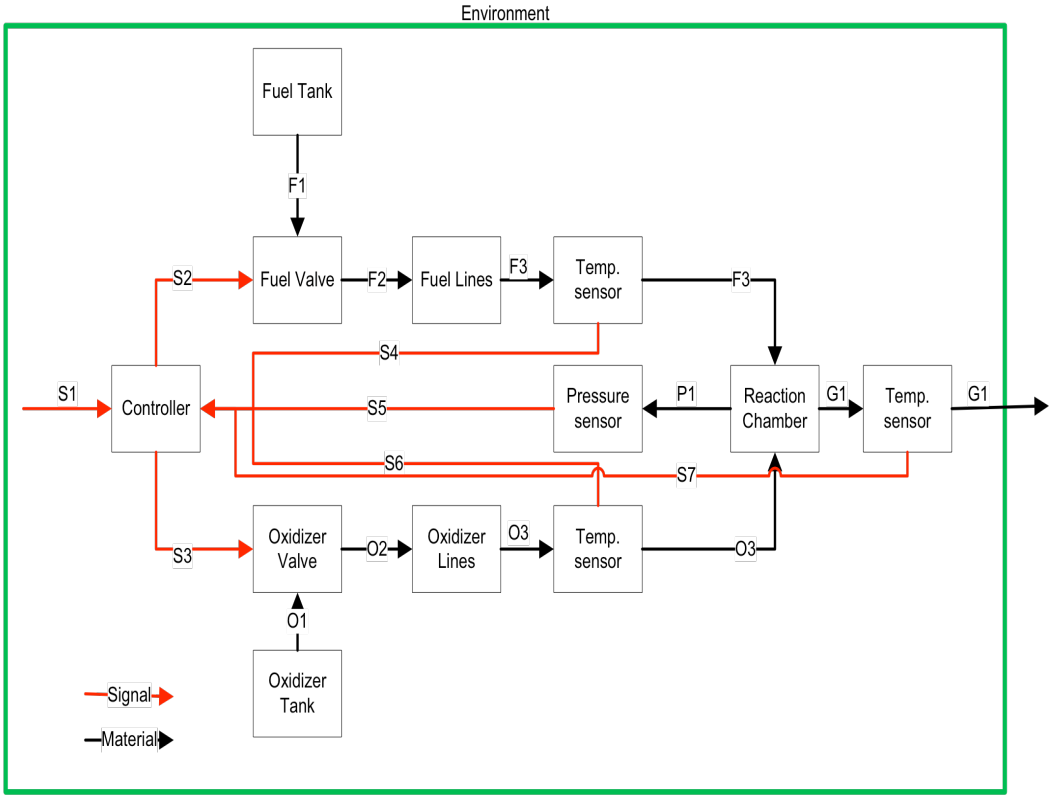


Figure 25. **Configuration Flow Graph of a simple liquid fuel jet.**

6.1.3 Simulation

For the third step of this method behavioral models were created using the LabView Software to implement the system simulation. For each component a table of propagation behavior was created to define the behavior model (see Tables 1A-7A in Appendix A). Integrated into this simulation is the FFL and FSL reasoners which evaluate function health and flow state respectively. Figure 26 illustrates the simulation environment and interface for the liquid-fueled rocket system.

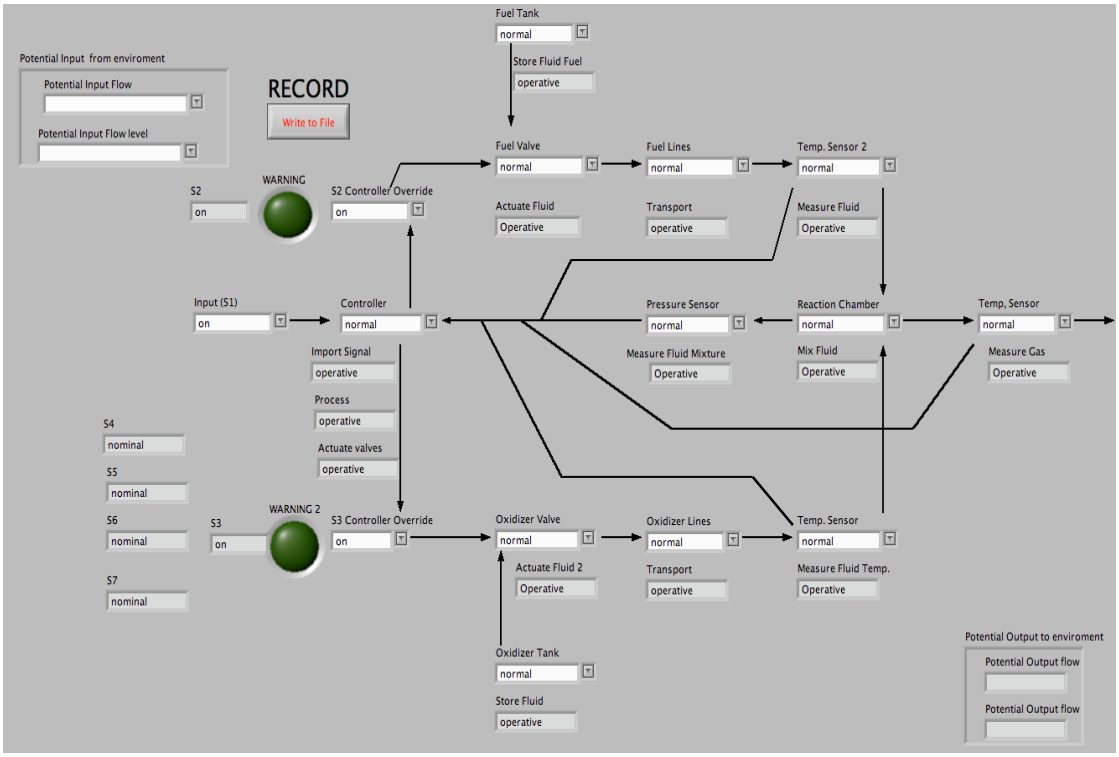


Figure 26. **Labview implementation of the FSL augmented FFIP framework for a liquid-fueled rocket engine.**

6.1.4 Fault Scenario Results

Initially a 40-entry catalog of single fault failures was created for this design based on the failed modes of each component using the original FFIP framework analysis. The catalog of fault descriptions is shown in Table 8A in Appendix A. These same failures

were applied to the FSL augmented framework to verify results. In addition a series of failure scenarios where EMS flows where injected and new flows resulting from leak failed modes where examined. These additional failure scenario descriptions are detailed in Table 9A in Appendix A.

The Function Failure Impact (FFI) for each fault scenario was calculated and the resulting fault propagation mapped to the functional model. Table 10A in Appendix A shows the FFI results for the original FFIP scenario sets and Table 11A in Appendix A expands these results to include the new failure scenarios analyzed by the FSL augmented FFIP framework.

7 Discussion of Flow State Logic Results

7.1 New Scenario Impact

The results of applying 40 single component fault scenarios was applied to an FFIP simulation of the liquid-fueled rocket engine design identified in Figures 24 and 25. Applying the same scenarios to an FSL augmented FFIP simulation was done to verify simulation results. Both simulations provided identical Function Failure Impact (FFI) ratings for the single component fault scenarios. With the addition of the FSL modules, 15 additional scenarios were identified as possible failures for this system. Some of these failures are similar to scenarios from the single fault set but with additional causes and effects that are outside of the scope of scenarios that can be analyzed with non-augmented FFIP simulation. These include liquid and gas material flows coming from components in leak modes. These new flows were simulated to affect other components in the system. Additionally, new environmental material and energy flows were simulated to affect system components. The FFI values resulting from applying the scenarios can be used as a comparison of the relative importance of scenarios. Table 11 shows how the single fault scenarios and the additional FSL scenarios rank with respect to functional impact (specifically, the resulting FFI for the scenarios). Other ranking methods could be used such as number of functions affected or based on designer expert opinion on impact. Table 11 demonstrates that the failure scenarios possible with the FSL augmented method can be significant and a reliability analysis of this system would be inadequate without taking into account failures propagating along non-designed EMS flows. The set of FSL scenarios presented here is not meant to be a definitive list of possibilities but rather a first step to demonstrate the value of implementing FSL in early design reliability.

Table 11. **Scenarios applied to FSL augmented FFIP simulation, ranked according to highest Function Failure Impact (FFI) results. Blue scenarios are the set of single component faults, green scenarios are a set of new scenarios capable of analysis with FSL.**

Scenario Description	Ranking
Pressure Sensor - No Signal	1
Reaction Temperature Sensor - No Signal	
Fuel Temperature Sensor - No Signal	2
Oxidizer Temperature Sensor - No Signal	
High Acoustic to Fuel Temp. Sensor	
Controller - No Power	3
Fuel Valve Leak to Controller	
High Solid Material to Controller	
Low Acoustic to Fuel Valve	4
Oxidizer Valve Leak to press sensor	5
Low Acoustic to Controller	
Fuel Line Leak to fuel temp sensor	6
Controller - Failed On	7
Oxidizer Valve - Stuck Open	8
Fuel Valve - Stuck Open	
Fuel Tank - Empty	9
Oxidizer Tank - Empty	
Pressure Sensor - Stuck High	10
Reaction Temperature Sensor - Stuck High	
Fuel Temperature Sensor - Stuck High	11
Oxidizer Temperature Sensor - Stuck High	
Fuel Tank - Pressurized	12
Fuel Tank - Leak	
Oxidizer Tank - Pressurized	
Oxidizer Tank - Leak	
Low Solid Matter to Valve, leak to controller	
High Human Material to Oxi Tank, Leak to Valve	

Scenario Description	Ranking
Fuel Temperature Sensor - Stuck Low	13
Oxidizer Temperature Sensor - Stuck Low	
Pressure Sensor - Stuck Low	
Reaction Temperature Sensor - Stuck Low	
Fuel Valve - Leak	
Fuel Valve - Stuck Closed	
Fuel Line - Leak	
Fuel Line - Blocked	
Fuel Temperature Sensor - Stuck Zero	
Reaction Chamber - Leak	
Reaction Temperature Sensor - Stuck Zero	
Oxidizer Valve - Leak	
Oxidizer Valve - Stuck Closed	
Oxidizer Line - Leak	
Oxidizer Line - Blocked	
Oxidizer Temperature Sensor - Stuck Zero	
Oxidizer Line Leak to Reaction Chamber	
Rxn Chamber Leak to Oxi line	
High Acoustic to Oxi Valve	
High Acoustic to Fuel Line, leak to Fuel valve	
Low Solid Matter to Line	
High Solid Matter to Chamber, leak to controller	
Fuel Temperature Sensor - Stuck Nom	14
Pressure Sensor - Stuck Nom	
Reaction Temperature Sensor - Stuck Nom	
Oxidizer Temperature Sensor - Stuck Nom	

7.2 Application to Prognostic Health Management

In the rocket engine case study the controller element represented three functions with the highest Functional Criticality Rating (FCR) based on the initial functional decomposition. This weighting can be seen in the resulting FFI values from the scenario simulations. The scenarios ranked 1-6 in Table 11 all either affect the controller by providing no signal to the controller or by direct adverse action against the controller. Without a signal to operate on the processing functionality of the controller is lost. The large group of scenarios ranked as 13 in Table 11 have the same FFI because the controller response to those scenarios is identical. This simple controller is designed to automatically close the fuel valves and oxidizer valves (thereby stopping the combustion) when any sensor data is irregular. The functional result is that "transport fluid" functions and the "combust mixture" functions are *Lost Recoverable* and the rest of the system reflects an accurate system state. This means that a fuel temperature sensor that is stuck at a *Low* value causes the controller to close the valves. Because the valves are closed the *Low* reading from the temperature sensor is then accurate, leading the Function Failure Logic (FFL) reasoner in the FFIP simulation to declare the "measure temperature" function as *Operative*.

Several insights can be used from this early analysis to aid PHM designers in the initial steps of controller design. For example, the singular response of shutting off the valves may be a safe approach but it may mask real faults in the system, as in the temperature sensor case above. To address the most critical scenarios identified in Table 11 the PHM system must be able to operate and determine a correct course of action when no signal is received from the sensors. While this last conclusion would likely by included in any thorough PHM development process, applying the simulation and evaluating the impact highlights the importance of flexible software.

7.3 Future work in Applying the FSL method

In determining how new EMS flows propagate from different components, two avenues of exploration are possible. The basis for this method is that EMS flows exist both as designed and as possibilities. Along this line, Monte Carlo testing of the effect of flow propagation between components might provide a breadth of failure impact. This might be considered a bottom-up approach to determining how new flows propagate. Alternatively, discrete event simulations of known system failures could be used to direct new flow propagation. This would be a top-down approach and would provide significant detail for each particular failure scenario.

An area that is only beginning to be addressed in the functional failure work is the timing of failure propagation [66]. So far failure simulations use discrete sequential events without respect to time intervals [67]. Future work must address the issue of timing to adequately analyze the electromechanical and software systems concurrently.

The Flow State Logic method presented here was used with a specific failure propagation analysis method, namely the Function Failure Identification and Propagation (FFIP) analysis framework [1]. Application and comparison of other methods represents an area of further research. Finally, to address the difficulty of behavioral model creation we recommend the creation of component libraries for specific design areas as has been useful with other methods [68].

7.4 Implications for the Field of Reliability Analysis During Conceptual Design

As shown in the first section of this chapter, the FFIP method without the FSL module would not have been capable of modeling numerous significant fault scenarios. This does not reflect a inadequacy of the FFIP as much as a limitation of the field of conceptual reliability analysis. Other methods that cannot process failures propagating along non-designed EMS flow paths would be hindered as well. For conceptual failure propagation analysis the functional modeling method is inadequate for system

representation. This is because the function flow relationships are predefined and are not flexible. For example, what a function like “store energy” does to a material or signal flow cannot be defined unless component information is considered. The FFIP method already supports this conclusion by the use of both a functional and a component configurational system representation.

Further, this limitation applies not only to functional modeling but to any system representation that does not consider non-designed EMS flow paths. FMECA methods have been used to evaluate the impact of component failures to the rest of the system. Designers may inadvertently limit the scope of effects to components and systems that are connected in some system architecture representation. Fault Tree methods are likewise limited when only connected components are accounted for as contributing causes of failures. Finally, historical data repositories can only provide relevant information on historical failures and provide little insight into the effect of new failure paths.

It is beyond the scope of this project to propose a method for augmenting every reliability method to evaluate new failure propagation paths. Instead, the limitation was uncovered through this research and a means of addressing it was applied to a specific method. This research may serve as an example for other methods to incorporate the concept of potential failure propagation.

7.5 Flow Sate Logic Summary

Most of the recent design stage reliability methods use functional modeling for system representation. For failure analysis it is assumed that failures propagate along energy, material, and signal (EMS) flow paths identified in the functional model. For a number of significant failures the original functional model of the designed system does not include EMS flows that exist as a cause or effect of a system failure. To address this issue a method of identifying flow state for potential flows is presented, namely, the Flow State Logic (FSL) methodology, and is integrated with prior work on failure propagation analysis. In conjunction with a thorough, propagation-focused behavioral

model of components, failures that propagate from unanticipated EMS flows were identified. An evaluation of a conceptual design for a liquid-fueled rocket engine with an existing reliability method, namely Function Failure Identification and Propagation (FFIP) framework, and that method augmented by the presented FSL method identifies the contribution of this work. Using the fundamental approach to failure of categorizing EMS flows in designed and potential states and the resulting FSL logic reasoner, the scope of the FFIP analysis is expanded. Multiple new failure scenarios are capable of being analyzed using the augmented method as well as failure propagation mapping for previously unrecognized failure states.

The impact of this is reflected in the use of this approach and modification of the FSL methodology to fit other conceptual risk analysis methods. By expanding the scope of applicable failures that a method can evaluate designers have a more accurate understanding of risk and reliability in the conceptual design phase for complex engineered systems.

8 Conclusions

8.1 Limitations and Continuing Issues of the Current Work

The Flow State Logic (FSL) method presented here highlighted the significant limitation of using the designed system representation for failure propagation. The FSL method was used to adapt the Function Failure Identification and Propagation (FFIP) framework to be capable of analyzing failures propagating along new flows. However, there are other limitations to the system representation used in the FFIP framework. First, subsystem designers tend to think in terms of components and not functions. To be a useable tool, this method should allow designers to either build models at the component or functional level without restrictions. Secondly, the functional representation at the heart of the FFIP method requires well-defined function flow relationships. Augmenting FFIP with FSL was limited to the component level as there seems to be no logical means for representing function/flow relationships for all flows. For example, “actuate fluid” presents a clear relationship between the function block and the flow but the function itself might cease to exist when a solid material flow is introduced.

In Chapter 4 it was shown how quantifying the FFIP method could provide designers with a decision-making tool. To illustrate this, a design for an electrical power system was created and analyzed using fault scenarios, comparing four designs with different levels of redundancy. The issue of design refinement becomes important when comparing designs for decision-making. Comparing a potential design with a more clearly defined behavioral model for components (indicating more implementation knowledge) than a less defined design would not be an equal comparison. For example, one design may fail because the flow value was “high” when only four flow levels are defined. Another, more refined design may not fail because more flow levels where defined and the flow value was within an acceptable range. In order to fairly compare designs they must be compared at the same refinement level. For this work

that was accomplished by using the same component models for different designs. This limits the scope of comparison to designs using same component types and same flow distinction levels. Future work will identify clear guidelines for comparing designs using different component types so that designs are compared at the same refinement level.

Related to the issue of refinement is the scalability of the presented analysis method. Complex systems are composed of multiple interconnected subsystems. The complexity of this analysis thus grows exponentially when scaled to the rest of the system. For this reason simulation and modeling approaches will use tools that have been successfully used in systems engineering to address scalability in the future work of this project. One example method is using the tool Model Center for wrapping subsystem simulations to form a system simulation.

The Function Failure Impact (FFI) values introduced in Chapter 4 present a continuing issue for identifying the real meaning of those values. For example, from total system effect perspective the FFI value from a scenario can be related to the overall system function state. The consequential cost factors thus become the measure of the FFI. If all functions in a system were at the same state as a result of a failure scenario the FFI would equal the consequence factor for that state. Generally, it is not expected that a failure would change the state of all functions in a system. Therefore, the FFI values for different scenarios and different designs are most relatable to one another. This approach requires a statistical approach to the impact and FFI values should be reported with corresponding averages for that design and standard deviations. In Chapter 4 the FFI values were related relative to a base level design with the concpt of reduction in risk. In Chapter 7 the FFI values were used in absolute terms and ranked highest to lowest. Future work will show that when System Health Management (SHM) is included into the analysis that the FFI value can be viewed as impact of the SHM reaction to failure. This could be used in the early development of SHM to provide designers with information on the best reaction to system failures.

Finally, the FSL method presented in this work provides a means for analyzing the impact of failures that do not propagate along designed EMS flows paths. While this analysis can be useful for determining the impact of failures the next step would be using this method to determine a way to mitigate these failures. One part of this approach is to expand the modeling representation to include the severity of new flow type failures. For example, specifying that a tank has a minor or major leak can provide system architecture information to designers through the resulting impact to different nearby components (where "nearness" would be related to severity). Currently, the FSL augmented FFIP method provides the impact of failures based on the underling behavioral models. The results could be presented in the opposite perspective and identify the parts of the component behavior that lead to failure. With this information, designers could select components with safer or even fault mitigating behavior.

8.2 Concluding Remarks

The ability to evaluate the effect of failures in the design stage can provide designers with decision-making tools to improve complex system safety and reliability. Previous work has identified function-based failure analysis as a possible means for such a tool. One method that has been developed that specifically identifies how failures affect other system elements is the Function Failure Identification and Propagation (FFIP) framework. In attempting to meet the challenges of modeling failures in complex systems with the FFIP framework several limitations were found and solutions to those limitations represent the content of this research. Specifically, a previous lack of formal software representation limited previous FFIP work to hardware systems. The expansion of the method for software/hardware systems using system monitors was the first contribution of this research. Second, the original FFIP framework lacked a way to formally compare designs or particular failure scenarios. By quantifying the impact of failures with respect to system functions this research shows how FFIP can be used as a design decision-making tool. The most significant limitation of the FFIP

framework was the reliance on “as-designed” system representation for fault propagation identification. This limitation is shared with many conceptual and actual design reliability methods. The Flow State Logic (FSL) method presented in this work is adapted specifically to the FFIP framework. This augmentation allows the FFIP based simulation to analyze the affect of failures propagating outside the nominal design energy, material, and signal flow paths. The implications of this final part of the research may affect the paradigm of the design reliability community. Specially, reliance on designed system representation for failure analysis has not only been shown to be inadequate but a means of analysis beyond that representation has been shown to provide meaningful results. Future work in developing the FSL augmented FFIP method will focus on providing system health management performance information to designers and formalizing the quantitative design comparison results.

Bibliography

1. Kurtoglu, T., Tumer, I. Y., *A Graph-Based Fault Identification and Propagation Framework for Functional Design of Complex Systems.* Journal of Mechanical Design, 2008. **Vol. 130**(No. 5).
2. Jensen, D., Tumer, I.Y., Kurtoglu, T. *Modeling the Propagation of Failures in Software Driven Hardware Systems to Enable Risk-Informed Design*. in *ASME International Mechanical Engineering Congress and Exposition*. 2008. Boston, Massachusetts.
3. Forbus, K., *Qualitative Process Theory.* Artificial Intelligence, 1984. **24**: p. 85-168.
4. Kuipers, B.J., *Qualitative Simulation.* Artificial Intelligence, 1986. **29**(3): p. 289-338.
5. Struss, P., *Mathematical Aspects of Qualitative Reasoning.* International Journal of Artificial Intelligence in Engineering, 1988. **3**(3): p. 156-169.
6. Weld, D., De Kleer, J., *Readings in Qualitative Physics*, ed. M. Kauffman. 1987.
7. Patton, R., Frank, P., Clark, R., *Fault Diagnosis in Dynamic Systems: Theory and Applications*. 1989, Hertfordshire, UK: Prentice Hall.
8. Console, L., Hamscher, W., Kleer, J., *Readings in model-based diagnosis*, ed. M. Kauffman. 1989.
9. Dvorak, D.K., B. J., *Model Based Monitoring of Dynamic Systems.* IJCAI, 1989.
10. Wiliams, B.C., Nayak, P. P. *A Model-based Approach to Reacting Self-Configuring Systems*. in *Proceedings of AAAI-96*. 1996.
11. Kurien, J., Nayak., P. *Back to the Future with Consistency-based Trajectory Tracking*. in *AAAI/IAAI Conference*. 2000.
12. Korbicz, J., Koscielny, J. M., Kowalczuk, Z., Cholewa, W. , *Fault Diagnosis: Models, Artificial Intelligence, Applications*, ed. Springer. 2004.
13. Giarratano, J.C., Riley, G.D. *Expert Systems: Principles and Programming*. 4th ed. 2004, Boston, MA: PWS Publishing Company.
14. Berenji, H., J. Ametha, Vengerov, D. *Inductive Learning For Fault Diagnosis*. in *12th IEEE International Conference on Fuzzy Systems*. 2003.
15. Yairi, T., Kato, Y., Hori, K. *Fault Detection by Mining Association Rules from House-keeping Data*. in *SAIRAS*. 2001.
16. Lyu, M.R., *Software Reliability Engineering: A Roadmap.* Future of Software Engineering (FOSE'07), 2007. p. 153-170.

17. Teng, X., Pham, H., *Reliability Modeling of Hardware and Software Interactions, and its Applications.* IEEE Transactions on Reliability, 2006. **Vol. 55**(No. 4): p. 571-577.

18. Huang, B., Li, X., Bernstein, J., Smidts, C. , *Study of the Impact of Hardware Fault on Software Reliability*, in *Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering*. 2005, ISSRE.

19. Martin, R.J., Marthur, A. P. . *Software and Hardware Quality Assurance: Towards a Common Platform for High Reliability*. in *IEEE Conference on Communication*. 1990: Supercomm ICC '90.

20. Iyer, R.K., *Hardware-related software errors: measurement and analysis.* IEEE Transactions on Software Engineering, 1985. **SE-11**(2): p. 223-230.

21. Kanoun, K., Ortalo-Borrel, M. , *Fault-Tolerant Systems Dependability-Explicit Modeling of Hardware and Software Component-Interactions.* IEEE Transactions on Reliability, 2000. **Vol. 49**(No. 4).

22. Chrsitmansson, J., Hiller, M.,Rimen, M. *An Experimental Comparison of Fault and Error Injection*. in *The Ninth International Symposium on Software Reliability Engineering* 1998: *issre*.

23. Lyu, M.R., *Handbook of Software Reliability Engineering*. 1996, New York: Mc-Graw-Hill.

24. Li, B., Li, M., Ghose,S., Smidts, C. . *Integrating Software into PRA*. in *14th International Symposium on Software Reliability Engineering*. 2003.

25. Defense, D.o., *Procedures for Performing Failure Mode, Effects, and Criticality Analysis*, MIL-STD-1629A, Editor.

26. Vesely, W., Goldberg, F., Roberts, N., Haasl, D, *Fault Tree Handbook (NUREG-0492)*. 1981, Washington, D.C.: Division of Systems and Reliability Research, Office of Nuclear Regulatory Research, U.S. Nuclear Regulatory Commission.

27. Henley, E., Kumamoto, H., *Probabilistic risk assessment*. 1992: IEEE Press New York.

28. Stamatelatos, M., *Probabilistic Risk Assessment: What Is It and Why Is It Worth Performing It?* NASA Office of Safety and Mission Assurance, 2000.

29. Suh, N., *Axiomatic design: advances and applications*. 2001: Oxford University Press, USA.

30. Pahl, G., Beitz, W., *Engineering Design: A Systematic Approach*. 1996: Springer Verlag.

31. Kececioglu, D., *Reliability engineering handbook, volume 1*. 2002: DEStech Publications, Inc.

32. Taguchi, G., *Introduction to quality engineering.* 1986.

33. Clausing, D., Frey, D., *Improving System Reliability by Failure-Mode Avoidance Including Four Concept Design Strategies.* Systems Engineering, 2005. **8**(3).
34. Frey, D., Li, X., *Validating robust parameter design methods*. 2004.
35. Greer, J., Jensen, D., Wood, K., *Effort flow analysis: a methodology for directed product evolution.* Design Studies, 2004. **25**(2): p. 193-214.
36. Otto, K., Wood, K., *Product Evolution: A Reverse Engineering and Redesign Methodology.* Research in Engineering Design, 1998. **10**(4): p. 226-243.
37. Hirtz, J., et al., *A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts.* Research in Engineering Design, 2002. **13**(2): p. 65-82.
38. Stone, R., Wood, K., *Development of a Functional Basis for Design.* Journal of Mechanical Design, 2000. **122**(4): p. 359-370.
39. Stone, R., Tumer, I.Y., Van Wie, M., *The Function-Failure Design Method.* Journal of Mechanical Design, 2005. **127**: p. 397.
40. Tumer, I.Y., Stone, R., Bell, D. *Requirements for a Failure Mode Taxonomy for use in Conceptual Design*. in *Proc. 11th International Conference on Engineering Design*. 2003. Stockolm, Sweden.
41. Hutcheson, R., McAdams, D., Stone, R., Tumer, I.Y. *A Function-Based Method for Analyzing Critical Events*. in *Proc. 18th International Conference on Design Theory and Methodology IDETC/CIE2006*. 2006. Philadelphia, PA.
42. Grantham Lough, K., Stone, R., Tumer, I.Y., *The Risk in Early Design (RED) Method: Likelihood and Consequence Formulations*. in *Proceedings of DETC'06*. 2006. Philadelphia, PA: ASME.
43. Krus, D., Grantham Lough, K. *Applying function-based failure propagation in conceptual design*. in *Proc. 19th International Conference on Design Theory and Methodology IDETC/CIE2007*. 2007. Las Vegas, Nevada.
44. Meshkat, L., Jenkins, S., Mandutianu, S., Heron, V., *Automated Generation of Risk and Failure Models during Early Phase Design*, in *Proc. IEEE Aerospace Conference*. 2008: Big Sky, MT.
45. Clarkson, P., Simons, C., Eckert, C., *Predicting Change Propagation in Complex Design.* Journal of Mechanical Design, 2004. **126**: p. 788.
46. Wang, K., Jin, Y., *An Analytical Approach to Functional Design*, in *Proc. 14th International Conference on Design Theory and Methodology IDETC/CIE2002*. 2002: Montreal, Quebec, Canada. p. 449-459.
47. Huang, Z., Jin, Y. *Conceptual Stress and Conceptual Strength for Functional Design-for-Reliability*. in *Proc. 20th International Conference on Design Theory and Methodology IDETC/CIE2008*. 2008. Brooklyn, NY.

48. Kurtoglu, T., Tumer, I. Y. *A Risk-Informed Decision Making Methodology for Evaluating Failure Impact of Early System Designs*. in *Proc. 20th International Conference on Design Theory and Methodology IDETC/CIE2008*. 2008. Brooklyn, NY.

49. Kurtoglu, T., et al. *Deriving a Component Basis for Computational Functional Synthesis*. in *International Conference on Engineering Design, ICED'05*. 2005. Melbourne, Australia.

50. Caughlin, D. *Integration of Object-Oriented and Functional Modeling and Design Methods*. in *Proceedings of SPIE - The International Society for Optical Engineering*. 1997.

51. Wang, E.Y., Cheng, Betty H.C., *Formalizing the functional model within object-oriented design.* International Journal of Software Engineering and Knowledge Engineering, 2000. **Vol. 10**(No. 1): p. 5-30.

52. Gruber, T.R., Vemuri, S., Rice, J. , *Model-based Virtual Document Generation.* International Journal of Human-Computers Studies, 1997. **Vol. 46**(No. 6): p. 687 - 706.

53. Patty, J., Dismukes, K. . *RCS Jet Selection*. NASA human space flight reference article 2008; Available from: http://spaceflight.nasa.gov/shuttle/reference/shutref/orbiter/rcs/select.html.

54. Kutoglu, T., Tumer, I. Y., *A Risk-Informed Decision Making Methodology for Evaluating Failure Impact of Early System Designs*, in *Procedings of the ASME 2008 International Design Engineering Technical Confrence & Computer and Information in Engineering Confrence*. 2008: Brooklyn, New York, USA.

55. Kurtoglu, T., Tumer, I.Y., Jensen, D., *A Function Failure Reasoning Method for Evaluation of Conceptual System Architectures.* Submitted to J. of Research in Engineering Design, 2009(April).

56. Otto, K., Wood, K., *Product Design: Techniques in Reverse Engineering, Systematic Design, and New Product Development*. 2001, New York: Prentice Hall.

57. Stone, R., Wood, K., *Development of a Functional Basis for Design.* Journal of Mechanical Design, 2000. **122**: p. 359.

58. Stone, R., Tumer, I.Y., Van Wie, M., *The Function Failure Design Method.* Journal of Mechanical Design, 2004. **127**(3): p. 397-407.

59. Tumer, I.Y., Stone, R.B., *Mapping Function to Failure During High-Risk Component Development.* Research in Engineering Design, 2003. **14**(1): p. 25-33.

60. Kurtoglu, T., et al. *Capturing Empirically Derived Design Knowledge for Creating Conceptual Design Configurations*. in *Proceedings of IDETC/CIE 2005*. 2005. Long Beach, CA: ASME.

61. Poll, S., et al. *Advanced diagnostics and prognostics testbed*. 2007.

62. Hutcheson, R., Tumer, I.Y., *Function based co-design paradigm for robust health management*. in *International workshop on structural health management*. 2005. Palo Alto, CA.

63. Wood, W.H., Agogino, A.M., *Decision based conceptual design: Modeling and navigating heterogeneous design spaces.* Journal of Mechanical Design, 2005. **127**(1): p. 2-11.

64. Bryant, C., Stone, R., Greer, J., McAdams, A., Kurtoglu, T., Campbell, M. *A Function-Based Component Ontology for System Design*. in *International Conerence on Enginering Design*. 2007. Paris, France.

65. Kitamura, Y., Mizoguchi, R., *Ontology-based description of functional design knowledge and its use in a functional way server.* Expert Systems With Applications, 2003. **24**(2): p. 153-166.

66. Kurtoglu, T., Johnson, S., Barszcz, E., Johnson, J., Robinson, P. *Integrating System Health Management into Early Design of Aerospace Systems Using Functional Fault Analysis*. in *International Conference on Prognostics and Health Management*. 2008. Denver, CO.

67. Kurtoglu, T., et al. *Integrating system health management into the early design of aerospace systems using functional fault analysis*. in *Prognostics and Health Management Conference*. 2008. Denver, CO.

68. Bohm, M., Stone, R., Szykman, S., *Enhancing Virtual Product Representations for Advanced Design Repository Systems.* Journal of Computing and Information Science in Engineering, 2005. **5**: p. 360.

Appendix A

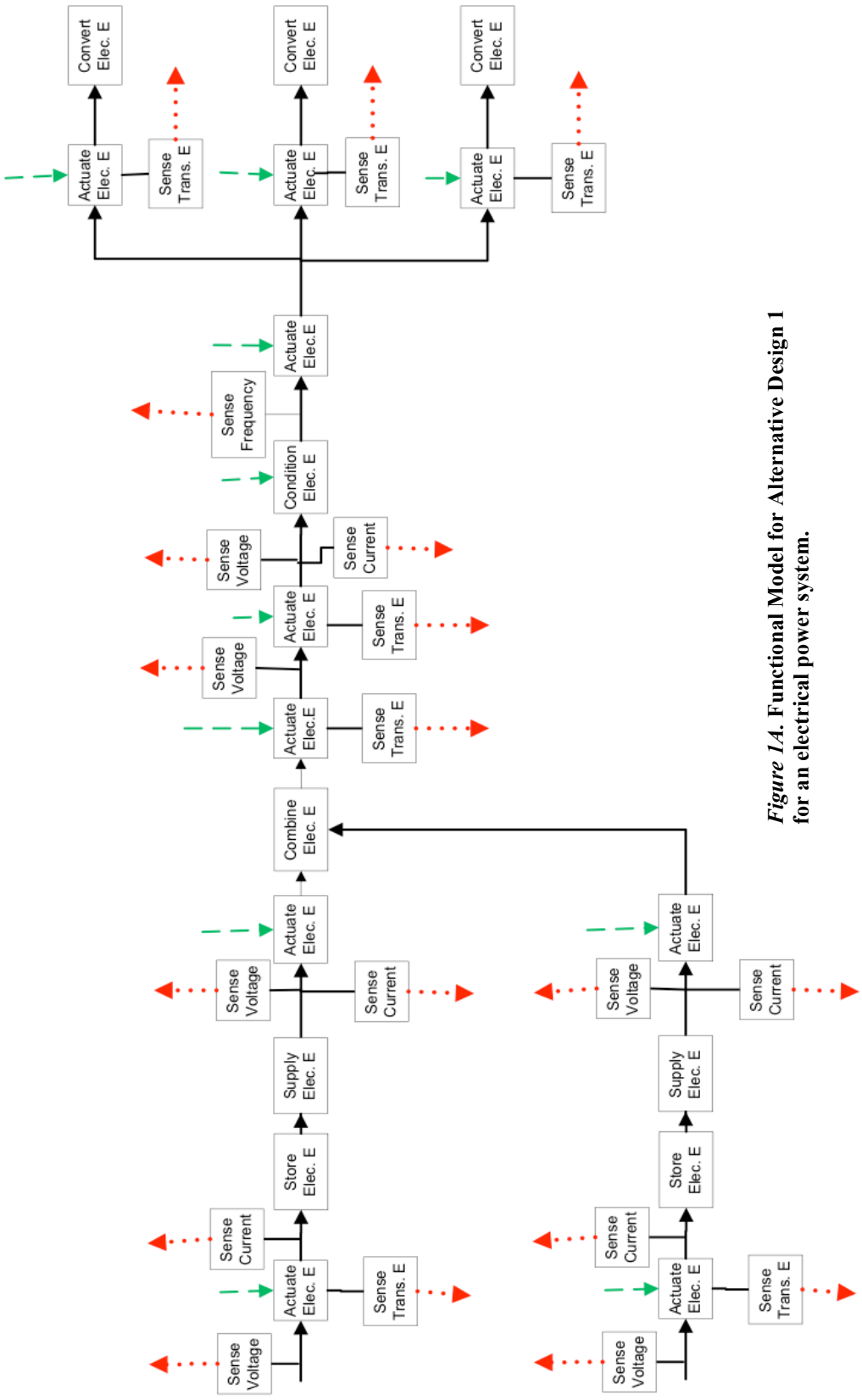


Figure 1A. **Functional Model for Alternative Design 1 for an electrical power system.**

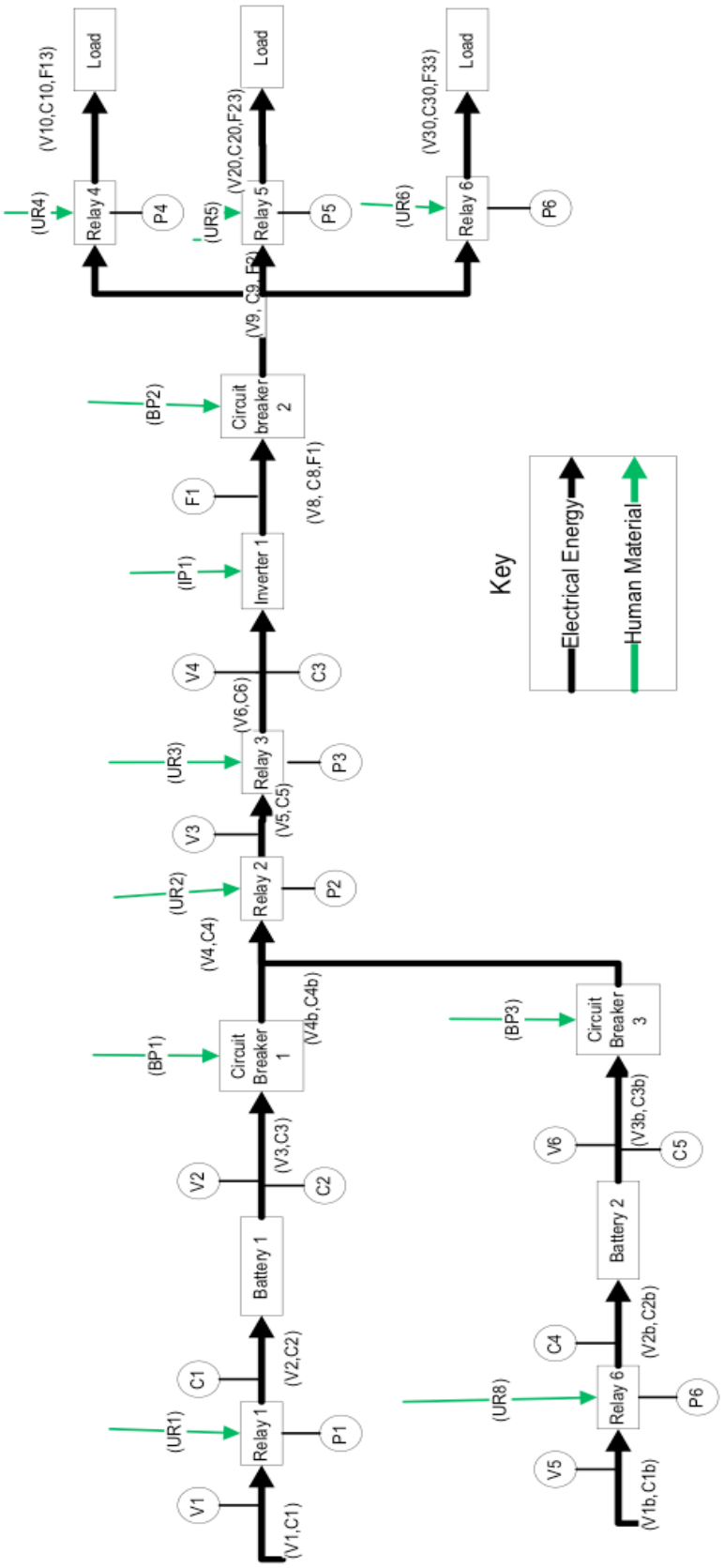


Figure 2A. **Configuration Flow Graph for Alternative Design 1 for an electrical power system.**

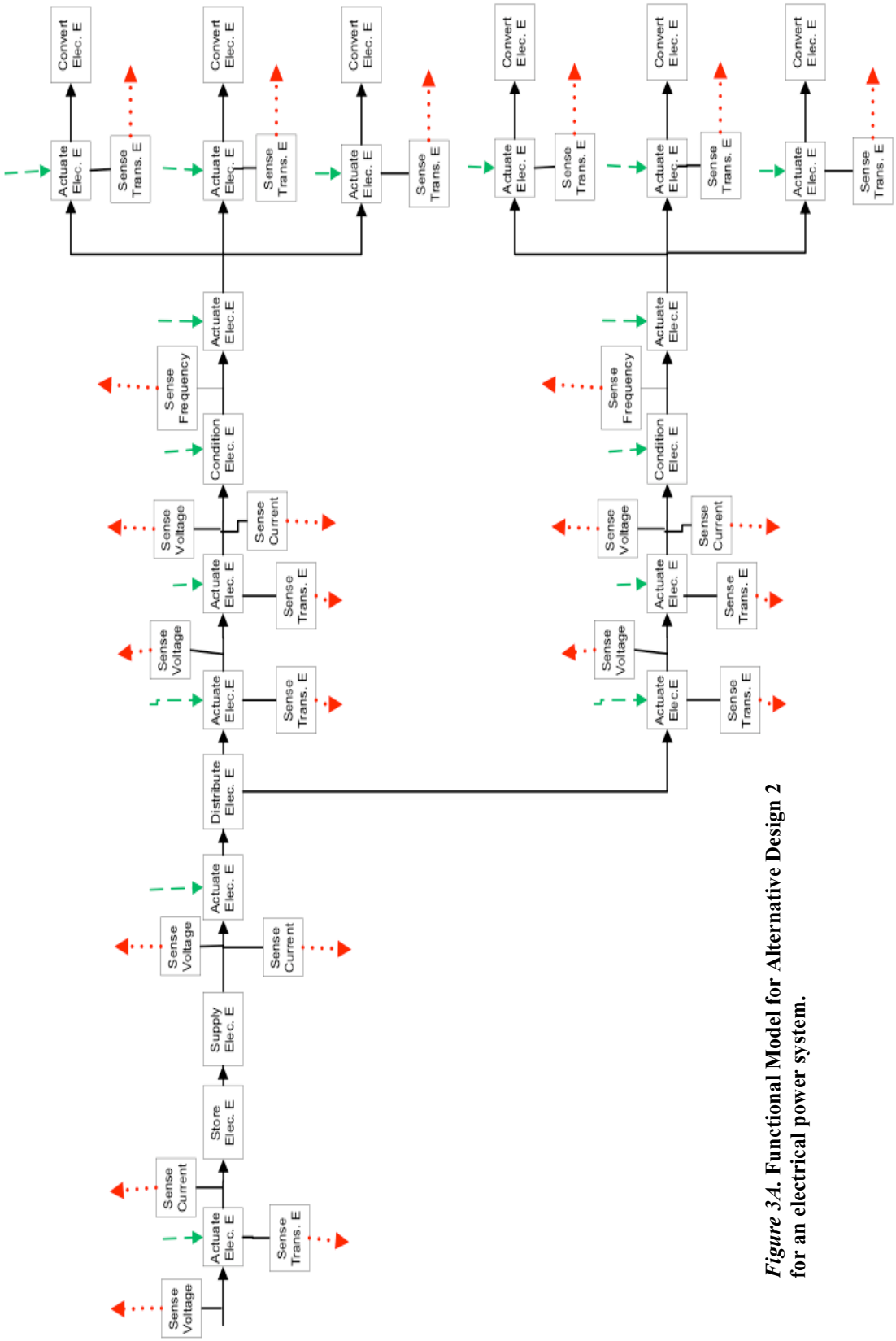


Figure 3A. **Functional Model for Alternative Design 2 for an electrical power system.**

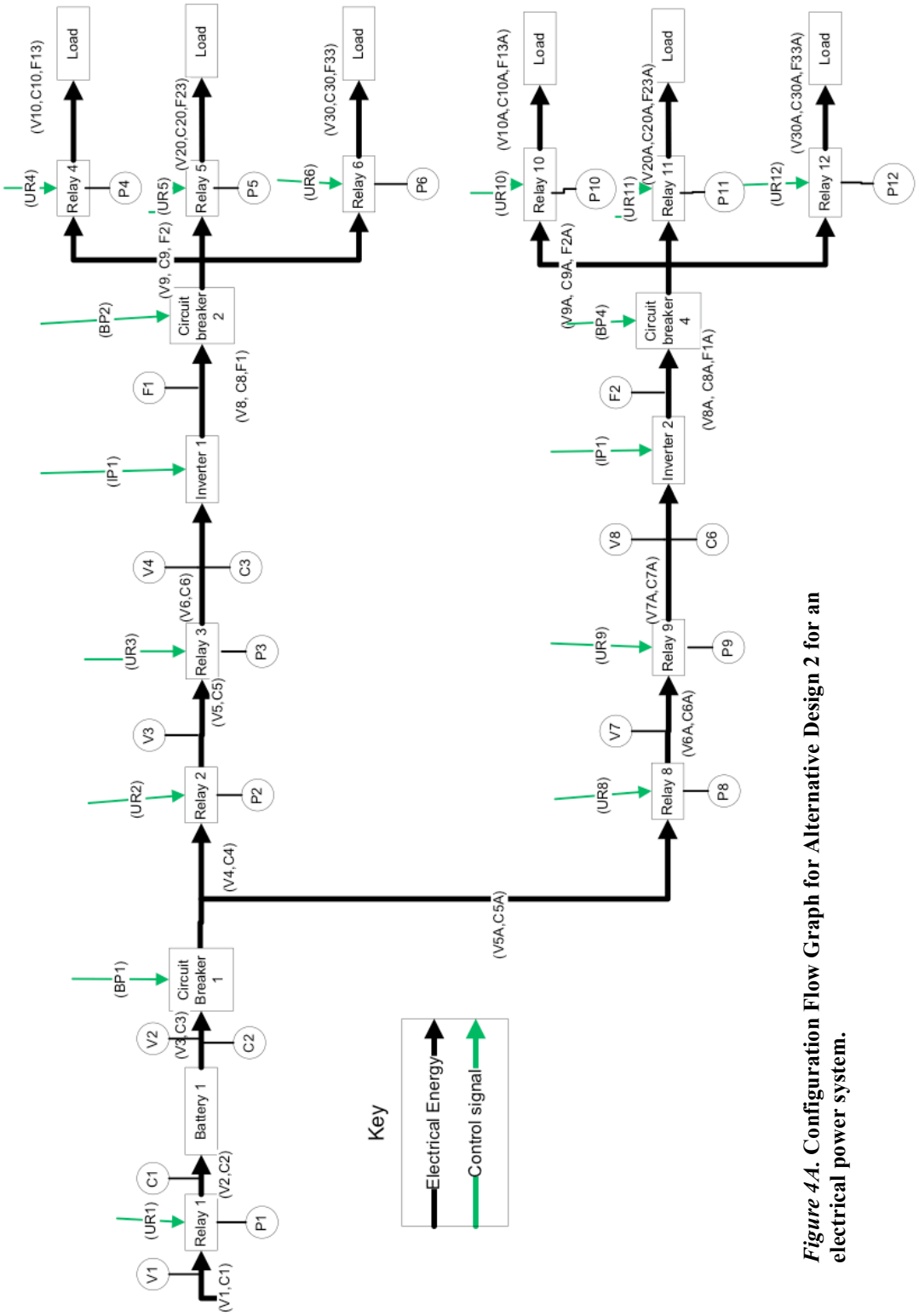


Figure 4A. **Configuration Flow Graph for Alternative Design 2 for an electrical power system.**

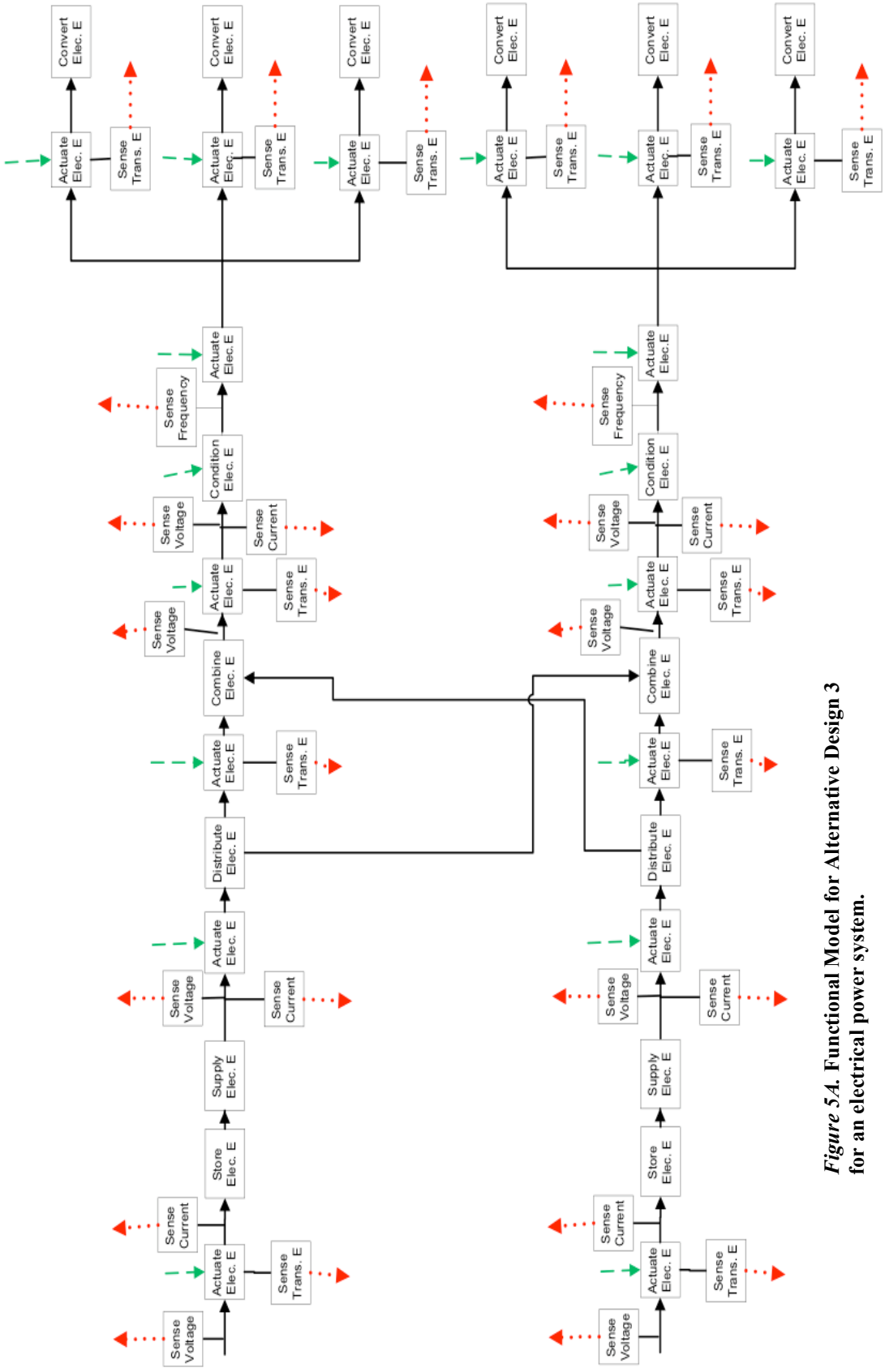


Figure 5A. **Functional Model for Alternative Design 3 for an electrical power system.**

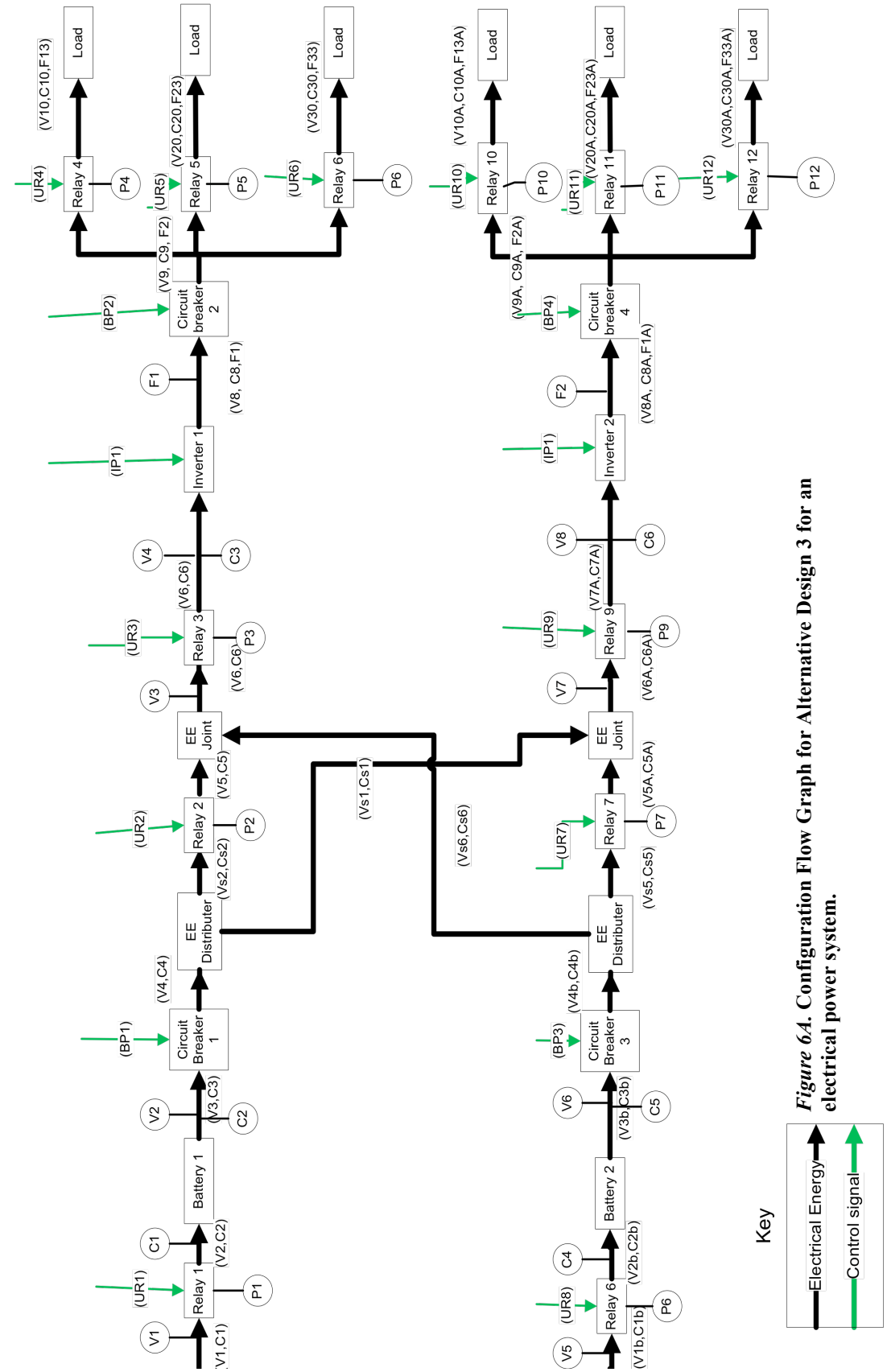


Figure 6A. **Configuration Flow Graph for Alternative Design 3 for an electrical power system.**

Table 1A. **Propagation Based Fuel and Oxidizer Valve Behavior**

VALVE DESIGN			
Flow	Critical Level	Propagates?	Mode Change
Designed Flows			
Liquid Material	None	Yes	None
Control Signal	On/Off	No	On/Off
Potential Flows			
Generic Energy	Low	Yes	Stuck Closed
Human Energy	Low	Yes	Stuck Closed
Acoustic Energy	High	Yes	Stuck Closed
Biological Energy	High	Yes	Stuck Closed
Chemical Energy	High	Yes	Stuck Closed
Electrical Energy	High	Yes	Stuck Closed
Electromagnetic Energy	Low	Yes	Stuck Closed
Human Material	Low	No	Leak
Gas Material	High	Yes	Stuck Closed
Liquid Material	High	Yes	Stuck Closed
Solid Material	Low	No	Leak
Plasma Material	Low	No	Leak

Table 2A. **Propagation Based Fuel and Oxidizer Line Behavior**

LINE			
Flow	Critical Level	Propagates?	Mode Change
Designed Flows			
Liquid Material	None	Yes	None
Potential Flows			
Generic Energy	High	No	Leak
Human Energy	High	No	Leak
Acoustic Energy	High	No	Leak
Biological Energy	High	No	Leak
Chemical Energy	High	No	Leak
Electrical Energy	High	No	Leak
Electromagnetic Energy	High	No	Leak
Human Material	Low	No	Blocked Flow
Gas Material	High	No	Leak
Liquid Material	High	No	Leak
Solid Material	Low	No	Blocked Flow
Plasma Material	Low	No	Leak

Table 3A. **Propagation Based Fuel and Oxidizer Tank Behavior**

TANK			
Flow	Critical Level	Propagates?	Mode Change
Designed Flows			
Liquid Material	None	Yes	None
Potential Flows			
Generic Energy	None	No	None
Human Energy	None	No	None
Acoustic Energy	None	No	None
Biological Energy	None	No	None
Chemical Energy	None	No	None
Electrical Energy	None	No	None
Electromagnetic Energy	None	No	None
Human Material	High	No	Leak
Gas Material	None	No	None
Liquid Material	None	No	None
Solid Material	High	No	Leak
Plasma Material	High	No	Leak

Table 4A. **Propagation Based Temperature Sensor Behavior**

TEMP SENSOR			
Flow	Critical Level	Propagates?	Mode Change
Designed Flows			
Liquid Material, Temp	None	No	None
Status Signal	None	Yes	None
Potential Flows			
Generic Energy	High	Yes	Stuck at High
Human Energy	High	Yes	Stuck at High
Acoustic Energy	High	Yes	Stuck at High
Biological Energy	High	Yes	Stuck at High
Chemical Energy	High	Yes	Stuck at High
Electrical Energy	High	Yes	Stuck at High
Electromagnetic Energy	High	Yes	Stuck at High
Human Material	Low	No	No Signal
Gas Material	Low	No	No Signal
Liquid Material	Low	No	No Signal
Solid Material	Low	No	No Signal
Plasma Material	Low	No	No Signal

Table 5A. **Propagation Based Pressure Sensor Behavior**

PRESSURE SENSOR			
Flow	Critical Level	Propagates?	Mode Change
Designed Flows			
Liquid Material, Temp	None	No	None
Status Signal	None	Yes	None
Potential Flows			
Generic Energy	High	Yes	Stuck at High
Human Energy	High	Yes	Stuck at High
Acoustic Energy	High	Yes	Stuck at High
Biological Energy	High	Yes	Stuck at High
Chemical Energy	High	Yes	Stuck at High
Electrical Energy	High	Yes	Stuck at High
Electromagnetic Energy	High	Yes	Stuck at High
Human Material	Low	No	No Signal
Gas Material	Low	No	No Signal
Liquid Material	Low	No	No Signal
Solid Material	Low	No	No Signal
Plasma Material	Low	No	No Signal

Table 6A. **Propagation Based Reaction Chamber Behavior**

RXN Chamber			
Flow	Critical Level	Propagates?	Mode Change
Designed Flows			
Liquid Material	Non-Nominal	No	Varies
Gas Material	None	Yes	None
Potential Flows			
Generic Energy	None	No	None
Human Energy	None	No	None
Acoustic Energy	None	No	None
Biological Energy	None	No	None
Chemical Energy	None	No	None
Electrical Energy	None	No	None
Electromagnetic Energy	None	No	None
Human Material	High	No	Leak
Gas Material	High	No	Leak
Liquid Material	High	No	Leak
Solid Material	High	No	Leak
Plasma Material	High	No	Leak

Table 7A. **Propagation Based Controller Behavior**

Controller			
Flow	Critical Level	Propagates?	Mode Change
Designed Flows			
Status Signals	Non-Nominal	No	Off
Control Signals	None	Yes	None
Potential Flows			
Generic Energy	Low	No	Failed Off
Human Energy	Low	No	Failed Off
Acoustic Energy	Low	No	Failed Off
Biological Energy	Low	No	Failed Off
Chemical Energy	Low	No	Failed Off
Electrical Energy	Low	No	Failed Off
Electromagnetic Energy	Low	No	Failed Off
Human Material	Low	No	No Power
Gas Material	Low	No	No Power
Liquid Material	Low	No	No Power
Solid Material	Low	No	No Power
Plasma Material	Low	No	No Power

Table 8A. **Single Component Scenarios Applied to Simulation**

Fuel Tank	Empty	
	Pressurized	
	Leak	
Fuel Valve	Leak	
	Stuck Open	
	Stuck Closed	
Fuel Line	Leak	
	Blocked	
Fuel Temperature Sensor	No Signal	
	Stuck Zero	Not Used
	Stuck Low	
	Stuck Nominal	
	Stuck High	

Reaction Chamber	Leak	
Pressure Sensor	No Signal	
	Stuck Zero	Not Used
	Stuck Low	
	Stuck Nominal	
	Stuck High	
Reaction Tempreature Sensor	No Signal	
	Stuck Zero	Not Used
	Stuck Low	
	Stuck Nominal	
	Stuck High	
Oxidizer Tank	Pressurized	
	Empty	
	Leak	
Oxidizer Valve	Leak	
	Stuck Closed	
	Stuck Open	
Oxidizer Line	Leak	
	Blocked	
Oxidizer Temperature Sensor	No Signal	
	Stuck Zero	Not Used
	Stuck Low	
	Stuck Nominal	
	Stuck High	
Controller	Failed Off	Not Used
	Failed On	
	No Power	

Table 9A. **Scenarios Applied to FSL augmented FFIP Simulation**

New Flow Scenarios

	Initial event	Impact 1	Impact 2
1	Fuel Valve Leak	Low liq from pot port	import to controller
2	Oxidizer Valve leak	Low liq from pot port	import to press sensor
3	fuel line leak	Low liq from pot port	import to temp sensor
4	Oxidizer line leak	Low liq from pot port	import to rxn chamber
5	Rxn chamber leak	high gas from pot port	import to oxidizer line
6	Low Acoustic to controller		
7	Low acoustic to valve		
8	High acoustic to valve		
9	high acoustic temp sensor		
10	high acoustic to fuel line	Low liq from pot port	import to valve
11	low solid mat to valve	Low liq from pot port	import to controller
12	low solid to line		
13	high solid to chamber	high gas from pot port	import to controller
14	high human material to tank	high liq from pot port	import to valve
15	High solid to controller		

Table 10A. **Results of the Single Component Fault Scenarios Applied to the FFIP Simulation.**

NIT	FAILURE	FFI
Fuel Tank	Pressurized	0.246007143
	Empty	0.38885
	Leak	0.246007143
Fuel Valve	Leak	0.150778571
	Stuck Open	0.460271429
	Stuck Closed	0.150778571
Fuel Line	Leak	0.150778571
	Blocked	0.150778571
Fuel Temperature Sensor	No Signal	1.547464286
	Stuck Low	0.150778571
	Stuck Nominal	0
	Stuck High	0.341235714
Reaction Chamber	Leak	0.150778571
Pressure Sensor	No Signal	1.579207143
	Stuck Low	0.150778571
	Stuck Nominal	0
	Stuck High	0.372978571
Reaction Tempreature Sensor	No Signal	1.579207143
	Stuck Low	0.150778571
	Stuck Nominal	0
	Stuck High	0.372978571
Oxidizer Tank	Pressurized	0.246007143
	Empty	0.38885
	Leak	0.246007143
Oxidizer Valve	Leak	0.150778571
	Stuck Closed	0.150778571
	Stuck Open	0.460271429
Oxidizer Line	Leak	0.150778571
	Blocked	0.150778571
Oxidizer Temperature Sensor	No Signal	1.547464286
	Stuck Low	0.150778571
	Stuck Nominal	0
	Stuck High	0.341235714
Controller	Failed On	0.603114286
	No Power	1.452235714

Table 11A. **Results of the Single Component Fault Scenarios and New Flow Failures Applied to the FSL FFIP Simulation.**

UNIT	FAILURE	FFI
Fuel Tank	Pressurized	0.246007143
	Empty	0.38885
	Leak	0.246007143
Fuel Valve	Leak	0.150778571
	Stuck Open	0.460271429
	Stuck Closed	0.150778571
Fuel Line	Leak	0.150778571
	Blocked	0.150778571
Fuel Temperature Sensor	No Signal	1.547464286
	Stuck Low	0.150778571
	Stuck Nominal	0
	Stuck High	0.341235714
Reaction Chamber	Leak	0.150778571
Pressure Sensor	No Signal	1.579207143
	Stuck Low	0.150778571
	Stuck Nominal	0
	Stuck High	0.372978571
Reaction Temperature Sensor	No Signal	1.579207143
	Stuck Low	0.150778571
	Stuck Nominal	0
	Stuck High	0.372978571
Oxidizer Tank	Pressurized	0.246007143
	Empty	0.38885
	Leak	0.246007143
Oxidizer Valve	Leak	0.150778571
	Stuck Closed	0.150778571
	Stuck Open	0.460271429
Oxidizer Line	Leak	0.150778571
	Blocked	0.150778571
Oxidizer Temperature Sensor	No Signal	1.547464286
	Stuck Low	0.150778571
	Stuck Nominal	0
	Stuck High	0.341235714
Controller	Failed On	0.603114286
	No Power	1.452235714
Fuel Valve Leak		1.452235714
Oxidizer Valve Leak		1.047514286
Fuel Line Leak		0.94435
Oxidizer Line Leak		0.150778571
Rxn Chamber Leak		0.150778571
Low Acoustic to Controller		1.047514286
Low Acoustic to Valve		1.357007143

High Acoustic to Valve		0.150778571
High Acoustic to Temp. Sensor		1.547464286
High Acoustic to Fuel Line		0.150778571
Low Solid Matter to Valve		0.246007143
Low Solid Matter to Line		0.150778571
High Solid Matter to Chamber		0.150778571
High Human Material to Tank		0.246007143
High Solid to Controller		1.452235714