AN ABSTRACT OF THE DISSERTATION OF

David Charles Jensen for the degree of Doctor of Philosophy in Mechanical Engineering presented on April 30, 2012.

Title: Enabling Safety-Informed Design Decision Making through Simulation, Reasoning and Analysis

Abstract approved: _____

Irem Y. Tumer

While many organizations claim to "put safety first," safety is rarely considered early in the design process when system-level architectural decisions are made. Instead, system design follows an abstraction-to-detail process to first meet functional and then performance requirements. Following this process, safety assurance occurs in the later stages of design through a rigorous expert review process. The significant cost of safety-based redesign and the growing complexity of engineered systems motivates a need for early design-stage fault analysis. This research presents a novel method of including safety into the model-based design and analysis of complex systems using low-fidelity behavior simulations. Specifically, this research demonstrates the adaption of the functional design process to explicitly include the system property of safety in the system representation. Next, early design fault analysis is extended to connect component failure behavior to system-level hazards. Finally, this research develops three methods of results clustering to provide different evaluation metrics of the system design. In summary, this research demonstrates a framework for incorporating safety into early design decision making. This research addresses safety and failure in the design of complex systems incorporating diverse technology domains as found in energy, transportation, and aerospace systems.

Enabling Safety-Informed Design Decision Making through Simulation, Reasoning and Analysis

by

David Charles Jensen

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented April 30, 2012
Commencement June 2012

Doctor of Philosophy dissertation of David Charles Jensen presented on April 30, 2012.

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

David Charles Jensen, Author

ACKNOWLEDGEMENTS

This dissertation represents the culmination of over four years of graduate research that would not have been possible but for the technical and personal support and guidance I received from many individuals. First, thank you to my advisor and friend Dr. Irem Y. Tumer for your guidance in helping me mature in both research and life over the last four and a half years. I am also grateful for the invaluable technical guidance of Drs. Christopher Hoyle and Tolga Kurtoglu whose expert advice made this research possible. The high quality review and regular feedback I received from my lab mates in the Complex Engineered Systems Laboratory enabled this work to go from rough and ill-formed concepts to what I have today. Thank you for that Douglas Van Bossuyt, Sarah Oman, Brady Gilchrist, Mike Koopmans, Joseph Piacenza, and Hoda Mehrpouyan. Additionally I am grateful for the hard work put in by the undergraduate researchers who did the programming grunt work with me, especially Josh Wilcox.

I am so thankful for the support and sacrifice my family has made for me. I know my wife and boys dearly missed the time with their husband and father and I will be forever grateful for their love, prayers and patience. Finally, I hope that my time here at Oregon State is more significant than these pieces of paper and bears fruit into eternity. Soli Deo Gloria.

TABLE OF CONTENTS

Page

1 Introduction 1

1.1 Complex Engineered Systems 1

1.2 Overview of the Work 4

1.3 Terminology 5

2 State of the Art 7

2.1 System Design 8

2.2 Reliability Methods and Tools 10

2.2.1 Validation Stage Approaches for Reliability Analysis 10

2.2.2 Design Stage Approaches for Reliability Analysis 12

2.3 Hazard Mitigation Methods 13

2.3.1 Validation Stage Approaches for Hazard Analysis 14

2.3.2 Design Stage Approaches for Hazard Analysis 15

2.4 Summary 15

3 The Safety-Centric Design Decision Enabling Framework 18

3.1 Overview 19

3.2 Methodology Use Case 19

3.2.1 Example System for Application of Methodology 21

3.3 Contributions and Limitations 22

3.3.1 Intellectual Merit 22

3.3.2 Limitations 23

4 Safety Guided Design 24

4.1 Background 25

4.1.1 Models and Modeling Languages 25

4.1.2 Function-Based Conceptual Design 29

4.1.3 System Theoretic Process Analysis 32

4.2 The Safety Function 34

4.3 Incorporating Safety Functions into Model-Based, Functional System Design 36

4.4 Example Application 37

4.5 Summary 39

TABLE OF CONTENTS (Continued)

Page

5 Function Failure Reasoning for Complex System Design 45

5.1 Background . . . 45
5.1.1 Reasoning about Faults During Operations . . . 45
5.1.2 Traditional Risk and Reliability Analysis . . . 46
5.1.3 Research in Function-Based Failure and Risk Analysis During Conceptual Design . . . 47

5.2 Design Analysis with the Function Failure Identification and Propagation Framework . . . 50

5.3 Function Failure Reasoning . . . 52
5.3.1 Step 1: System-Level Modeling . . . 53
5.3.2 Step 2: Computing the Function Criticality Rating (FCR) . . . 54
5.3.3 Step 3: Computing the Functional Failure Impact (FFI) . . . 55
5.3.4 Step 4: Computing the Reduction in Risk (RIR) . . . 59

5.4 Electrical Power System Testbed Design Example . . . 59
5.4.1 EPS Testbed Baseline System Architecture and its Modeling using the FFIP Framework . . . 60
5.4.2 EPS Testbed Alternative System Architectures . . . 61

5.5 FFR Method Applied to the Electrical Power System Design . . . 66
5.5.1 Step 1: System Models for the EPS . . . 66
5.5.2 Step 2: Determination of Functional Criticality Ratings for the EPS 67
5.5.3 Step 3: Computing the Functional Failure Impact for the EPS Scenarios . . . 68

5.6 Step 4: Computing RIR for the EPS Alternative System Architectures . . . 71

5.7 Discussion of Results . . . 74

5.8 Summary . . . 75

6 Expanding FFIP Analysis with Fault Mode Dependency and Flow State Reasoning 80

6.1 Introduction . . . 80

6.2 Background . . . 82

6.3 Flow State Reasoning Framework . . . 83
6.3.1 Establishing a Labeling Scheme for Potential EMS Flows . . . 84
6.3.2 Identifying Potential Flows by Mapping Failure Modes . . . 86
6.3.3 Extending a Design Stage Failure Analysis . . . 88

TABLE OF CONTENTS (Continued)

Page

6.4 Application to a Failure Analysis Tool for Concept Evaluation 89
6.4.1 Early Design Failure Analysis Using FFIP 90
6.4.2 Reasoning About the State of Flows in a System 93
6.4.3 Expanded Behavioral Models 93
6.4.4 Fault Simulation and New Propagation Paths 95
6.4.5 Discussion of Results 96
6.5 Summary 100

7 A Safety Function Failure Reasoning Method 102
7.1 Background 102
7.2 Safety Function State Reasoning 103
7.3 Example Application 105
7.3.1 Case 1: Safety Function Control Structure Component Failure . . 105
7.3.2 Case 2: Fault-Dependent Failure Scenario 107
7.4 Summary 108

8 Clustering Failure Analysis Results to Enable Design Decision Making 110
8.1 Introduction 110
8.2 Background 112
8.2.1 Latent Class Analysis 113
8.3 Overview and Comparison of Methods 115
8.4 Example Application 115
8.5 Consequence and Risk Metric 119
8.6 Latent Class Analysis 121
8.7 Functional Similarity Metric 124
8.8 Summary 125

9 Special Issues: Large Complex Systems and the Analysis Sandbox 129
9.1 Introduction 129
9.1.1 Challenges for Complex System Design 129
9.1.2 Prior Work 130
9.1.3 Objectives and Contributions 131
9.2 Background 132

TABLE OF CONTENTS (Continued)

Page

9.2.1 Risk Assessment in Complex System Design 132
9.2.2 Complex System Design Representation 133
9.2.3 Simulation-Based Design 134
9.3 Methodology 135
9.4 Case Study: Analysis of a Boiling Water Reactor Design 140
9.4.1 Abstractions: Functional Model and Configuration Flow Graph . . 144
9.4.2 Behavior and Function-Failure Logic Reasoning 146
9.5 Automation framework and user interface 147
9.6 Simulation Results 149
9.7 Conclusions and Future Work 155

10 Conclusions 157

Bibliography 159

Appendices 177
A Subsystem and Component Models 178
B EPS Design Alternatives 183

LIST OF FIGURES

Figure Page

1.1 Top-Down System Engineering approach represented as a "V" diagram (after Buede in [11]). 4

2.1 A model of how failures lead to unsafe system states. 17

4.1 OMG SysML diagrams used for modeling a system, from the OMG SysML website: [105]. 28

4.2 Contrasting two views of function found in the literature. 31

4.3 Generalized factors that are used to identify potential safety constraint violation scenarios in the STPA methodology [84]. 33

4.4 The general control structure for implementing a safety function which inhibits the system transition from a hazardous state to a mishap state. . 36

4.5 A SysML block definition diagram illustrating the composition of the control structure for implementing the safety function "Prevent permanent loss of orbital trajectory" for the hazardous system state of manual operator control. 40

4.6 A SysML block definition diagram illustrating the functional decomposition of the spacecraft maneuvering system. 41

4.7 A SysML internal block diagram illustrating the function structure of the spacecraft maneuvering system. 42

4.8 A SysML internal block diagram illustrating the connections of the control structure for implementing the safety function "Prevent permanent loss of orbital trajectory" for the hazardous system state of manual operator control. 43

4.9 A SysML block definition diagram illustrating the composition of the control structure after mapping to the components derived from the functional design process. 44

LIST OF FIGURES (Continued)

Figure Page

5.1 The schematic of the basic electrical power system (EPS) design. The system is used to provide power to various components including pumps, fans, etc. In this chapter, three alternative conceptual system architectures of the EPS design are studied. 52

5.2 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. 56

5.3 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. 62

5.4 Behavioral models for generic "relay" and "inverter" components. 64

5.5 Function Failure Logic relates the behavioral model to specific functional health. 65

5.6 The GUI for functional failure reasoning. 70

6.1 Overview of the Flow State Reasoning (FSR) Framework and its role in early design failure analysis. 85

6.2 *Top*: Nominal-state design system representation. *Bottom*: EMS flows that potentially exist for a system. 86

6.3 An initial fault cause-symptom mapping for identify potential fault propagation between components. 88

6.4 Graphical overview of the FFIP framework prior to the integration of the proposed Flow State Reasoning. 90

6.5 Functional model for a conceptual design of a liquid-fueled rocket engine. 91

6.6 Component configuration model for a conceptual design of a liquid-fueled rocket engine. 91

LIST OF FIGURES (Continued)

Figure Page

6.7 *Top*: The FFL reasoner inputs as compared to FSL reasoner inputs. *Bottom*: The logic used in the FFL reasoner as compared to the logic used in the FSL reasoners. 94

6.8 An example behavioral model from the original FFIP and the necessary addition of fault causes and symptoms. 95

7.1 Generalized factors that are used to identify potential safety constraint violation scenarios in the STPA methodology [84].*Repeated for clarity.* . . 104

8.1 Functional model of the electrical power system (EPS). 117

8.2 Component model of the Electrical Power System (EPS). 118

8.3 An example of the functional state output from simulating several failure scenarios. Each row represents the final functional impact of the component-level functions. 119

8.4 Representing the cost of each of the 677 failures simulated results in three clear groupings for this system. 120

8.5 The Akaike Information Criterion (AIC) as a function of the number of classes for the latent class model applied to the function failure simulation data. Minimum is at 5 classes of the latent variable of "System Failure." 122

8.6 A selection of some of the failure simulations classified in each of the five classes of the latent variable "System Failure." 123

8.7 Similarity maps created by comparing a state to all other listed states for functional similarity. 126

8.8 Part of the similarity and exact lists of scenarios with respect to the inverter fault. 127

9.1 Piping and Instrumentation Diagram of the example process. 136

9.2 Functional model of the example process. 136

9.3 Configuration flow graph of the example process. 137

LIST OF FIGURES (Continued)

Figure Page

9.4 Function Failure Logic for the Supply liquid material (white liquor) function. 138

9.5 Behavioral model of the white liquor tank component. 139

9.6 A flowchart describing how every combination of parameter values is simulated systematically to determine those combinations of parameter values that result in degradation or loss of functions. 141

9.7 The result of performing the procedure in Figure 9.6 when parameter 1 is RefInputLiquidFlow and parameter 2 is leakSize. 142

9.8 Top level CFG model for boiling water reactor core and its steam outlets. 145

9.9 Internals of the Reactor component in Figure 9.8. 146

9.10 Behavioral logic for the PressureControlValve component in Figure 9.8. . 147

9.11 Stateflow chart of the FFL reasoner for the "Transmit thermal energy" function in Table 9.1 148

9.12 User interface for specifying a set of FFIP simulation scenarios. 149

9.13 Algorithm for generating the set of parameterized FFIP simulations. . . . 150

9.14 Temperature of fuel rods in first phase FFIP simulation scenarios with parameter values resulting in healthy FFL verdicts. 151

9.15 Temperature of fuel rods in first phase FFIP simulation scenarios with parameter values resulting in degraded FFL verdicts. 152

9.16 Temperature of fuel rods in first phase FFIP simulation scenarios with parameter values resulting in lost FFL verdicts. 153

9.17 Temperature of fuel rods in second phase FFIP simulation: emergency power present. 154

9.18 Temperature of fuel rods in third phase FFIP simulation scenarios with narrowed ranges for parameter values. 155

10.1 Distinguishing synthesis from analysis in the context of design decision making. 158

LIST OF TABLES

Table		Page
1.1	Terminology Used Throughout This Work	6
5.1	Function States and Consequential Cost Factors	58
5.2	Components modes for the baseline EPS design	63
5.3	Flow State Variables for the Baseline Design	64
5.4	Distribution of FCR values for the Baseline EPS Design and the Three Alternative Designs.	67
5.5	Summary of Scenarios Selected as a Basis of Comparison	69
5.6	Function Failure Impact (FFI) and the Subsequent Reduction In Risk (RIR)	72
5.7	Function Failure Impact (FFI) and the Subsequent Reduction In Risk (RIR) Results	76
6.1	Functional losses found by the original FFIP simulation.	98
6.2	Example set of new failures investigated by adding the FSR logic to the FFIP simulation.	99
7.1	Safety Function State Reasoning Rules	106
8.1	Comparison of the Three Methods for Fault Simulation Clustering	115
8.2	Relational Matric for identifying the distance between function states.	124
9.1	The Output of Function Failure Logic for the Simulation in Figure 9.7	143
9.2	Mapping of System Functions to Components	145

LIST OF APPENDIX FIGURES

Figure Page

A.1 A SysML block definition diagram illustrating the decomposition of the electrical power subsystem. 179

A.2 A SysML internal block diagram illustrating the connections (structure) of the electrical power subsystem. 180

A.3 A SysML internal block diagram illustrating the functional structure of the rocket subsystem. 181

A.4 A SysML internal block diagram illustrating the connections of components in the design of the rocket subsystem. 182

B.1 Configuration Flow Graph for Electrical Powers System Alternative Design #1. . 184

B.2 Low-level Functional Model for Electrical Powers System Alternative Design #1. 185

B.3 Configuration Flow Graph for Electrical Powers System Alternative Design #2. . 186

B.4 Low-level Functional Model for Electrical Powers System Alternative Design #2. 187

B.5 Configuration Flow Graph for Electrical Powers System Alternative Design #3. . 188

B.6 Low-level Functional Model for Electrical Powers System Alternative Design #3. 189

Chapter 1: Introduction

The defining aspect a complex engineered systems is the achievement of a set of high-level behaviors that would be impossible to complete with simpler machines. From single complex systems that launch, land and operate robotic science laboratories on other worlds to the large-scale power and transportation systems which fuel and carry billions of users daily; modern technological development have made "engineered systems" a practical and ordinary part of life all over the world. For a variety of reasons, public dread regarding failures of these complex systems is greater than the concern over simpler machines. Thus, the trust the public must place in the engineers who design, build, and operate these systems is great. Validating this trust is the driving motivation of this and all other work in assuring the design of safe and reliable complex systems.

1.1 Complex Engineered Systems

Modern technologies involving any organized combination of software, hardware, people, facilities, and procedures are refereed to as an "engineered systems. These are distinguished from systems which are observed to occur naturally (e.g., ecosystems). Further, one system can often be composed of multiple subsystems and many systems can interact to form a system of systems. The challenges of defining, developing, and deploying systems has led to the creation of the systems engineering discipline. The purpose of designing a system is that a desired function cannot be accomplished by a single element. For example, a pair of scissors is not a system because the act of cutting could still be accomplished with only half the sheers (though not as well). Further, many of methodologies and tools developed for systems engineering provide little benefit compared to the effort involved when applied to simple products like coffee makers or hand-held drills. Therefore, there exists a property of a system with interacting components that justifies using these advanced methods. This property is complexity.

Since the Industrial Revolution technologies have been growing in complexity. While there are various subtle difference in the way complexity is evaluated [1], the most con-

sistent definition is related to the amount of interaction between the elements which compose the system. A system with many components can still be simple if the interactions between component are very limited. However, if a function of a system can only be achieved through the interaction of multiple components covering different technology domains, this system is considered a "complex engineered system." A specific form that incorporates highly integrated software and hardware networks is referred to as cyber-physical systems [2]. To achieve specified objectives, cyber-physical systems (as well as all other types of systems) implement a concept of emergent behavior. That is, behavior of elements interacting produces a system level behavior that is more complex than an additive behavior of the elements. For comparison, consider the braking system in a car. The pedal depression causes an increase in pressure in the brake lines which then pushes brake to the wheel. This is an example of simple or additive behavior. However, at a higher-level, the driver, antilock brake controller, and the sensors and hardware form an interacting loop. The overall behavior of the system is defined by this feedback loop. The result is a high-level stopping behavior that cannot be achieved without the interacting component behaviors. However, undesired emergent behavior is also one of the most significant challenges for system design.

"Systems Thinking" is a unique approach developed to design, analyze, and implement systems based on this concept of emergent behavior. Checkland [3] attributes the foundation of systems thinking as developing from the science of biology in the early 17th century. Advancements in biology showed that organisms were complex and that they were more than the sum of their parts. Living organisms were shown to have behavior and functional capabilities based on the behavior of their basic components and the interaction of those components. Systems thinking began to be formally applied to technologies after World War II, specifically in the development of the Atlas Inter-Continental Ballistic Missile [4]. In 1948 the RAND corporation was set up to use systems level approaches to solve US Air Force problems [5]. The systems analysis techniques developed at RAND spread to other research labs like Willow Run Research Center at the University of Michigan and AT&T's Bell Laboratories. The work generated from Willow Run resulted in the first text dealing with systems analysis of large systems by Goode and Machol [6]. A few years later the seminal work by Hall presenting the first integrated method of large scale system analysis and synthesis was published based on the research in communication systems at Bell Laboratories [7]. These methodologies were also used

extensively for weapons systems and led to the US Department of Defense creation of military standards in 1974 to define and guide the systems engineering processes [8].

Throughout the systems engineering discipline, the most common description of system deployment is as a "V", where high level requirements are refined to the discipline level subsystems implementation while validation and verification occur at each subsequent step of refinement [9, 4, 10, 3, 11, 6, 7, 12, 13, 14, 15, 16, 17]. The traditional approach to systems engineering, described in Figure 1.1, subdivides and refines the system into different technical domains and then integrates and validates refined subsystems designs. The challenge posed by emergent behavior becomes the most critical element for systems engineers at the validation stage because they must identify potential negative interactions and eliminate these before the design hits the factory floor. As technical systems have become more advanced the interactions, and thus the complexity, of systems has grown exponentially. However, the standard approach to systems engineering has not substantially changed. Numerous tools have helped facilitate the process but there has not been a significant change in the design and validation process to address the growth in complexity of modern system.

This lack of critical change in the systems engineering process motivated a recent program in the Defense Advanced Research Project Agency (DARPA) called Adaptive Vehicle Make [18]. The main motivation for this program is a perception in the defense community that while the time required to move from concept to build has maintained constant with the growth of complexity is some areas such as automative or integrated circuits, the same cannot be said of aerospace systems. In contrast this time has increased linearly with the exponential growth of complexity. Further costs have also increased similarly leading to Norm Augustine's famous (and overly dramatic) "Final Law of Economic Disarmament", projecting the cost growth into the year 2054 to indicate that the entire defense budget would purchase just one aircraft [19]. Combining these two observations imply that the cost and design cycle time growth with the traditional approach to developing systems is not sustainable for the continued advancement of complex technologies. Thus the goal of this Adaptive Vehicle Make program is to identify and develop novel design, analysis, and production methods for application to complex engineered systems to significantly reduce the design cycle and resulting high costs. This goal also motivates this research in that considering safety and reliability early in the design stage will be an enabler of this novel approach to complex system

design. In essence, traditional approaches to validating systems for safety and reliability are a significant contributor to the long design cycles. Therefore, this work sets out to develop a novel framework to generate concepts from a safety perspective and perform analysis of complex systems early in the design that enable risk and safety-based design decisions.

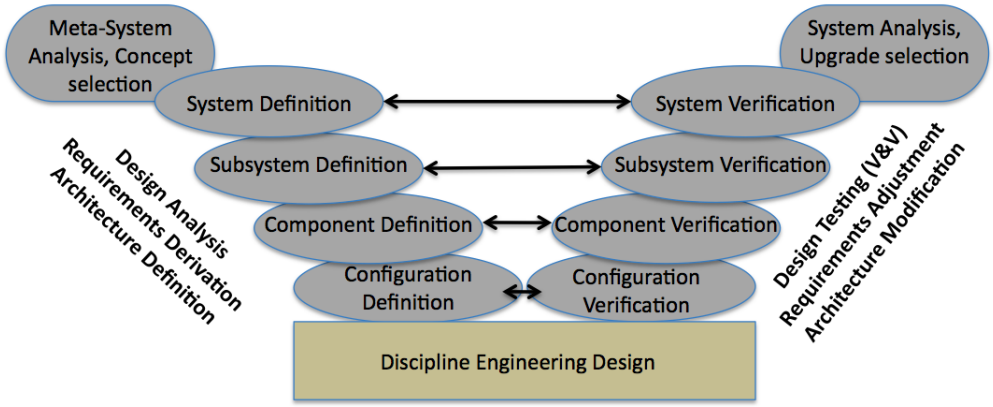


Figure 1.1: Top-Down System Engineering approach represented as a "V" diagram (after Buede in [11]).

1.2 Overview of the Work

The overall objective of this work is to enable a risk-based design decision tool for complex engineered systems. Specifically, this work looks to address three areas. First, generating concepts and designs from a safety perspective where hazard-mitigation is explicitly and formally linked to components and software algorithms using models. The second phase uses this design representation and develops reasoning to identify the impact of critical events in terms of functional reliability and safety. Finally, a set of results clustering analysis methods are used to enable designers to make design decisions.

This dissertation is divided into several parts. First, Chapter 2 provides an overview of the traditional approach and state of the art regarding how safety and reliability are

evaluated in complex system design. The second part of this dissertation spans Chapters 3-9 and provides the main contribution of this work. Chapter 3 summarizes the entire safety-based design framework and the contributions and limitations of this framework. The detailed parts of this framework are provided in Chapters 4-8. Chapter 4 details the safety-based design method referred to as phase one above. The second phase of scenario identification and system response to those scenarios is described in Chapter 5-7. The third phase of the framework is described in Chapter 8 on how to understand the scenario response results in order to enable design decision making. Chapter 9 provides some additional insights gained from applying these analysis methods to complex systems. Finally, Chapter 10 is the conclusion of this work.

1.3 Terminology

It is necessary to use several terms that have multiple meanings in different domains. Therefore, the table below provides the definitions for many terms as they are understood in this work. While all these definitions are supported by a larger body of research, this list is not intended to be authoritative outside the contexts of this research.

Table 1.1: Terminology Used Throughout This Work

Term	**Definition**
Behavior	The set of expected and unexpected interactions that a component or algorithm has with it's environment. In this work a discrete approach is taken where behavior can be divided into modes and transitions, though continuous behavior may exist within a mode.
Complex Engineered System	A technology that incorporates multiple interacting elements to perform a desired function through the use of emergent behavior.
Component	The lowest level element (building bock) considered in this framework. It has a behavior and this research will describe either software algorithms or physical hardware as components.
Cyber-Physical System	A special category of complex systems that utilizes a closely coupled networks of computational and physical elements to achieve a set of desired functions.
Emergent Behavior	Behavior that is not generated or modeled by a single component that exists when sets of different behaviors interact with one another.
Event Scenario	A change or set of changes in a system. This work uses a discrete time analysis and thus events also have a sequence. In this research an event scenario has a set of time steps where, during simulation, one or more component mode changes is triggered and/or a flow value changes.
Failure	A undesired behavior at the subsystem or system level
Fault	A discrete mode of a component with a specified behavior that is different than the nominal behavior.
Flow	Energy, material or signal that connects components and is acted on by functions.
Function	The intended behavior or actions of a system.
Function Health	The quality of the operation of the function with respect to its intention. Defined as *Healthy, Degraded, Lost,* or *Inoperative (No Flow)*
Hazardous State	The system state where from which an event or scenario may lead to a mishap state.
Mishap State	The system in an accident or undesired event.
Reliability	The property of a system or component to be available for intended operation.
Safety	The property of the system to resist the transition from hazardous state to mishap state.
Safety Function	The part of the system which performs the actions which inhibt the system state transition from hazardous to mishap.

Chapter 2: State of the Art

Complex systems such as aerospace platforms and power generation facilities exhibit complex forms of failure. These safety-critical systems undergo rigorous testing and validation to assure safe operation. However, highly publicized, costly, and sometimes fatal accidents still occur, usually as a result of some failure that affects more than one type of technical subsystem. Because these systems can experience multiple and cascading failures that propagate through the system, validation through testing can only provide probabilistic assurances of safety. Further, even when a potential failure is detected through testing, the redesign and revalidation process can be extremely expensive. For this reason, a growing field of research has developed to move safety and risk analysis into the early design stage to achieve "safe system design." To understand these methods we must have a core understanding of the nature of failure in systems.

A fault is an undesired behavior in a component or set of components that can lead to losses in system functionality. When those losses occur the system, experiencing some kind of hazard, can fail to prevent itself from being in an unsafe state. This is the simple model of failure and safety that forms the basis of this research and is illustrated in Figure 2.1. The simplicity of this model disguises the challenges faced by design analysis methods in moving from one block to the next. In general, two distinct categories of methods exist for addressing these challenges. First, reliability-based approaches focus on the identification of faults and use a variety of means to determine the impact and likelihood of that fault occurring throughout the system lifecycle [20]. Most of these methods are used (and often developed) outside of the systems engineering context as well. In the reliability perspective "expected risk" is evaluated as a single value based on the consequence of a fault multiplied by the likelihood of it occurring at a certain time. The probability of a fault occurring is modeled as one of a few distributions with respect to time (typically Weibull or Poisson distributions). Traditional approaches to reliability-based methods and their strengths and weakness are detailed in Section 2.2.

In contrast to reliability-based methods which are fault-centric, hazard-based methods are system state centric. Hazard-based approaches focus on an undesirable system

state and attempt to identify paths of reaching that state and the likelihood of that path [21]. In the hazard perspective the "risk" of a system moving from a hazardous state to a mishap (or accident) state is a distribution based on the probability of paths to reach that state and the level of severity of the accident state. For example, the Farmer curve is a carefully created limit line comparing the amount of radioactive material released in an nuclear reactor accident and the time frame between release incidents [22]. The risk of the hazard is distributed over the severity and likelihood of the release and the limit line indicates a threshold of unacceptable risk. Methods and tools for hazard analysis are discussed in Section 2.3.

The goal of this work is to enable design analysis and decision making based on the risk of a system experiencing a mishap state. Further, for this analysis to be useful for making design alternative decisions, the connection between the mishap state transition and the sequence of faults and interactions which led to the transition must be made explicit. Therefore, this work leverages benefits from hazard -based and reliability-based perspectives and incorporates them into a system design process. The following section will detail the traditional approach to system design to provide the context of this work.

2.1 System Design

As mentioned in the Section 1.1, the traditional model of systems development follows the "V" diagram (see Figure 1.1). The left side of the "V" diagram is the design development and has been divided into stages by many authors. This research utilizes the framework presented by Pahl and Beitz [23], which categorizes the phases of design as: 1) Requirements analysis, 2) Conceptual design, 3)Detail design, 4)Validation and testing. This research specifically focuses on design refinement and evaluation at the conceptual design stage. Validation at any of these stages requires an analysis and evaluation method. Trade studies, a method of trading parameters between abstract subsystem representations, have been shown to be an effective tool for the design of complex systems [10, 11]. Trade studies allow for design space exploration where many different systems designs can be evaluated for multiple metrics such as cost, reliability, or risk [24]. One motivation for this research is to provide an evaluation metric based on safety to guide early design trade-offs.

A key enabler of conceptual design validation and trade studies is the model-based

approach [25, 11, 14, 26, 27, 28]. Model-based systems engineering (MBSE) allows the focus of system design to move from textual descriptions to reusable, computable, and verifiable models. To realize the goal of complete MBSE design and validation, domain specific and general modeling languages have been developed to represent the requirements, structure, and behavior of a system [26, 29, 30]. One important aspect of a modeling language is to provide a schema for demonstrating the relationship of how a system's structure exhibits behavior to satisfy requirements. The language provides a formalism for representing the milestone in the system design refinement process.

The process of complex system design is an extension of traditional design methods and techniques. The unique challenges presented by complex systems for design methodologies is scaling to large numbers of components and designing for subsystem interactions. Traditional approaches focus on iterative refinement from abstract functions (that satisfy requirements) to detailed subsystem and component embodiment. This general approach to system design, where the concept of function is used to identify components is called "functional design" [23, 31, 32]. This functional design process, which takes place in the conceptual design phase, begins with a high-level function and decomposes it into a function-flow block diagram to model how flows of material, energy, and signal are transformed by functions or processes within a specified environment. The process of functional decomposition continues until functions can be embodied by components. A design at this stage is a functional model with one or more cyber-physical architectures. In contrast to this design refinement, Suh [1] presents a cyclic approach where functional requirements are satisfied in components that further generate subrequirements. The Contact and Channel Approach [33, 34] also presents system functionality as tightly linked to the system structure. In this approach a function is achieved through two or more working surface pairs (a geometric interface) and a channel and support structure (the flow between those interfaces). Implementing this method as a software tool provides the capability of identifying new functionality required based on the chosen physical solution [34]. Function-Means Trees [35] demonstrate a conceptual way to show multiple solutions to each function providing an overview of the solutions space. The process of determining solutions to requirements is design synthesis [36, 37]. Methods in design synthesis attempt to search through past solutions [38, 39] and solutions principles [40, 41]. Much recent research in functional design methods has focused on novel ways of generating functional models [42] and algorithmic means of generating system

architectures from functional models[43, 44].

2.2 Reliability Methods and Tools

Reliability analysis is the study of component's and processes' tendency to fail or remain operational [20]. This is the time-variant reliability of behavior in contrast to the time-invariant structural reliability dealing with uncertainties in the composition of the system [45]. This latter form of uncertain variance will be addressed later in this method and reliability in the remainder of this work refers to the time-variant approach. System reliability is an aggregate property based on the reliability of all its members. Therefore, a system will be only reliable as its least reliable part necessary for successful operation. Reliability is quantified in terms of failures with respect to time, such as Mean Time to Failure (MTTF), or Mean Time Between Failures (MTBF). Uncertainties in failure mechanisms, component operation, and natural variability lead to representing the probability of a failure as a distribution with respect to time. A components "failure rate" is based on this distribution. For example, many wear-based failures are modeled as following an exponential distribution with respect to time with the probability density function in Equation 2.1. The failure rate is λ and the MTTF is $1/\lambda$.

$$f(t) = \lambda e^{-\lambda t} \tag{2.1}$$

A variety of methods have been developed to address reliability concerns. The following sections will discuss traditional approaches to reliability assessment which occur at the validation stage as well as novel methods in the literature for design stage reliability analysis.

2.2.1 Validation Stage Approaches for Reliability Analysis

Most reliability analysis methods that are commonly used in industry are applied at the validation stage to provide assurance that a design meets a set of reliability or risk requirements. These methods require mature designs where the detailed component failure information and likely fault propagation path can be identified from testing, historic data, or expert knowledge. For example, computationally simple methods such

as aggregate failure rates [46] or component property distributions [47] are data-driven and require a well defined design to provide meaningful results. In practice there are two main analysis tools that form the basis for other tools and methods, namely Reliability Block Diagrams (RBD) and Failure Modes and Effects Analysis (FMEA).

Reliability Block Diagrams (RBD) are a graphical method of representing component failure rates and the dependency of components in the system for operational success [48]. RBDs represent component failure rates as blocks connected in either series or parallel with one another. Blocks are in parallel if the system can reach success through an alternate path (such as in the case of redundancy) and in series if each operation is necessary for system success. By following simple combinational rules, the entire RBD diagrams can be simplified to a series structure where the potential for reaching success (the reliability of the system) is the product of the rate of failure in each block. RBD forms the basis for more sophisticated techniques and analyses. Dynamic RBDs where developed by Distefano and Xing to provide availability information for non-static systems [49]. Further, RBDs can be converted into Fault Trees (to be discussed in Section 2.3) for a system level analysis [50]. Finally, for more complex systems with dynamic behavior it is necessary use special techniques to solve for the resulting reliability or availability. Most common is the use of Markov analysis to identify the reliability of complex and multi-state systems [51, 52]. These methods provide a well accepted evaluation of reliability as long as the two foundational assumptions hold true. These assumptions are that the failure rates for components are accurate and that the failures are independent (allowing the probabilities to be multiplied). The first assumption can be violated when historic data is used which might not reflect the component in situ or within the same operating context. The second assumption is often violated in catastrophic complex system failures when one failure leads to another.

The second well accepted method of reliability analysis, Failure Modes and Effects Analysis, is a method that systematically examines individual system components and their failure mode characteristics to assess risk [53]. The output of an FMEA is a table relating low level faults and their impact to the system. FMEA starts with decomposition of the system into subsystems and finally into individual components. Failure modes for each component are then examined to separately identify and record the effect of each failure at the component level and then at the system level. FMEA can be a tedious exercise and can fail to accurately capture all the potential effects of single

failures [21]. While a variety of tools exist to automatically generating FMEA output tables [54, 55, 56], they either require explicit modeling of the fault propagation path or the path is assumed to follow the nominal component architecture. An extension of the FMEA approach is to add a Criticality Analysis, which uses expert knowledge to quantify the failure mode effects according to system level hazards (such as loss of life, mission failure, etc.). While a functional FMEA can provide the format for representing the effect of the loss of functions [57], FMEA is not a useful tool in the early design stage of complex system based on the need to know failure propagation paths and the inability to represent multiple or cascading faults.

2.2.2 Design Stage Approaches for Reliability Analysis

In practice the role of reliability in the design stage is managed through highly reliable design practice. That is, well formed and used methods such as the Taguchi [58] or Design for Six Sigma [59] provide a structured design process to increase reliability by reducing variance in the design and manufacturing of products. However, there is a perceived need in the literature for developing tools and methods to enable design for reliability addressing the component behavior variance.

Early work to move reliability assessment into the conceptual design stage focused on: 1) Describing the nature of faults in the conceptual design perspective [60], 2) How those faults may affect the performance of other components in the system [61, 62, 63]. These reliability methods took a qualitative perspective on failure. In contrast, quantitative methods use descriptions of fault probability to provide a risk assessment at the early design stage [64, 65, 66]. In order to provide an assessment at the concept stage failure was viewed in terms of its effect on function.

The Function Failure Design Method (FFDM) [67, 64] was one of the first methods to formally connect faults to functional losses with the goal describing the space of potential system failures. 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 functions. Based on historical failure data for these types of components, it is then possible to establish likely failure modes for a given function. Because historical fault propagation data is configuration specific, the FFDM method is limited to single fault impact analysis. Additionally, Grantham et al. developed the Risk in Early De-

sign (RED) method of formulating functional-failure likelihood and consequence-based risk assessment classified as high-risk to low-risk function-failure combinations [65, 68]. These methods have limitations similar to those seen in traditional approaches in that historical single fault failure data may not be applicable to new designs or multiple dependent faults. Further, these methods do not consider the impact of multiple failures and failures that result from system interactions. To overcome this limitation, other work has focused on including the propagation of failure in the analysis.

Krus and Grantham present a Failure Propagation Analysis Method [69] developed as a direct extension of the FFDM and RED methods to capture the 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 introduced by Clarkson et al. [70] to apply to the functional level. Finally, the Function Failure Identification and Propagation (FFIP) framework [71, 72, 62, 73, 74] was introduced 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 to identify failure propagation paths by mapping component failure states to function 'health'. This approach uses simulation for determining fault propagation and fault effect, providing 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. These methods are bottom-up approaches that investigate the contribution of the component state to the system state. Therefore, these methods can provide designers with an evaluation of the likelihood of occurrence [65], the propagation path of faults [73, 69] and the effective functional reliability [72] of a design.

2.3 Hazard Mitigation Methods

In contrast to the reliability-based methods discussed in the previous section, this section details the hazard perspective. These two perspectives are not conflicting and often overlap in the areas considered and the tools used. The distinction being made for this research is the methodology starting point. The starting point for the reliability methods discussed previously was on component failures. From this point the methods provide a variety of tools for exploring the impact and significance of those failures. In contrast, the

starting point for hazard-based approaches is the undesired system state. As described in Figure 2.1 and found in literature [75, 21], hazard-based techniques rely on the underlying model that a system transitions from some hazardous state to a mishap (accident) state based on a set of initiating mechanisms. Therefore, the goal of hazard based approaches is to first identify the potential hazards and then systematically identify the mechanisms and sequences of events which can cause the system to transition to a mishap in the presence of those hazards.

2.3.1 Validation Stage Approaches for Hazard Analysis

Fault Tree Analysis (FTA) [76] is performed to capture the potential paths that lead to an undesired system state. A Fault Tree (FT) is a graphical way of representing the logical sequence of events using logic gates (AND, OR, etc.). Using this approach, possible event paths from failure root causes to top-level consequences can be captured. These fault paths can incorporate probabilities to identify the likelihood for a high-level event to occur assuming either fault independence or known fault correlation. The main objective of performing an FTA is the identification of cut sets, or the set of events which must occur in order for a top-level event to occur and a variety of algorithms have been developed to find these automatically [77]. A common criticism of FTs is that their size grows rapidly with complex systems. For example, FT may have hundreds of gates and even more paths for a complex system yet if that system is in a different operating use case or time that FT may no longer represent the actual system. Since the failure domain is represented using events in FTA, low-level component interactions and dynamics leading to failure are only considered informally, during expert identification of event-consequence relationships. Formally capturing component interactions and system dynamics of hardware-software systems is however crucial for supporting design decisions during early concept development of complex systems.

Probabilistic Risk Assessment (PRA) [78] is a method used for quantification of failure risk [79]. PRA combines a number of fault/event modeling techniques such as master logic diagrams, event sequence diagrams and fault trees, and integrates them into a probabilistic framework to guide decision-making during design. Some research has attempted to extend PRA to include event/behavior simulation into the analysis as demonstrated [80]. As with the reliability methods mentioned previously, this extension demands a

fully specified system model as part of the analysis. Such detailed, high-fidelity models of hardware-software systems, however, are not available during conceptual design.

2.3.2 Design Stage Approaches for Hazard Analysis

There are a few efforts to move system hazard analysis into the design stage. These methods rely heavily on expert review and utilize many of the same techniques as validation stage approaches but utilize the less refined design stage information. Preliminary Hazard Analysis is a method to guide safety engineers to identify potential hazards, the mechanisms and their impact [81]. This is done in the early design stage using functional diagrams, reliability block diagrams and indentured equipment lists [21]. By using a hazard checklist and a worksheet for each identified function or equipment, engineers are guided to identify and record any potential hazard. Further, expert opinion can be used to estimate the likelihood of the hazard mechanism.

Dulac and Leveson proposed a systems theory-based accident model which accounts for mishaps as occurring through violations of safety constraints and unsafe control actions [82]. This model forms the basis for a design stage hazard analysis and safety guided design process, the System Theoretic hazard and Process Analysis (STPA) [83, 84]. Using this approach designers identify scenarios based on the generic ways that the control structure safety constraints can be violated. This approach enhances the design method by guiding designers to identify design changes that reduce the likelihood of those constraint violations.

2.4 Summary

The traditional, ad hoc approaches to system design has been effective for the last 50 years. However, the growing complexity of modern systems and the high cost of late design-stage validation have motivated a need for methods and tools for early evaluation of system properties. Model-based design methods enable this by allowing designers to communicate and evaluate properties and attributes. A functional approach to system design has been proposed by several researchers. These approaches enable the analysis of a system in terms of the how well the system executes the desired actions.

Two approaches to assessing failures in systems work towards different goals. Meth-

ods to evaluate and improve system reliability focus on reducing the impact and likelihood of component faults. In general, reliability methods require detailed design information, such as fault propagation paths, to provide meaningful results. Further, methods based on the statistical occurrence of an event often require assumptions of independence and cannot address multiple or related fault cases. For this reason, reliability and fault analysis typically occur in the later design validation stage.

Alternatively, research from the safety perspective starts with the undesired system state and applies tools and methods to find the potential causes and paths to reach that state. In general, these methods can be applied earlier in the design stage but are limited to guiding safety engineers to consider potential scenarios and assume that cause likelihood cannot not be determined due to the large set of potential paths to the top-level mishap.

Therefore, this research serves as bridge between component reliability perspective and the safety-based hazard approach through the simulation of low level component behavior and reasoning about the top level safety property of the system.

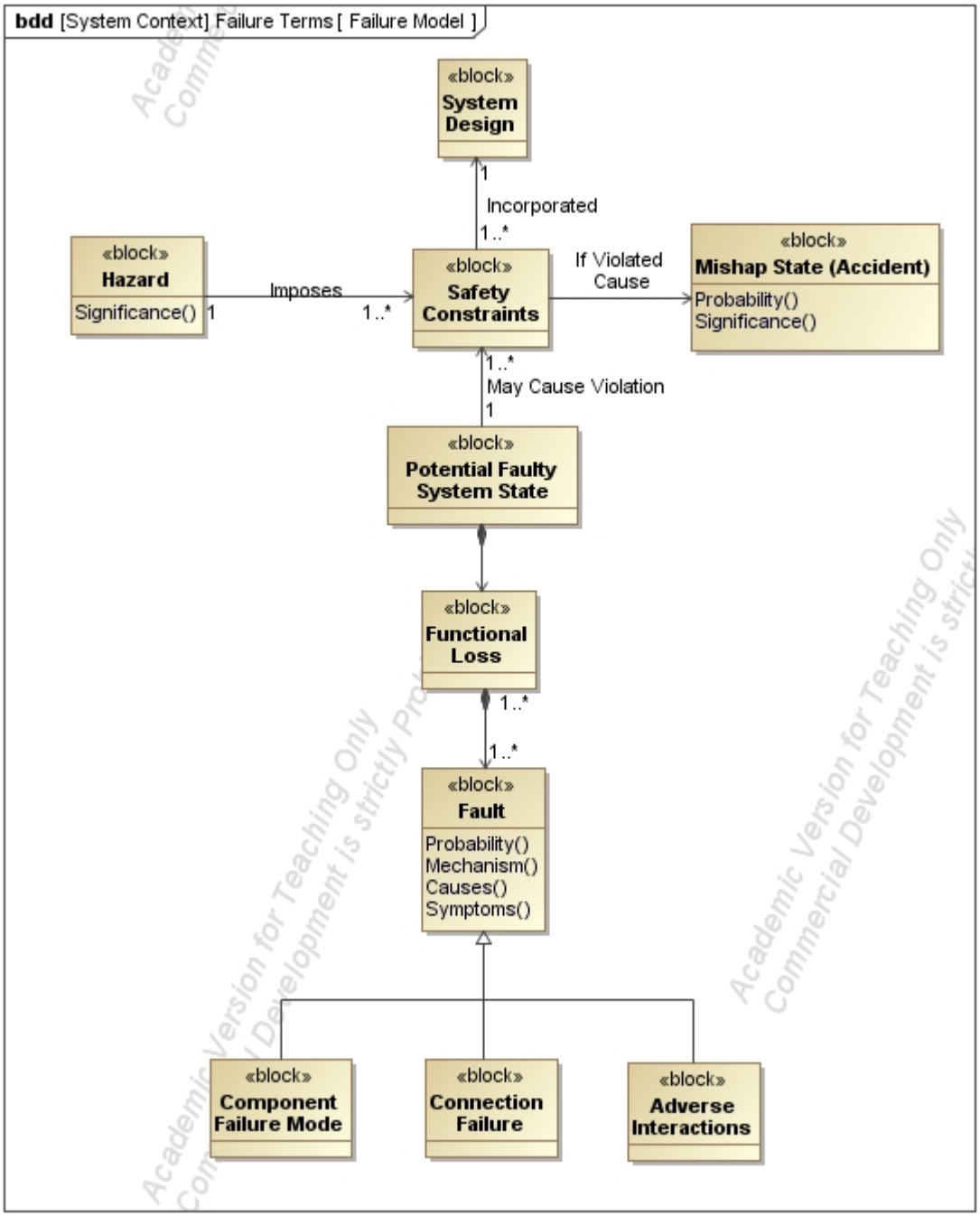


Figure 2.1: A model of how failures lead to unsafe system states.

Chapter 3: The Safety-Centric Design Decision Enabling Framework

Mishaps and accidents in complex engineered systems have incredibly high costs and the public awareness of these accidents often leads to long-lasting negative opinions of the organization or technology. Recent accidents such as the Fukushima nuclear reactor leak, the Deep Water Horizon oil rig fire, and the loss of NASA's Columbia Space Shuttle underscore the significant cost and public shock that these accidents entail. Despite high public awareness, significant accidents in complex systems are rare. What makes these failures rare is the rigorous safety review and assurance process these system undergo after the detailed design phase. Normal accident theory stipulates that due to the complexity of modern systems, specifically unintended interactions and tight coupling lead to an inevitability of accidents in these systems [85]. Therefore, the growing complexity of these systems has made this validation process long and expensive. Further, in this approach safety and reliability are requirements that the systems must meet rather than drivers of design process.

This dissertation presents a framework to enable safety-centric design decision making. Specifically, this dissertation presents a method of explicit inclusion of safety into the earliest stages of design, an analysis approach where component failure modes can be mapped to potential system hazards, and a means of enabling the designer to explore the design space from a safety perspective.

Application of this work presents several significant advances to the fields of safety engineering and design. First, the system property of safety is incorporated into the model-based functional design framework. Secondly, this work presents a simulation-based reasoning approach that allows designers to move from low-level fault causes to system level accidents in a way not possible with traditional risk and reliability methods. The method developed in the following chapters forms the basis for design decision making with respect to safety and functional robustness. Extending this method into a multi-objective decision-making framework will provide designers the ability to explore, select, and validate designs with a lower potential for failure at a significantly reduced design cost.

3.1 Overview

This research is composed of three parts. The first part develops a model-based approach for evaluating safety. Specifically, the concept of a safety function which inhibits the system state transition from hazardous to mishap is developed and connected to the functional design method. This process is presented in Chapter 4. The second part of this research explores the behavior simulation approach to fault analysis in complex systems. In Chapter 5 we introduce a method of fault simulation that allows for the quantification of the functional impact of critical scenarios. One understanding of "complexity" is the unknown or unexpected interactions between components of a system. Chapter 6 provides a method of identifying some of the unexpected connections between components that only occur in certain failure scenarios. Chapter 7 links the new safety function modeling approach to the fault simulation approach of the previous two chapters completing the components of the analysis phase. As an addition, Chapter 9 presents the use of part of this analysis approach to explore the design of a large complex system, namely a boiling water nuclear reactor. Finally, the third part of this research aims to convert this analysis into an assessment of safety and functional robustness of a design. However, this cannot be reduced to a single objective and Chapter 8 presents three methods for grouping the results of multiple simulation results to provide the designer with the "big picture" perspective with respect to safety.

3.2 Methodology Use Case

The overall objective of this work is to move the validation of system safety requirements to the early design stage from later prototype or physical testing stages. This dissertation presents a set of tools to be integrated into the design of complex systems. Assuming a novel system design, the process is as following: First, designers identify the requirements for their system including functional performance, and safety requirements. Next a preliminary hazard analysis is conducted to identify potential unsafe transitions states the system may take. Here the first component of this dissertation is integrated into the design process. Using these identified transitions and the functional requirements, a system concept is developed following a functional-design approach for implementing principal functions with components and safety functions with control structures. This

interrelated control-structure and component architecture represents a system concept. The next stage of this dissertation is then incorporated by implementing a critical failure analysis of the design. This is done by identifying the generic components of the design and creating a system simulation based on qualitative behavioral models of components that contain both nominal and failure mode behavior descriptions. Following this, a set of critical scenarios can be tested with that system simulation and the impact to the operational state of the principal and safety functions using a set of logical rules. The result of testing a large set of scenarios provides designers with an assessment of the quality of that design with respect to functional reliability and safety. One approach to design is to select between many alternatives. This could be accomplished by comparing design performance between two concepts or through defining the envelop of allowable performance. The third aspect of this research was developed aid both of these approaches. By clustering the results of multiple failure scenarios a risk profile for a design provides a quantitative way to compare two different concepts. Alternatively, by exploring scenarios with similar functional effect and identifying the high-level failure behavior of the system, designers can narrow-down acceptable behavior (with respect to safety) and the components responsible for moving the system out that behavior envelop. Additionally, this research provides a means of exploring the modeling and simulation sensitivity to that envelop of performance. The end result of using these tools is an assurance that the conceptual design meets safety requirements. The detailed subsystem design effort can use this as a guide of developing their respective subsystems. If detailed design identifies behavior that differs from the abstract component behavior representation then this schema can be updated with more precise component behavioral models. After this proposed analysis a system performance evaluation should also be conducted to validate as many of the performance requirements as possible.

The approach presented in this dissertation is based on simulating system nominal and faulty behavior to identify risk. Detailed mathematical models such as finite element analysis (FEA) can describe some aspects of a components behavior. However, large systems with many components restrict detailed behavioral modeling with FEA based on computation difficulties. Therefore, this work uses an approximation of behavior based on qualitative descriptions of the component nominal and faulty behavior. Using this abstraction approach, computational concerns begin to emerge when considering interaction of 30 or more components that have an average of 3 discrete states. Since

complete exploration of the potential statespace is not necessary (not every component will fail in one scenario), a large set of scenarios can be evaluated in a short period of time. For example, Chapter 8 presents an analysis of 600 failure scenarios that takes approximately 30 minutes to simulate for a 30 component system. For the validation of larger systems two different approaches are proposed. First, subsystems that likely will not interact except through their nominal connections can be assessed separately. This approach can be seen in the separate subsystem analyses in Chapters 5 and Chapter 6. Alternatively, a system level analysis can be conducted using subsystem approximations. Chapter 8 illustrates a statistical approximation approach for capturing the functional failure behavior of a subsystem as discrete states. In this research the challenge of scaling the analysis to large systems is addressed through these two approaches.

The aim of this work is to assist designers in the automated, model-based analysis and validation of conceptual design of complex systems. It is the vision of this work that the methodology presented in the following chapters would be integrated into a multi-objective decision making framework. This methodology assumes that a team of designers can utilize a component model database to construct one or more system architectures and an integrated platform for modeling and conducting the analyses presented in this work. Designers attempting to make new designs using historic components as well as designers making incremental improvements could benefit from this safety-centric decision making framework. The method presented here follows a novel design paradigm. To implement a "one-off" design, engineers would need to start with identify the safety functions from the existing system rather than the preliminary hazards list.

3.2.1 Example System for Application of Methodology

To illustrate this methodology in a practical example this dissertation uses a spacecraft subsystem called a reaction control system. Reaction Control Systems (RCS) are a set of maneuvering thrusters that have been used on various spacecraft from satellites and shuttles to lunar and martian landers. These RCS thrusters provide a small amount of thrust allowing the craft to maneuver in space. As an example system for implementing this framework, we consider the early design of an RCS on a satellite. This example includes software control, an electrical power system for supplying the power to the computing hardware and the actuating the valves on the thruster system. These three

subsystems based on different technology types work together to achieve the system level function of attitude control.

3.3 Contributions and Limitations

3.3.1 Intellectual Merit

The main objective of this research is the development of a safety-based functional design and analysis framework. The intent of this framework is to guide the design of complex system from the earliest stages to gravitate towards highly reliable and overall safe designs. Risk and hazard analysis have independently worked towards this goal. However, the novelty of this research is the integration of design and analysis methods to incorporate both the low-level causes of failure and the high-level effect on system safety. Specifically, by modeling the safety properties and the functional properties, a well structured link is created between component states and system safety. This method provides a means of mapping component fault knowledge to system hazard knowledge. Through the identification and simulation of critical scenarios a design can be evaluated for safety and functional robustness. This work also demonstrates how the results from these scenarios can be clustered in different ways to provide designers with different views of their system.

The approach presented in this research offers significant advantages compared to traditional approaches for validating safety and reliability. First, this approach is implemented at the concept stage of system design. In contrast to traditional methods that validate a well refined design, this research allows early design selection to be guided by safety concerns. In contrast to methods that exist to address safety in early design this research is unique in using qualitative simulation of behavior to identify safety issues. Instead of providing a guide for designers to think about potential hazard causes this dissertation presents a simulation and reasoning method for exploring a concept design and explicitly capturing the paths of potential system failure.

3.3.2 Limitations

The current industry practice for safety and risk involve using expert engineers in lengthy safety design reviews. While the difficulty of this with highly complex systems and the cost motivate this early design analysis method, it is not the goal of this work to replace the knowledge and experience of seasoned engineers. The most obvious comparison of this work is to the traditional ad-hoc methods. Because is cannot be known what failures are identified or missed through an expert review process prior to analysis, it is not possible to formally describe the increased scenario coverage provided by this method. Instead, this framework is presented as an means to augment designer expertise.

Because this method centers around identifying the impact from simulated potential scenarios, there exits a finite limit to the number of scenarios that can be reasonably tested. It is possible to identify the percentage of the potential states covered using methods from verification and validation research. However, this is outside the scope of the current research.

Finally, this work revolves around the use of component behavioral models. Uncertainties and inaccuracies in these models have a significant effect on the analysis output. This research assumes that component failure modes can be identified a priori and that general behavior can be specified for those components in those failure modes. This research does not address insufficient or unpredicted component failure behavior.

Chapter 4: Safety Guided Design

The functional approach to design provides engineers with a conceptual level abstraction to effectively model, develop, and analyze a product or system. This is evidenced by the use of function-based methods in reverse engineering, one-off, and novel design research. The power and simplicity of functional modeling lays in the simplicity of purpose. A functional model serves to show only how a design achieves a given set of functional requirements. Only through additions to the functional representation can engineers begin to describe how a design will meet other types of requirements such as performance or safety. Further, most research in function-based methodologies has viewed satisfaction of these other requirement types as part of the detailed design stage. This leads to supporting the traditional approach of relegating safety to the design validation stage. However, recent work in systems engineering methods has focused on supporting a safety-centric design process [83, 84]. The perspective of these approaches is that safety should be a driver for design. Thus the objective of this chapter is to introduce a safety-centric method of developing a design based on the functional modeling paradigm.

Specifically, this chapter presents a method of representing the safety property of a system explicitly in the model-based design approach. Further, the function of achieving safety is mapped to the performance functions of the system. This chapter demonstrates a process of concurrently developing a system concept with safety and functional perspective. The end result of this process is a system architecture where components of the system are explicitly mapped to both the functions they perform and the role it plays in ensuring safe operation. The benefit to designers of using this approach is having a system representation that allows for analysis of critical events and off-nominal component behavior to identify potential losses in function and safety constraint violations.

4.1 Background

4.1.1 Models and Modeling Languages

Morgan and Morrison describe models as mediators between humans and phenomena [86] and Hodges defines models from a historical perspective as being "an object in hand that expresses the design of some other objects in the world" [87]. Further, models are an abstraction and an incomplete representation of reality [11]. The main purpose of modeling is to represent information and the relationship between pieces of information. This is generally done with one of two different approaches. System information can be represented in node connections or in block diagrams. The node connection approach uses the connection between nodes to represent system information. For example, the transformation that occurs from *state 1* to *state 2* is represented by the type of connection between the nodes for *state 1* and *state 2*. This is the approach used in Bond graphs [88, 89], Petri Nets [90, 91], and tree structures. Alternatively, system information can also be represented with block diagrams. Block diagrams represent system information with blocks and these blocks can have attributes to describe other pieces of related system information. To show a similar transformation as above, the change in *state 1* and *state 2* is represented by the *transformation* block. Block diagram approaches are used in families of languages like IDEF [92] and SysML [26]. The basic distinction between block and nodes diagrams is where information transitions occur, either at blocks or between nodes. Long presents a concise overview of the relationship between different common graphical representations in [93]. The three main categories or types of models are data models, process models, and behavioral models [11].

4.1.1.1 Data Models

The purpose of data modeling is to represent artifacts of information about a system and the relationships between those artifacts. Data models are used for defining causal rules in model-based reasoning [94]. Types of data models are also used to represent organizational structure in systems management [95] and can be seen in system design to identify the relationship between system inputs and outputs and basic system architecture [11]. The oldest form of data models are entity-relationship diagrams [96]. These simple diagrams connect two or more pieces of information (entities) with a marker describing

the relationship and a directed arc indicating the order of the relationship. For example, the entities *People* and *Money* are related by the arcs *Spend* and *Save*. These arcs are directed so that the model accurately describes people spending and saving money and not the other way around. For more complex systems, the data relationships are difficult to capture without nesting the information into classes. One implementation to this was presented by Harel in the creation of higraphs, which combined Venn diagrams and entity-relationship models [97]. In higraphs, entities can be shown to span classes to show shared relationships between classes and complex relationships more clearly. An alternative solution for representation is to assign attributes to data entities and to specify the relationship between these attributes and entities within a class. This is the approach found in Object Oriented Modeling [27] .

4.1.1.2 Process and Functional Models

The purpose of process modeling is to represent how to a achieve a desired purpose. The earliest from was presented by Taylor [98] as workflow models to represent the tasks required to complete a job, with an emphasis on efficiency. Data flow diagrams developed from a need to capture the relationship between processes in a system [99]. In data flow diagrams, activities or functions are connected together with directed arcs to represent the "flow" between these activities. Data flow diagrams also have terminals for representing the connection of the system to its environment. Function flow block diagrams (FFBD) are an extension of data flow diagrams and use the same verb-object graph syntax [23]. The significant addition of FFBDs is the inclusion of control constructs between connected functions to represent sequences of functions. N^2 charts were developed in the 1960s by system engineers to relate system functions, with a focus on the inputs and outputs of those functions [100]. Because branching connections are not used in N^2 charts, representation is done at the lowest level of functional decomposition and functions must be repeated as often as needed [11]. Thus a drawback for N^2 charts is the repetition of functions in complex systems. System engineers have also used this distinction of N^2 charts to allocate functions to components to minimize component interactions [100].

4.1.1.3 Behavioral Models

The purpose of behavioral modeling is to represent the dynamics of a system. The behavior described by these models is discrete, event triggered behavior. The solvable set of modeled behaviors caused by an event or state is a simulation. One of the first approaches developed was behavioral diagrams [101]. In behavioral diagrams, behavior is represented through a progressive hierarchal set of sequenced functions. Functions are connected to each other with control arcs that are annotated to identify the order of sequencing. Enhanced FFBDs overlay a control arc and are equivalent to behavioral diagrams [102]. State transition diagrams originated in the software system engineering field as a way of representing sequential finite state machines [103]. State transition diagrams model event-based, time dependent behavior by boxes (states) and directed arcs indicating allowable transitions. Unlike the previous representations, state transition diagrams do not focus on the system functions but on the events that trigger transitions. Statecharts expand state transition diagrams by allowing for hierarchal representation by using higraphs [97]. Petri nets are mathematically based representations designed to rapidly turn a model into a solvable simulation. Petri Nets consist of places, transitions and tokens. Tokens move through the Petri net at intervals and transitions are allowed to fire (move a token onward) when its proceeding places contain tokens [90]. Tokens are values for variables in the system. Later modifications to Petri nets include using colored tokens to allow for multiple transition timings [91] and stochastic transition firings [104].

4.1.1.4 Modeling Languages

The most generally used languages for system information representation are IDEF and SysML. The Integrated Computer-Aided Manufacturing Definition (IDEF) language was developed as part of a US Air Force program to increase manufacturing productivity [92]. The IDEF family of languages has several methods for representation. IDEF0 is used to describe process or functions, IDEF1X for data modeling, and IDEF3 for process and state transition modeling. The IDEF0 representation method represents the functional decomposition where *functions* are connected to *inputs, controls, outputs,* and *mechanisms*. IDEF0 can be seen as the combination of data-flow graphs and N^2 charts [11]. In IDEF1 and IDEF1X, data is modeled as classes containing entities with

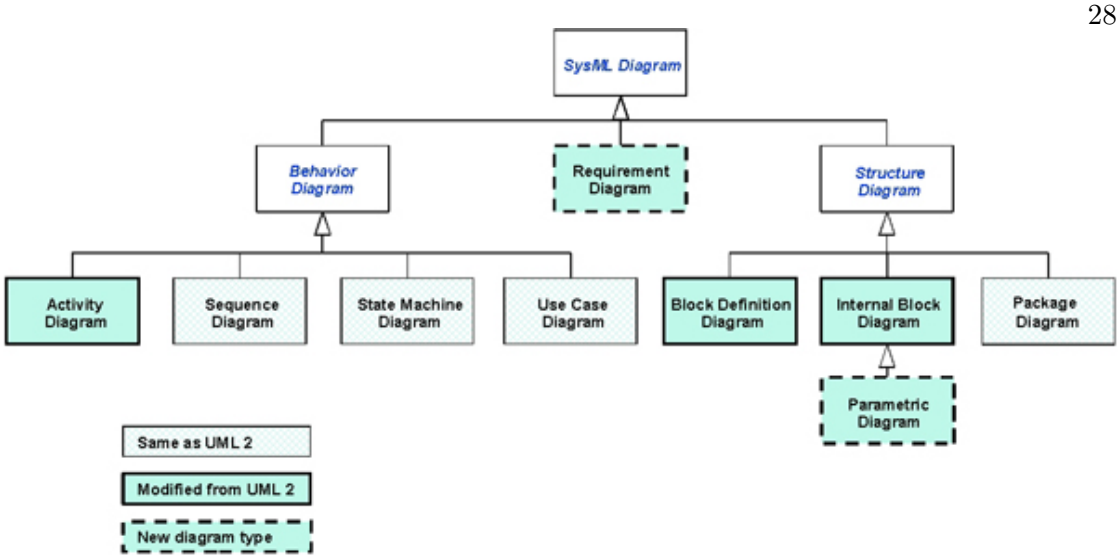


Figure 4.1: OMG SysML diagrams used for modeling a system, from the OMG SysML website: [105].

associated attributes. Different entities are related based on inheritance relationships, either parent or child connections, to show shared attributes.

OMG SysML^{TM} is a modeling language developed and managed by the Object Management Group (OMG) [105]. SysML is an extension of the Unified Modeling Language (UML 2), based in the software engineering field, that includes diagrams useful in the synthesis and assessment of complex systems. As with IDEF, SysML uses a set of block diagrams, each with a distinct purpose. In contrast to IDEF, each SysML diagram type is limited in scope and is designed to be complimentary to other diagrams [26]. In this way a system model in SysML is the collection of diagrams containing all the different aspects of system information. SysML uses nine diagrams under four headings to represent system information. Figure 4.1 from OMG's website succinctly describes the different diagrams and their relationship to one another. The use of software engineering vocabulary has been a hinderance to the wide-scale usage of SysML, however, the support of the International Council on Systems Engineering (INCOSE) has been significant in increasing the use of SysML in both academic research and industrial applications [106].

The U.S. Department of Defense Architecture Framework (DoDAF) is a standard representation used throughout the U.S. weapons systems. DoDAF presents system architecture information in three distinct views: Operational, System and Services, and

Technical Standards. The three views can be constructed using UML or SysML and thus, DoDAF does not represent a unique modeling language.

Other modeling languages have been developed within the context of systems engineering but with a focus on specific areas of research or methodologies. The Function Behavior Representation Language [107] and the Function Behavior Structure schema [108] are examples of tools used for the research done in subfields of systems engineering; system design and agent-based information management, respectively. Information management in the context of agent-based approaches to systems engineering led to the development of tools under the term Agent Oriented Language for capturing the dependencies of system information, such as AgML [109] and Brahms [110].

4.1.2 Function-Based Conceptual Design

As mentioned in Section 2.1, functional modeling is a process of concept refinement applicable to both product and system design. However, in the literature there is a variety of meaning and formulations in the concept of function and the role in which functional representation is used in the concept development phase of design. The following sections will discuss the different understandings of the concept of function in the literature and how those understandings lead to various approaches for achieving a product or system design.

4.1.2.1 The Concept of Function

As noted by Chandrasekaran and Jospheson [111], one of the consistent descriptive words of function in research literature is intent. In contrast to all other aspects of the design, function expresses the intention of the designer. The concept of function bridges the gap between human intention and physical reality [112] and represents the goal the designer has for the system [113]. Erden et al. provide a broad survey of functional modelings approaches [114]. While significant difference exist the main distinction useful for the current discussion is the view of function as object-centric or effect-centric.

In the majority of research literature, functional modeling begins with an object-centered view of functions [113, 32, 31, 115, 116, 117, 37, 118, 119]. That is, a function is part of the action that some component or set of components in the system performs.

This understanding of function leads to viewing functions as verbs acting on flows as nouns. Further, functional descriptions can be understood as input-output relationships of black boxes, leading to function block diagrams. In the case of Bracewell and Sharpe, their bond graph inspired approach has functions acting on both effort and flow [38]. These methods tend to follow the "no-function-in-structure" principle of De Kleer and Brown [118], in that all the functions the system performs should be captured in the functional model and no functionality is implied or inherited through the connectivity. Though some argue that this principle is largely unattainable in real systems [120], the principle as a goal serves to lead to a modular approach. This modular approach allows functions to be linked directly to components [121] and enables design decision making based on selecting between a variety of potential function solutions (components) [122].

In contrast to the object-centric view of function, the effect (or process) view understands functions as the set of physical principles used to achieve an intended goal. In this perspective the direct result of developing the functional model is the identification of the intended behaviors needed to achieve the desired functions. The Function-Behavior-State modeling scheme of Umeda et al. [123, 124, 112, 125, 126, 127], the Function-Behavior-Structure of Gero et al. [128, 129], Qualitative Process Theory of Forbus [130], and the functional ontology presented by Keuneke [131] are all examples of the effect or process view of function. One of the benefits of this approach is that, because behavior is the intermediary between functions and components, novel and redundant functionality can be identified through finding associated behavior [132].

Chandrasekaran and Josephson provide a clear distinction between the effect view of function and the device (object) view of function as well as a means of mapping between the two based on an understanding of the environment and the mode of deployment [111]. Following their example of a door buzzer, the device view of function is "to make a sound come from box2, when the switch in box1 is closed." Whereas the effect (on the environment) view of the function is "to provide a means by which a person at location 1 may cause sound to be produced at another location" [111]. Rather than being at different abstractions levels, these two perspectives describe the same system differently. The device view of function provides "the how it works," while the effect view provides "what must be achieved" (thus this is also called the teleological perspective). Figure 4.2 illustrates how these two views of function result in a different sequence of design refinement.

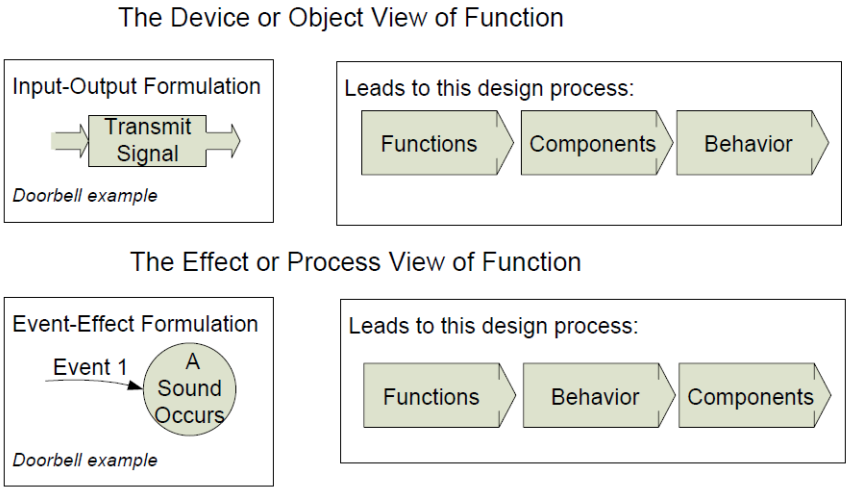


Figure 4.2: Contrasting two views of function found in the literature.

For this research, both views of function are adopted in different ways. The object or device view of function is adopted for software and hardware component architecture representations. This view of function is chosen because the behavior is derived from the components used. For nominal behavior, either perspective may arrive at similar system behavioral representations. However, the goal of this research is to evaluate how a design will behave in critical event scenarios involving one or more component failures. Functions are intermediaries between intention and reality and have no physical existence and thus cannot have failure modes. (Chapter 5 will discuss evaluating functional operating states which will be distinguished from component failure modes.) When a failure in a system occurs, it is the components which behave differently not the functions. From this need to evaluate the functional impact as a result of behavior of a system in critical scenarios, the device view of function is used for developing the system component connection architecture. However, this research proposes using an additional view of the system defined in section 4.2 as "Safety Functions." This terms is used to describe the property of a system to resist moving from a hazardous state to an accident state. The effect view of function is used for understanding how "Safety Functions" work in a system.

The key difference between these two understandings of function is related to the

formulation of behavior. As mentioned, the device view uses an input-process-output formulation. This research assumes that following some functional basis such as Hirtz et al. [133] and using a library of component behavioral models, a system simulation can be generated early in the design stage. Similar to the Functional Basis mentioned above, Keuneke developed a ontology based on an effect understanding of function [131]. Two of the 4 categories of function are adopted from that work to describe the types of safety functions. Namely, *ToPrevent* and *ToMaintain*.

4.1.3 System Theoretic Process Analysis

The Systems Theoretic Process and hazard Analysis (STPA) developed by Dulac and Leveson [82, 84] is a method for identifying and mitigating the causal factors of an accident based on the STAMP model [134] of accident causality. As discussed in Section 2.3.2, STAMP is a model of system failure that proposes that all systems move into accident states when safety constraints of the socio-technical control structure are violated. Using this mode the STPA method is a design process centered around representing the control structure of a system and identifying how any of the four types of unsafe control actions can occur. In [84], Leveson describes the four types of unsafe control actions as:

1. A control action required for safety is not provided or is not followed.
2. An unsafe control action is provided that leads to a hazard.
3. A potentially safe control action is provided too late, too early or out of sequence.
4. A safe control action is stopped too soon (for continuous or nondiscrete control actions).

The STPA methods begins with identifying the control structure to mitigate a potential hazard. Next, the method provides a generic list of control structure failure modes which might lead to one of the four types of unsafe control actions listed above. These failure modes (called causal factors in [84]) are illustrated in Figure 4.3. Finally, using the control structure model and the identified factors as a guide, designers are led to develop and investigate scenarios that might lead to these factors occurring. Thus incorporating this into a design decision making process involves making changes based on what mitigates the existence or impact of these factors.

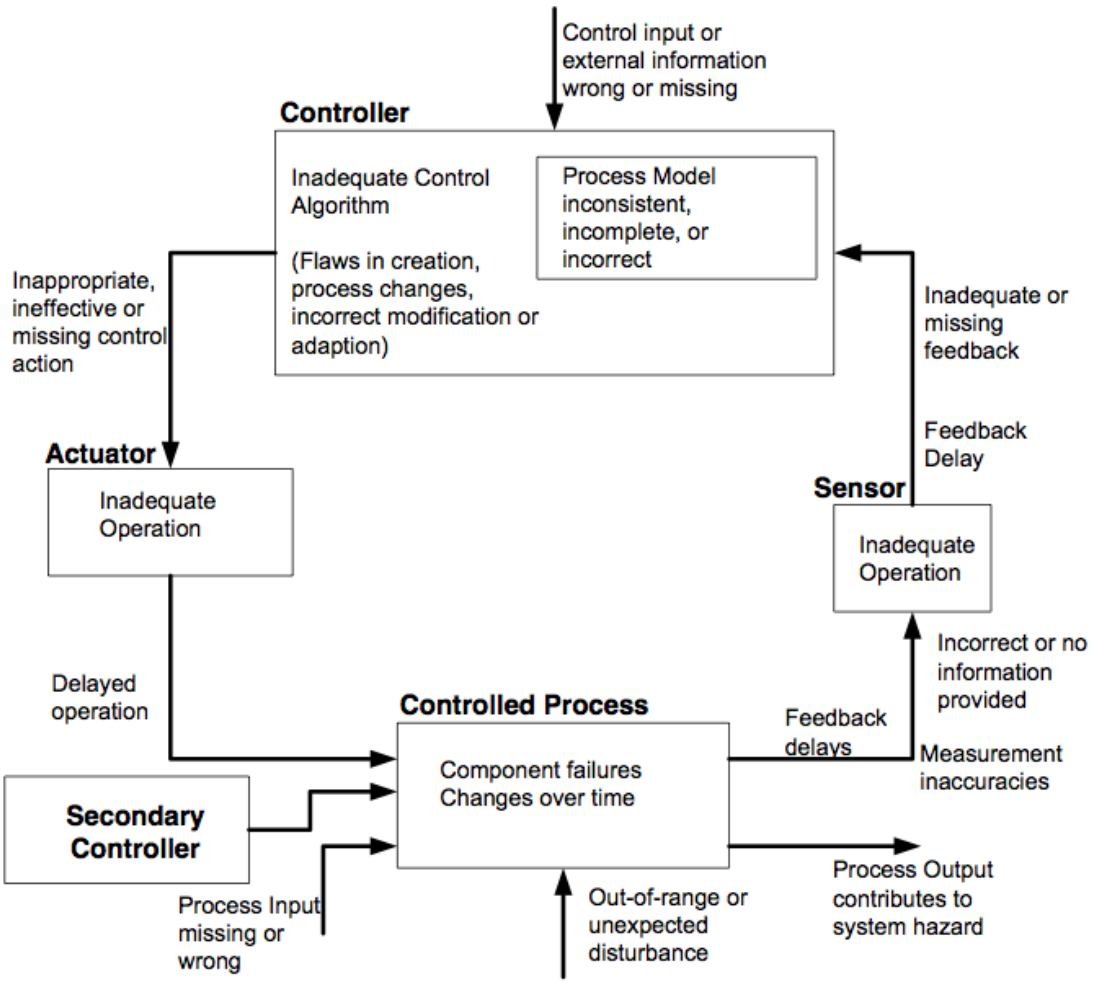


Figure 4.3: Generalized factors that are used to identify potential safety constraint violation scenarios in the STPA methodology [84].

This research adopts a similar top-down view of safety as a system property as found in the STPA method. This research proposes a concurrent safety-based control structure design with the device-centric functional design. Instead of expert judgment in STPA to identify scenarios, this research proposes a model to identify triggers of the causal factors through system simulation. Considering the successful use of expert-based methods such as Fault Trees and HAZOP, this research is focused on automating the design decision making process with respect to safety. It is not attempting to replace the important role played by system safety experts but rather augmenting this process with behavioral simulation.

4.2 The Safety Function

This research uses a hazard-failure model depicted in Figure 2.1. This model describes the accident process as a system moving from a hazardous state to a mishap state. Following the Systems Theoretic Accident Model [134], this transition occurs when a system safety constrain is violated. Therefore, a measure of safety for a system is the relative difficulty (or likelihood) for those constraints to be violated. To enable design-stage decision making this metric of safety must be evaluated early in the design cycle to provide the greatest benefit to designers. In a model-based design framework this indicates a need to explicitly represent the system property of safety.

To that end, we define a new concept called a "Safety Function." A safety function is a subset of the emergent behavior of a system that inhibits the system state transition from hazardous to mishap. In this way the safety function is analogous to the system's inertia, causing the system to resist moving to the mishap state. This means that when a system is in a hazardous state it can only transition into the mishap state if the safety function is lost or otherwise ineffective. It is this property of the safety function that enables safety-based design evaluation. Thus, the first goal of this work is to develop a method of representing the safety functions as part of the design.

The word function is used because, similar to the way that functions are the actions of what a system does, safety functions are a type of action. Using part of the Keuneke function ontology [131], safety functions either *maintain* the system state or *prevent* the system transition. While the functions of the system can be decomposed into sub-functions and function structures, this is not the case for the safety function. Where a

component type function acts on a flow to exhibit some behavior, the safety function reflects a behavior at the system level. Further, this behavior only exists at the system level and cannot be decomposed into sub-safety functions. The safety function follows an effect view of function, representing a phenomena and not an input-output relation.

The identification of safety functions should occur at the early system design stage. After a Preliminary Hazard Analysis (PHA), system designers should have a list of expected hazardous system states. The established approach in industry for conducting a PHA utilizes hazard identification causation lists to guide designers in identifying hazardous system states [21]. A hazardous system state is description of the system and its environment that has the potential for becoming an accident. This list should grow as the design becomes more refined, however the identification of safety functions can begin with this list. The development of this list is left to expert knowledge of the system and its potential operating environments. Desired safety functions can be identified through an "if...then" type of analysis. For example, *if* a car is operated at night and the driver cannot see *then* the car may hit something. In this example, the car and driver is the system and its operating environment is at night. The hazardous state is not being able to see in this operating environment. The first stage in a safety guided design process is to attempt to eliminate hazardous states. However, this cannot always be done with a high degree of certainty or is undesirable. One way to avoid the example hazardous state is to make it impossible to drive vehicles as night (undesirable) or provide headlights (the state probability is only reduced). Thus, the desired safety function is *to prevent* the vehicle from hitting anything when the car-person system is in this hazardous state. In this example, the person is the controller and this safety function is typically implemented by the driver stopping the vehicle. This safety function would need to be explicitly implemented in an autonomous vehicle.

In this way, a desired safety function is identified for every hazard to mishap transition. Each safety function is implemented in the system design with a control structure. This control structure is not the entire system control scheme (which implements many safety functions), but rather one specific loop focused on mitigating the hazard. As seen in Figure 4.4, a control structure has four elements. Namely, the *controller* which directs the *actuators* to allow a desired *process* which is monitored with *sensors*. Using the previous vehicle example, the human controller actuates the braking process and observes through sensors both the arrival of the hazardous state and the result of actuating the

braking process. As evident from the example, the parts of the control structure are real components (either physical or cyber) that exist in the system. In this way, we can connect the function-based cyberphysical architecture design to safety functions though the physical components which implement those safety functions. This process is detailed in the following section.

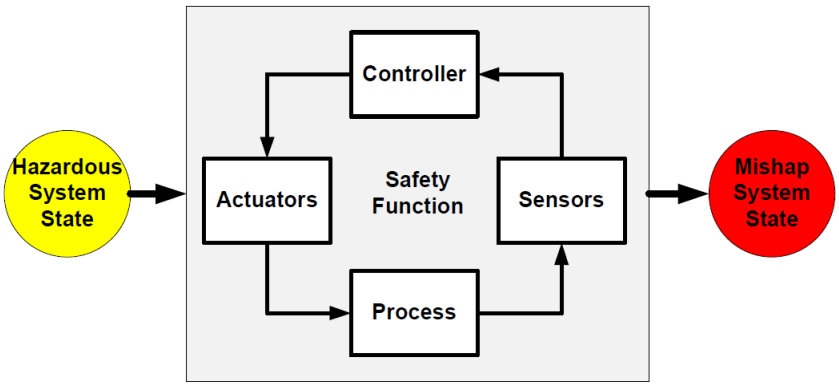


Figure 4.4: The general control structure for implementing a safety function which inhibits the system transition from a hazardous state to a mishap state.

4.3 Incorporating Safety Functions into Model-Based, Functional System Design

As discussed in section 2.1, there exists in the literature variety of tools and methods for both model-based design and function-based system design. This section illustrates adding the representation of safety functions into one approach to system design and using a particular set of tools. Specifically, we follow a device-centric view of functional system design [23, 135, 133, 122], where high-level system functions are decomposed into functions structures of functions acting on flows. Based on this structure, generalized components are identified to implement those functions and the architecture of those components represents the cyberphysical design. Further, we use the systems modeling language (SysML) [105] to express these design representations. Using other methods

and tools may require minor modifications of the following steps, however, the main concept is broadly applicable.

The propose safety-centric design process is as follows:

1. Identify the set of hazard-to-mishap transitions that will require implementing safety functions. These transitions come from a preliminary hazard analysis conducted at the system level.

2. Generate the generalized control structure representation for implementing each safety function.

3. Follow the functional design process to decompose high-level functions into an implementable function structure.

4. Generate a system cyberphysical design architecture by implementing each function with one or more general components. The mapping between function to component is important for later analysis of the design.

5. Identify the components in the cyberphysical architecture that correspond to the general elements for each safety function.

6. Use this mapping to generate the detailed control structure for each safety function.

To illustrate the steps of this processes we develop an early design model for a spacecraft maneuvering system.

4.4 Example Application

As an illustrative example, consider the early design of a maneuvering system for a satellite. One of the key features of this system is that it is composed of multiple technology domains. This design requires a controlled electrically driven thruster. Further, this system needs to be capable of manual and autonomous operation. Thus this system is composed of subsystems containing electrical, mechanical, thermal-fluids and software control. This conceptual design demonstrates that the emergent property of safety at the system level is often a result of multiple subsystem behaviors.

For this work, the SysML block definition diagrams (BDD) and internal block diagrams (IBD) are used to represent the functions and components of the system. In these

diagrams there are blocks representing either functions or components (or component classes) and arcs connecting these blocks expressing some type of relation. There are multiple relations specified in the SysML specifications, however, for this example we will only use three. The first type of relation is a directed composition. This type of relation specifies that one block is composed of the connected blocks. Figure 4.5 is a BDD illustrating the elements of the generic control structure depicted in Figure 4.4. The second type of relation is used for requirements and indicates that a component *satisfies* a requirement. This relation can also be seen in Figure 4.5 indicating the safety function that is satisfied by this control structure. The BDD shows the relation of parts to the whole but not the connection of parts to each other. The IBD is used to show how parts of the design relate to one another. Here the third type of relation is used to link blocks together to indicate the transfer of material, energy, or signal.

This design process assumes that the functional requirements and PHA have been generated. The identification of the safety function from the desired hazard mitigation follows the process identified in the previous section, assigning a generic structure like the one in Figure 4.4. This is the explicit representation of the safety property of the system. For this example, we assume that a potential system hazard state is defined as the system operating under manual control. The potential mishap state is that the craft may lose (to a point of no return) the planned orbital trajectory. Therefore, the designers intend to implement a safety function to prevent this transition. This safety function and the generic control structure component types which implement it are represented in a BDD (see Figure 4.5).

The next step is the functional design of the system. A high level functional decomposition is represented in the BDD in Figure 4.6. Further each of the functional blocks will interact with each other. This high-level functional structure is illustrated in the IBD of Figure 4.7. From this diagram it can be seen that the electrical power system provides the electrical power for the controller software to operate as well as the power for rocket controlling actuators. The rocket provides thrust to the system and the controller will receive status and send control signals to the other blocks as wells as receive and send signals to the operator. The individual subsystems are further decomposed into an implementable function structure. Figure A.1 in Appendix A illustrates the BDD showing the components that are chosen to implement the electrical power subsystem. The system architecture is based on the function structure and represented as an IBD

in Figure A.2 in Appendix A. Likewise the rocket subsystem functional structure and component architecture are depicted in Figure A.3 and Figure A.4 in Appendix A. From these figures the overall maneuvering system design is formed. The maneuvering controller (which operates on hardware powered by the electrical system) uses the electrical subsystem to control the fuel and oxidizer valves of the rocket subsystem. This represents one possible design solution for this system. One important point that can be seen in the electrical power subsystem design (Figure A.2) is that this system provides electrical power to other subsystems not considered in this example. This reflects the reality that subsystems perform multiple functions and further that behavior of one subsystem cannot be considered entirely apart from other subsystems. This system has a suite of sensors in the electrical and rocket subsystems that the controller can use to determine the current state.

The next step is identifying what components implement the specific safety function control structure. As mentioned previously, this model does not reflect the entire control schema but just the safety function implementation. At the high-level the function structure of Figure 4.7 can be used to define the connections in the IBD of the safety function control structure. This is shown in Figure 4.8. This is further refined based on the components used in the cyberphysical structure diagrams to a detailed safety control structure. In this example (see Figure 4.9), implementing the safety function that prevents the loss of orbital trajectory requires components from all three subsystems. These components are not physically directly connected, however, they form a structure for performing the safety function at the system level. In this way, the safety function control structure provides a different view of the same system as the cyberphysical structural diagrams.

4.5 Summary

In this chapter we demonstrated a method of safety-based design where the system property of safety is explicitly modeled in the design process. This process uses the functional design process as a model for the design using “safety functions”. Safety functions perform one of two roles in the system. They either prevent the system state transition from hazard to mishap or maintain the current system state. In the same way that functions are implemented with components in the functional design process, this

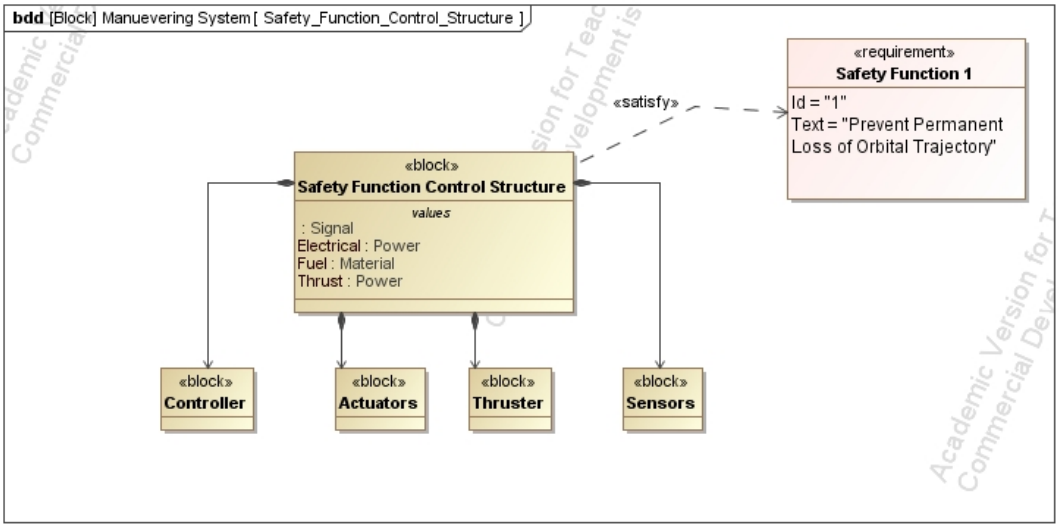


Figure 4.5: A SysML block definition diagram illustrating the composition of the control structure for implementing the safety function "Prevent permanent loss of orbital trajectory" for the hazardous system state of manual operator control.

chapter shows how safety functions are implemented with control structures. As evident by the example of the satellite maneuvering system, the components that form the safety function control structure are often found in different subsystems and are not directly connected in the physical architecture.

The main outcome of following this approach is the explicit mapping between components in the system and the execution of safety functions. This has limited impact by itself. What has been presented in this chapter does not answer the question of *how well* the safety function is implemented in this particular design. Instead this chapter forms the foundation for how reasoning on component behavior can be linked directly to hazard mitigation. The following chapters will discuss how component behavioral simulation and failure reasoning can be used to evaluate designs to answer the *how well* question.

The first assumption for this process is that the safety function is lost or inoperative when there is a failure in the control structure. This is based on the STAMP accident model [134]. Therefore, the consequence of any behavior in the system that causes a

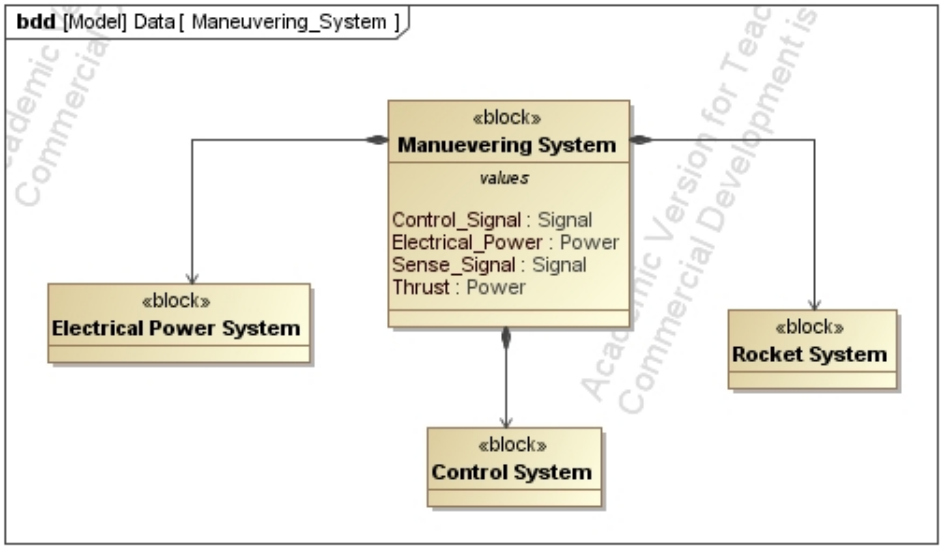


Figure 4.6: A SysML block definition diagram illustrating the functional decomposition of the spacecraft maneuvering system.

failure of the safety function control structure is the loss of mitigation for that specific hazard. Thus the mapping presented in this chapter enables the link between component fault analysis and system hazard analysis.

One approach to evaluating a design based on this safety function control structure is to look at the reliability of the components implementing the control structure. While this would provide a preliminary analysis, it does not address failures (or behaviors) that are not part of the control structure but affect the components within it. Thus the focus of this research is a behavior simulation approach where the impact of faults and critical scenarios is evaluated in terms of the effect to component and safety functions. This evaluation occurs through functional failure reasoning, where logical rules are used to evaluate the simulated system behavior to identify the operating state of both the component type functions and the system safety functions. The following chapters explore how to evaluate the system design using this approach.

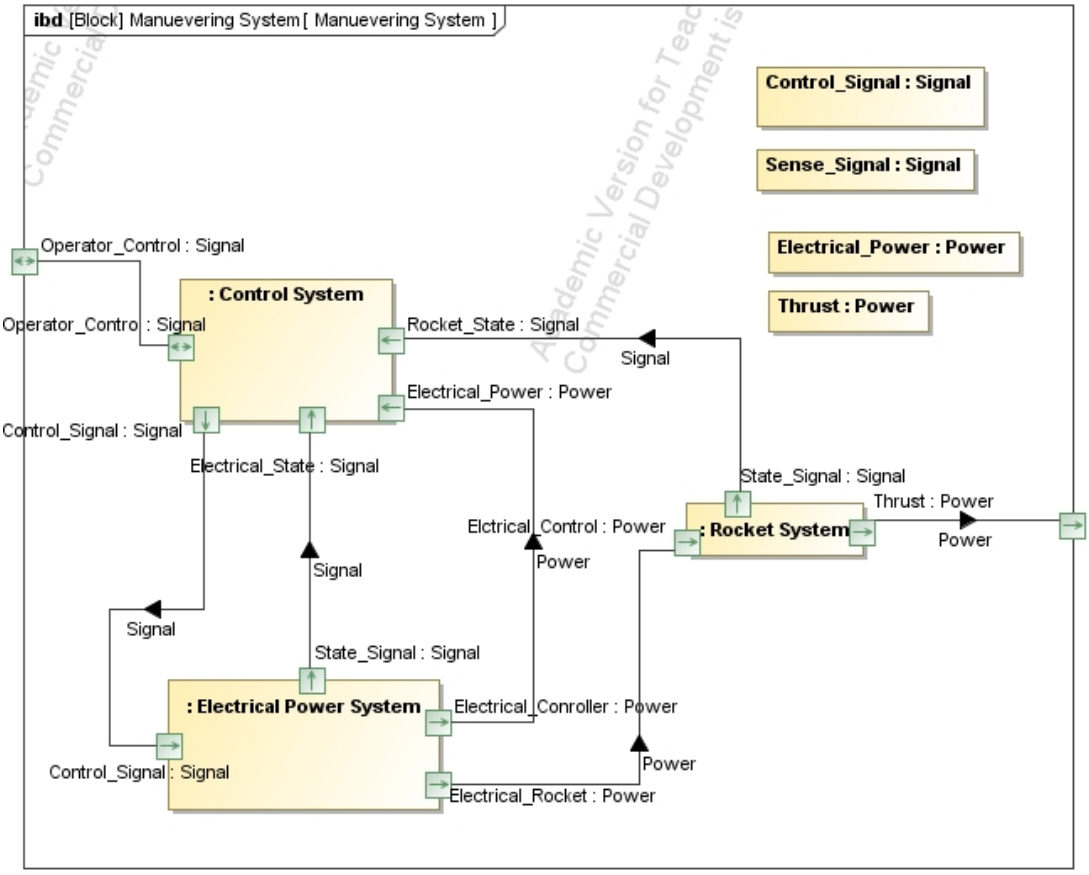


Figure 4.7: A SysML internal block diagram illustrating the function structure of the spacecraft maneuvering system.

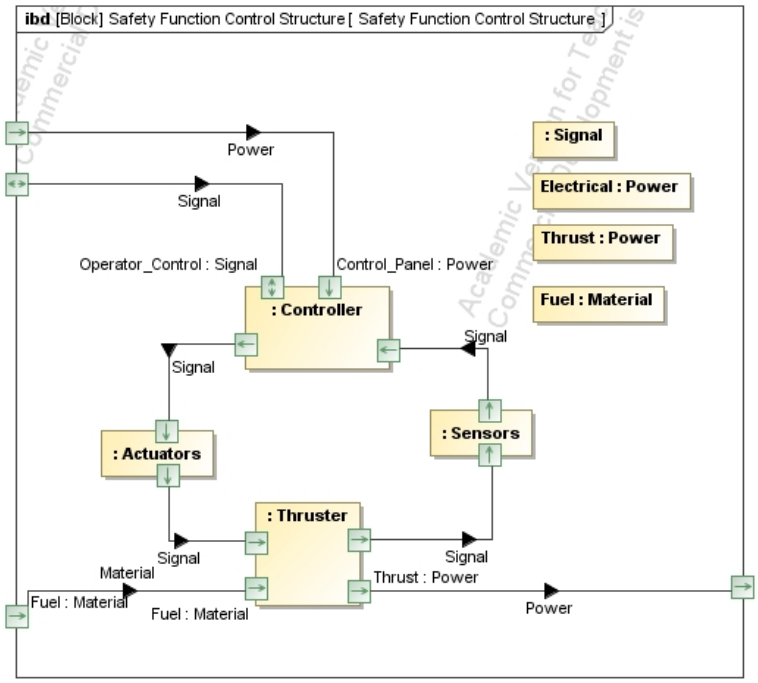


Figure 4.8: A SysML internal block diagram illustrating the connections of the control structure for implementing the safety function "Prevent permanent loss of orbital trajectory" for the hazardous system state of manual operator control.

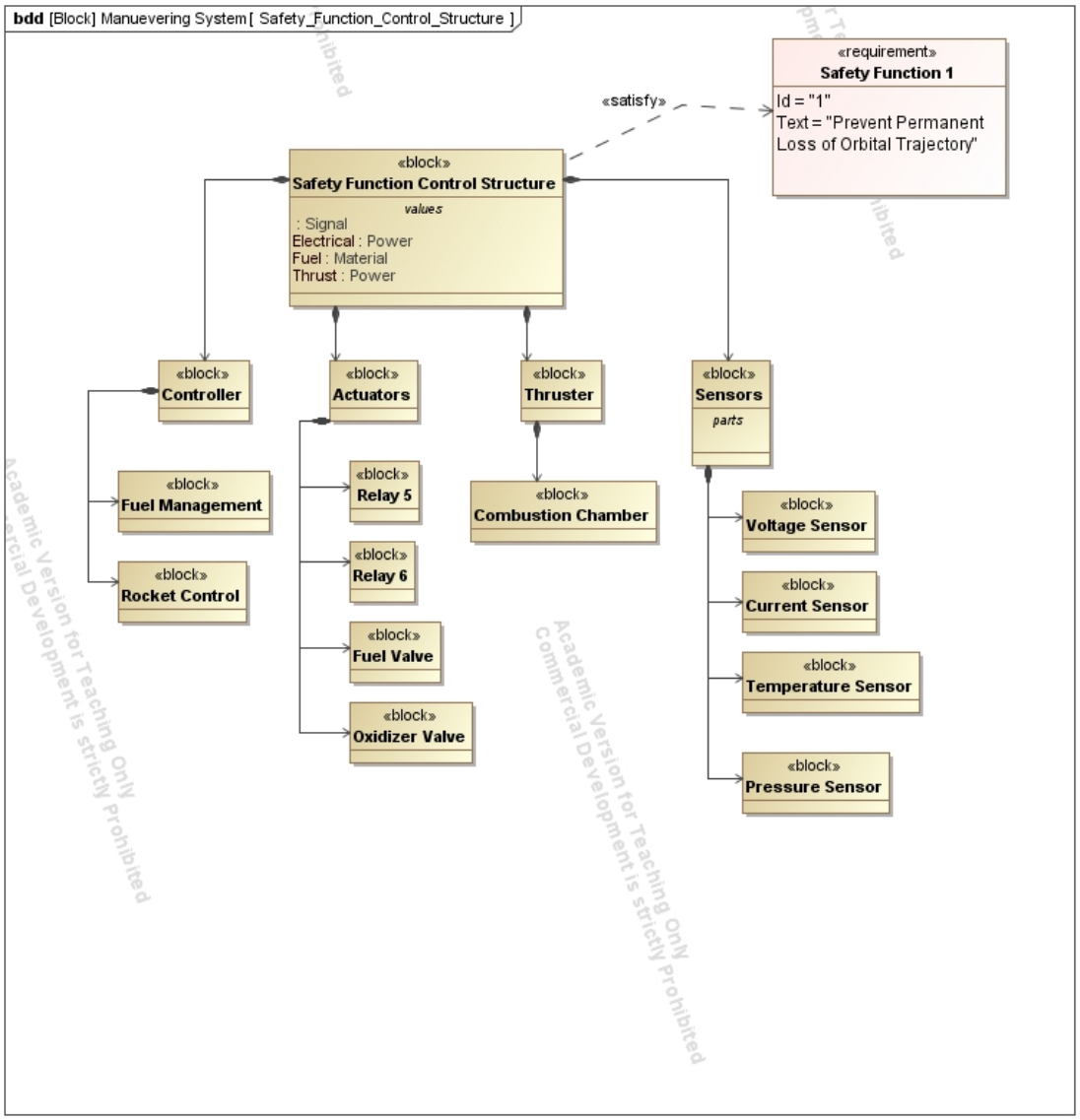


Figure 4.9: A SysML block definition diagram illustrating the composition of the control structure after mapping to the components derived from the functional design process.

Chapter 5: Function Failure Reasoning for Complex System Design

This chapter begins the analysis section of this research. The goal of the work presented in this chapter is to assess the functional impact of component faults using qualitative simulation. This work is further focused on quantifying that impact as a means of enabling design decisions, specifically system architecture. The application is to an electrical power system which also forms the basis of the electrical subsystem in the safety-guided design framework case study. The content of this chapter was published in the Journal of Research in Engineering Design and was cowritten by David Jensen, Tolga Kurtoglu and Irem Y. Tumer [72].

5.1 Background

The state-of-the-art on diagnostics and fault management has been on the diagnostic reasoning to mitigate faults when they happen, based on matching data to models during operations. A fairly recent trend has been to push the analysis of potential failures into the design stage to better understand and design against the types of faults that can happen when systems are in their operational environment. To help with this challenging task, risk based design methods have emerged as a means to bridge the gap between operations and design. The following sections summarize the relevant background and prior work in these areas.

5.1.1 Reasoning about Faults During Operations

The operational faults in complex systems have been a central theme in diagnostic reasoning, which originated from the Artificial Intelligence community. Fault diagnosis is the process of determining the cause of any abnormal or unexpected behavior in a complex system [136]. De Kleer and Kurien [137] have distinguished two traditions within fault diagnosis. The first has been primarily concerned with analyzing diagnosability and testability of the system and what instrumentation is needed to accomplish diag-

nostic functionality. These techniques are used to analyze the degree of observability of a system and modify the system to meet a set of goals for observability. The inherent testability of a system is determined during the design cycle. This analysis is usually performed before any tests are designed and is based on the physical topology of the system and proposed instrumentation locations. Achieved testability is a maintenance characteristic that describes the ability to observe system behavior with the implemented instrumentation. Industry leaders in this field are TEAMS from Qualtech Systems, Inc. (QSI) [138], and eXpress from DSI [139]. These tools use a model, which captures the physical connectivity of system components and maps failure modes and instrumentation points onto a dependency graph [140].

The second fault-diagnosis tradition is model-based diagnosis (MBD), based largely on early work in qualitative physics and qualitative reasoning [141, 130, 142, 143]. 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 [144]. The artificial intelligence community for MBD employed system configuration and qualitative behavior models for diagnosis tasks [145]. Livingstone [146] and its extension L2 [147] 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 [148]. 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 [149], and statistical and probabilistic classification methods are applied where physical behavioral is difficult to model in analytical form [150, 151].

5.1.2 Traditional Risk and Reliability Analysis

The first response in the research community to move the assessment of potential failures and risks into the early stages of design has focused on using traditional risk and reliability analysis techniques. These efforts look at system components, critical events, and system characteristics to assess risk and reliability during the design phase. Specific techniques include failure modes and effects analysis (FMEA) [53], fault tree analysis (FTA) [76], and probabilistic risk assessment (PRA) [78, 79, 152]. FMEA is a bottom-up approach that follows forward logic to determine critical component failures and their

consequences. FMEA analysis starts with decomposition of the system into subsystems and finally into individual components. Failure mode(s) of components are then recorded and assessed separately to determine what effect they have at the component level, and then at the system level. FTA is logic-based analysis, which starts with identification of a high-level failure event. A backward logic is then followed to drive contributing events that could lead to the occurrence of immediate higher-level events. At the end, the analysis presents the chain of events combined with logical gates in a tree structure. Probabilistic Risk Assessment (PRA) typically uses event trees and fault trees to identify the possible causes of undesirable outcomes and the consequences of those initiating events. PRA defines risk mathematically as the 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. In addition, there are numerous approaches for combining both static and dynamic fault trees and event trees into a single model [153, 154].

5.1.3 Research in Function-Based Failure and Risk Analysis During Conceptual Design

The use of the functional representation for design stage failure analysis can be summarized into two paradigms. The first paradigm uses the functional representation with the goal of operations-stage failure effect mitigation. The second paradigm uses functional representation with the goal to formally document system failures and effects and inform design decision making. Because of the difference between the goals of these two paradigms the methods found in each have a different set of outcomes and limitations.

The first paradigm focuses on mitigating failures at the operations stage. Examples can be found in the work of [155, 132], [30], Sasajima et al., [29], [66]. Specifically, the Functional Redundancy Designer was developed to identify functional redundancies in a design [155]. This was accomplished by analyzing the structural architecture of a system to identify physical features of a design that are capable of performing identical func-

tions. Building from this work a model-based reasoner was developed for diagnosis and reactive control [132] that can identify functional losses, a set of fault cause candidates, and determine control actions for recovering from functional failures. FBRL was also used as a basis for computer-aided support of FMEA type of analysis [66]. In failure analysis, connecting failed state component behavior to intended function can be challenging because failure behavior can eliminate, or significantly alter the function that a component performs.

In work aimed at assessing reliability at the concept stage, Smith and Clarkson present a systematic method for designers to manually think about potential failures and their consequences [61]. In this method, designs are represented as entity-relationship diagrams linked to functional requirements. These relationships are used as guides for identifying the entity generating detrimental effects, those entities and relationships which transfer those effects and the entities that are sensitive to those effects. Derelöv [156] also presents a concept analysis method for identifying potential failures based on identifying physical failure phenomena with components. When there are similarities between the system representation and the phenomena representation a potential fault may occur and is linked to an event tree. However, fault propagation is still expected to follow the nominal component connections.

The second paradigm focuses on assessing and documenting the effects of failures for decision making during early design. Of these, the Function Failure Design Method (FFDM) [67, 64] was one of the first methods to formally connect faults to functional losses with the goal describing the space of potential system failures. 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 functions. Based on historical failure data for these types of components, it is then possible to establish likely failure modes for a given function. Because historical fault propagation data is configuration specific, the FFDM method is limited to single fault impact analysis. Several other methods built upon the FFDM methodology. An extension of this work was to enable the design of health monitoring systems concurrently with system design in order to reveal, model, and eliminate associated risks and failures [157]. Another extension introduced the Risk in Early Design (RED) method of formulating functional-failure likelihood and consequence based risk assessment, classified as high-risk to low-risk function-failure combinations [65, 68].

The above methods are limited in their ability to assess the impact of multiple failures and failures that result from system interactions. To overcome this limitation, other work has focused on including the propagation of failure in the analysis. The Failure Propagation Analysis Method [69] was developed as a direct extension of the FFDM and RED methods to capture the 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 [70] to apply to the functional level. The Change Prediction Method utilizes a Design Structure Matrix to assess the likelihood and impact of design changes that propagate through a system [70]. Both the Failure Propagation Analysis Method and the Change Prediction Method utilize the nominal system representation to assess propagation through the system. Other methods have avoided using detailed historical data on failure and instead focused on qualitatively representing nominal and failed system behavior while focusing on the functional effect of failures. A Bayesian network analysis tool was introduced to evaluate the properties of function structures based on dependencies between flows and functions [60]. In this method, the causation relationship is identified between a flow and every functional failure for each identified high-level function. Failure propagation is then 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 measure to the failure of each function in the function model. An extension of this work is the Conceptual Stress and Conceptual Strength Interference Theory (CSCSIT) method [63], where the conceptual strength of a function is defined as 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.

Another effort has looked at the cost-benefit relationships between functional design and risk mitigation. Specifically, a risk based decision making method was developed by Mehr and Tumer [158], namely, the Risk and Uncertainty Based Integrated and Concurrent (RUBIC) Design methodology, fueled by the need to assess the risk of integrating prognostic and health management capabilities in large aerospace systems, at the system design stage. In this work, risk was defined by a triplet of fault type, fault probability, and fault consequential cost, and used to determine optimal resource allocation for the

detailed design phase; however, risk mitigation attributed to the prognostic and health management capabilities was not explicitly quantified in this formulation. An extension to this framework was to enable a cost-benefit analysis (CBA) of integrating new technologies to large complex systems [159]. The CBA framework provided an optimization framework for the allocation and cost justification during functional design, based on a formulation using probabilistic reliability metrics such as system availability, cost of detection, etc.

Finally, the Function-Failure Identification and Propagation (FFIP) analysis framework introduced by Kurtoglu and Tumer [62] significantly extended the process of identifying functional failures during the system design stage by combining failure identification with model based reasoning. The FFIP method provides a unique way of doing failure analysis at the very early stages of design, combining decision-making and automated reasoning driven by functional failure analysis. Most fault analysis tools including model-based and data driven diagnostic methods share an after-the-fact approach that looks at symptoms of faults and traces them back to the causes of those effects [136]. These methods are required to constantly monitor a system during operation, estimate the systems physical state, and often times react to faults through self-repair and reconfiguration. For such tools it is also important to reason about the temporal progression of fault manifestations as they gradually develop. Contrary to these approaches, FFIP is not intended to be a tool that "reacts" to component or functional failures in real-time during system operations. Instead, it is developed as a design tool that aims to eliminate or reduce the likelihood of reaching certain possible futures by formal analysis of risk of failures early in the design process and by proper guidance of decisions before the design becomes solidified. In this regard, it can be thought of as a conceptual evaluation tool that is used to perform relative risk comparison of competing designs and associated design decisions. FFIP will be explained in more detail in the next section, as it is used as the starting point for the current paper.

5.2 Design Analysis with the Function Failure Identification and Propagation Framework

This section introduces a new methodology for reasoning about the functional failures during early design of complex systems. The identification of risks of a potential loss of

system functionality during the earliest stages of designing complex systems is of growing importance for risk sensitive industries. Early stage design provides the greatest opportunities to explore design alternatives and perform trade studies before costly decisions are made [160, 161]. The goal of this research is to develop a formal framework that enables system architecture analysis of complex systems during the conceptual design phase. The analysis of potential failures and associated risks of functional losses performed at this earliest stage of design will facilitate more informed decision making at the system architecture level, and thus the development of more robust and reliable system architectures [161, 159, 158].

Many methods have been introduced in recent years to move risk based analyses and decisions into the early stages of design. The intended goal of what we generally call risk-based design (RBD) is to use formal methods to understand and characterize risk drivers as the design develops and incorporate this information into principles, tools, or methodologies. The methods are then intended to assist designers in making design decisions that reduce risk while meeting overall system goals. To achieve this goal, we assert that the designers must identify functions, risks, and failure modes related to design decisions and enable making design decisions and choices based on risk and failure information. One way of doing this is by understanding the nature of the failure and its potential impact on the functionality of the system. This kind of impact assessment requires establishing a computable relationship between components and their failure modes, the functionality of components, and the propagation of failure effects. In prior work, we have introduced the Functional Failure Identification and Propagation (FFIP) analysis framework that integrates all these aspects into a formal framework to enable the analysis of functional failures and their impact on overall system functionality [62]. In this chapter, we develop several unique capabilities that will improve upon FFIP to result in a Functional Failure Reasoning (FFR) framework by: 1) Integrating quantifiable measures that define risks based on functional decomposition and the role of functionality in accomplishing design goals, and, 2) Relating impact analysis results to system level architecture design decisions based on a reasoning scheme using models and rules relating functionality and failures.

In order to demonstrate the proposed methodology, this chapter analyzes the design of an electrical power system (EPS) shown in Figure 5.1 as an application. The design problem was to develop a representative testbed facility to be used for testing,

evaluating, and maturing of diagnostic and prognostic health management technologies at NASA Ames Research Center, known as the Advanced Diagnostic and Prognostic Testbed, or ADAPT. Specifically, three examples are shown illustrating how the relationship between system redundancy and criticality of functional failures can be explored to analyze alternative system architectures.

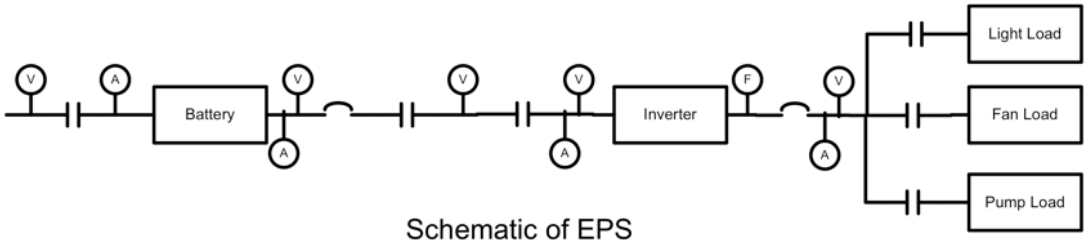


Figure 5.1: The schematic of the basic electrical power system (EPS) design. The system is used to provide power to various components including pumps, fans, etc. In this chapter, three alternative conceptual system architectures of the EPS design are studied.

5.3 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. The FFIP framework uses a simulation-based analysis of functional failure propagation. The modeling scheme used in this framework is common for modeling conceptual dynamical systems. For example, function-behavior-structure (FBS) paths developed by Qian and Gero [30] constitute a method that presents a formalism to represent function, structure, and behavior of systems. In this technique, relations among function, behavior, structure, and processes are utilized to define FBS paths, which are then used to retrieve design information to conduct analogy-based design. However, FBS paths only model nominal behavior of components. FFIP models both nominal and faulty behavior of components and failure conditions of functions so that functional failures and their

propagation can be assessed from a system model using behavioral simulation. The simulation and reasoning approach developed has its roots in qualitative physics [141] and qualitative reasoning [130, 142, 143] and utilizes a finite state representation of system behavior and performs reasoning based on qualitative relationships between functional and behavioral models of system components.

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.

5.3.1 Step 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 [62]. 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.

Functional Representation: System function is represented using function structures [113, 31, 119], establishing a formal function-based design paradigm based on the concept of functional modeling [119, 133]. 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[133], and the functional modeling processes proposed in the literature [113, 31] are used to develop the functional models for the systems under study.

Architectural (Configurational) Representation: The architecture, on the other hand, is captured using configuration flow graphs (CFGs) [162]. 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 [163]. The component types in a CFG can be thought of as generic abstractions of common component concepts.

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 [62].

5.3.2 Step 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 systems 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 [113], where the overall function of a system (i.e., the black-box representation) is decomposed into smaller lower level functions in a hierarchical manner. For example, for the EPS design (Figure 5.1), the overall functionality of the system is to provide power, which then can be

decomposed into three top-level functions, namely supply power, distribute power, and operate loads. 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 5.2 where two system architectures derived from the same overall function are shown. In Figure 5.2(a), 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 5.2(b) functions 1 and 2 are implemented by two elemental sub-functions and function 3 is implemented using three sub-functions.

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 5.2. 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 5.2(a), an equal criticality distribution is assumed for the projection of FCR values, whereas in Figure FCR(b), 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.

5.3.3 Step 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.

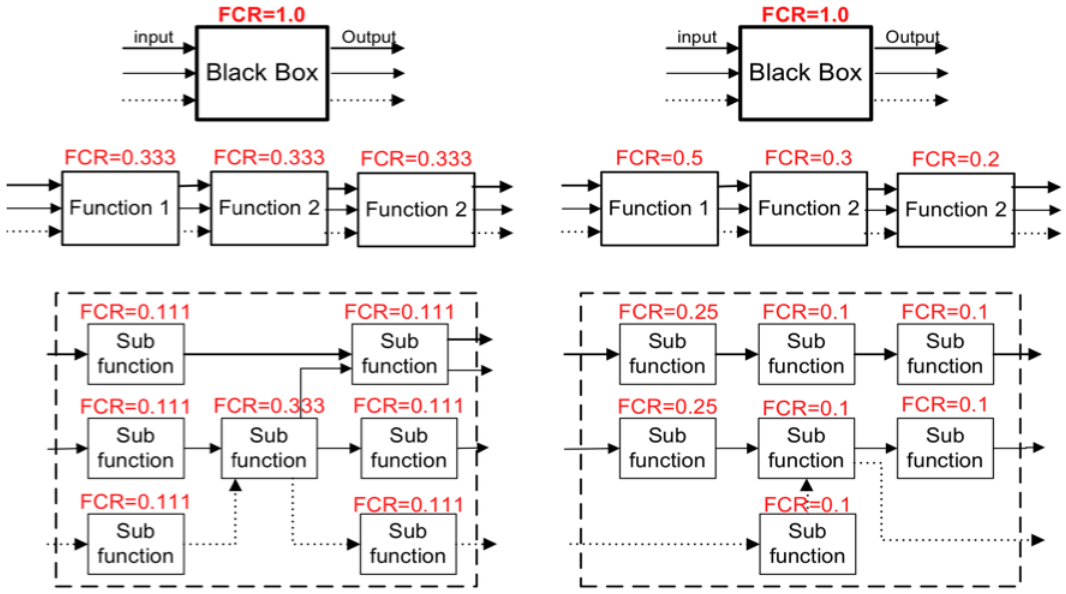


Figure 5.2: Illustration of functional decomposition and the estimation of functional criticality ratings: (a) FCR values for system architecture A estimated using equal criticality distribution, (b) FCR values for system architecture B estimated using skewed criticality distribution.

Step 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 [62].

Step 3.2: Running the FFIP behavioral simulation on the set of scenarios. 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 [62]. 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?"

Step 3.3: Running the FFIP function failure logic on the set of scenarios. 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 [62]. Accordingly, system functions are classified as 'operating, 'degraded, 'lost recoverably or 'lost, defined as follows:

1. Operating: Function operates on a flow as designed
2. Degraded: Function operates on a flow not as designed
3. Lost Recoverably: Function has no flow to operate on because of a different functional failure
4. 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.

Step 3.4: Calculating the Functional Failure Impact (FFI) for the selected set of scenarios. Finally, after the FFIP analysis is run, the Functional Failure Impact of a selected scenario can be calculated by summing over the Functional Criticality Ratings (FCR) of all functions that are classified as degraded, lost, or lost recoverably during the simulation using:

$$FFI = \sum (C_i \times FCR_i) \tag{5.1}$$

Where C_i is the consequential cost factor determined by the functional state of function i (Table 5.1), and FCR_i is the functional criticality rating of function i as determined by Step 3.2.

Table 5.1: Function States and Consequential Cost Factors

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

The consequential cost factors shown in Table 5.1 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.

5.3.4 Step 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 (FFI_m)" 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:

$$RIR\% = (FFI_m - FFI)/FFI * 100 \tag{5.2}$$

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 introduced in Figure 5.1.

5.4 Electrical Power System Testbed Design Example

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 5.1) 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 [164].

5.4.1 EPS Testbed Baseline System Architecture and its Modeling using the FFIP Framework

The EPS testbed was originally designed using the Function-Failure Based Design (FFDM) methodology described in Section 5.1.3 at the early concept design phase [165]. 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 overcurrents 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 [164].)

Figure 5.3 shows a functional and a configurational model of the electrical power system (EPS), which is used as the baseline architecture in this chapter. 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 5.3, 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 5.3, 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 component mode and state variables are identified in Tables 5.2 and 5.3.

To illustrate the component-oriented behavioral modeling approach of FFIP, Figure 5.4 shows behavioral models for two generic components ("relay" and "inverter") from the baseline EPS system of Figure 5.3. 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 fail 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.

Finally, Figure 5.5 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.

5.4.2 EPS Testbed Alternative System Architectures

The baseline design, shown in Figure 5.3, 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

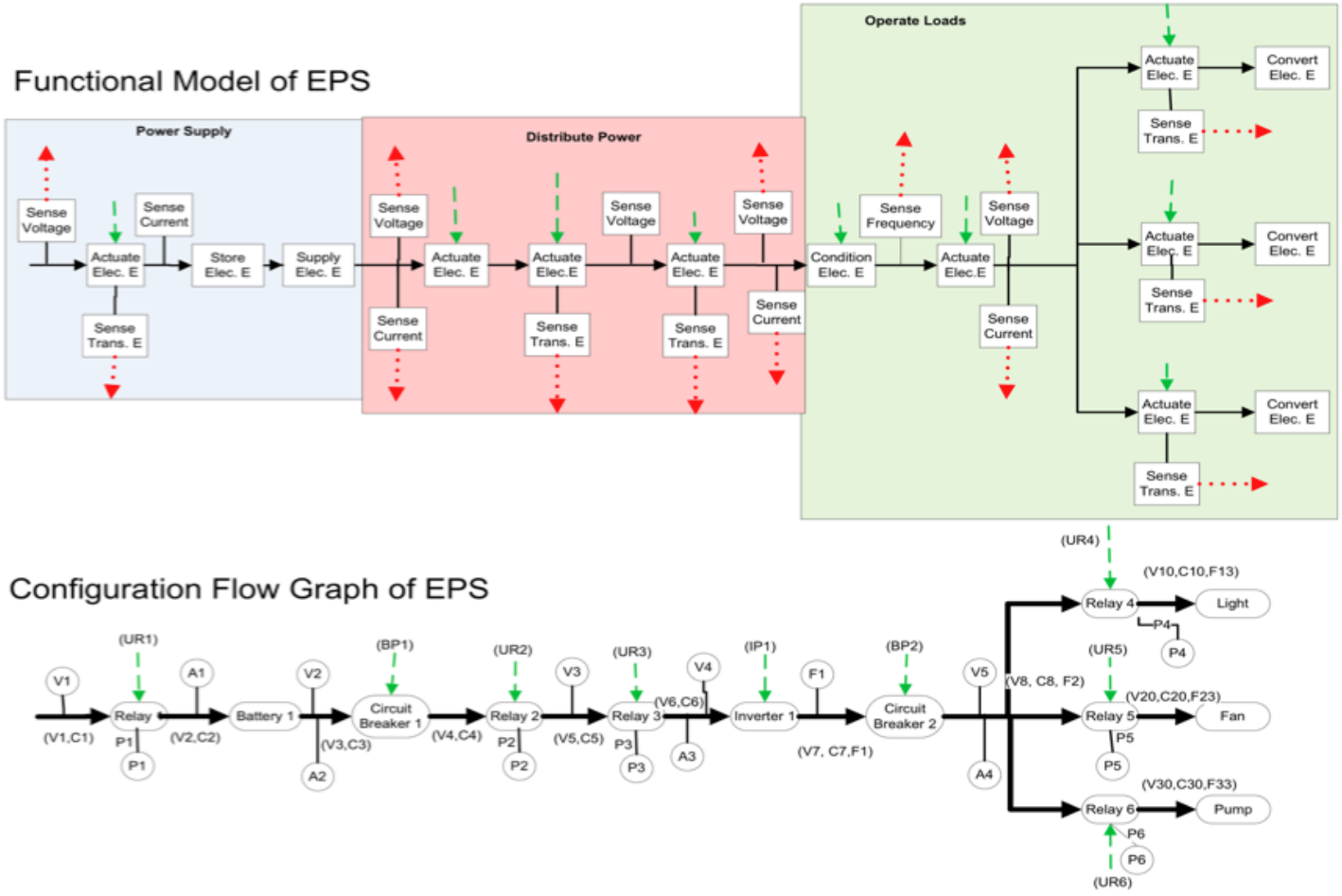


Figure 5.3: A high-level functional and architectural model of the electrical power system baseline design. The functional model illustrates the three top-level functions (i.e. supply power, distribute power, and operate loads) of the system and their decomposition.

Table 5.2: 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 5.3: Flow State Variables for the Baseline 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		

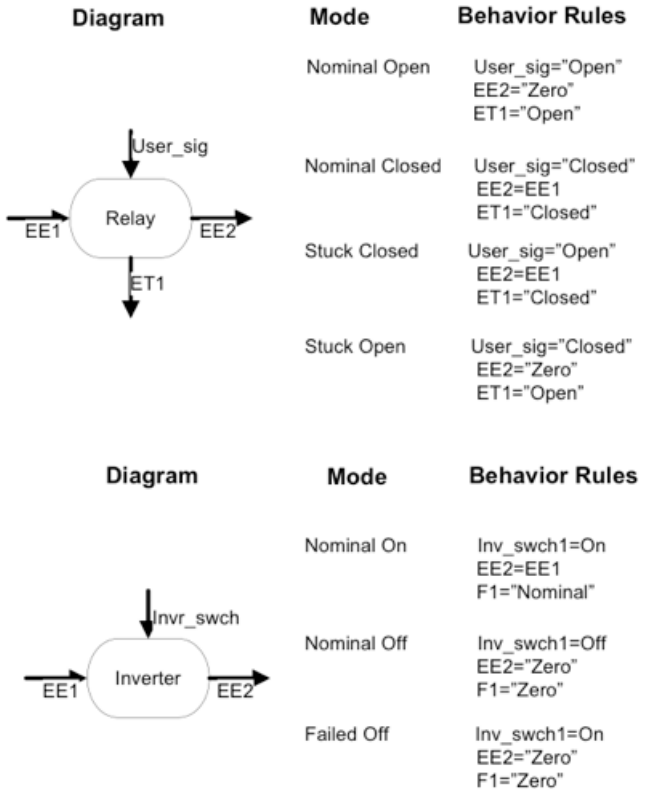


Figure 5.4: Behavioral models for generic “relay” and “inverter” components.

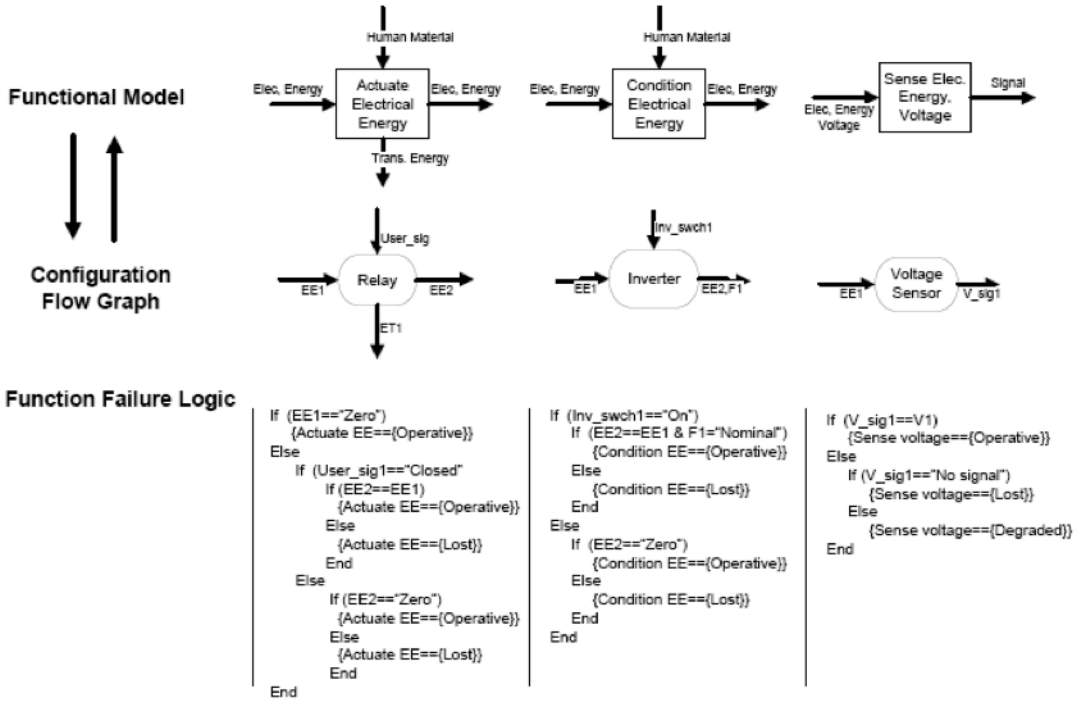


Figure 5.5: Function Failure Logic relates the behavioral model to specific functional health.

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 Appendix B. 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.

5.5 FFR Method Applied to the Electrical Power System Design

Three alternative system design architectures (shown in Appendix B) and the baseline design of Figure 5.3 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 5.3.

5.5.1 Step 1: System Models for the EPS

This step is summarized in Section 5.4.1 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.

5.5.2 Step 2: Determination of Functional Criticality Ratings for the EPS

The function criticality ratings (FCR) for the sub-functions of the baseline design (see Figure 5.3) and the alternative designs (see Appendix B) are estimated using the process summarized in Section 5.3.2. 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 5.3 implements the "supply power" top-level function by decomposing it into six sub-functions (shown with the module box labeled "power supply" in Figure 3), 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, losing 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 5.4.

Table 5.4: Distribution of FCR values for the Baseline EPS Design and the Three Alternative Designs.

Function	**FCR**	**Baseline FCRs**	**Design 1 FCRs**	**Design 2 FCRs**	**Design 3 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

5.5.3 Step 3: Computing the Functional Failure Impact for the EPS Scenarios

Step 3.1: Selecting a Set of Scenarios of Interest. To evaluate and compare the alternative conceptual system architectures, 30 critical fault scenarios are identified and summarized in Table 5.5. 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.

Step 3.2: Running the FFIP Behavioral Simulation on the Set of Scenarios. 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 5.5. 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.

Table 5.5: 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

Step 3.3: Running the FFIP Function Failure Logic on the Set of Scenarios. As described earlier, during the simulation, the functional state of each function is assessed through the FFL reasoner. This is illustrated in Figure 5.6 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.

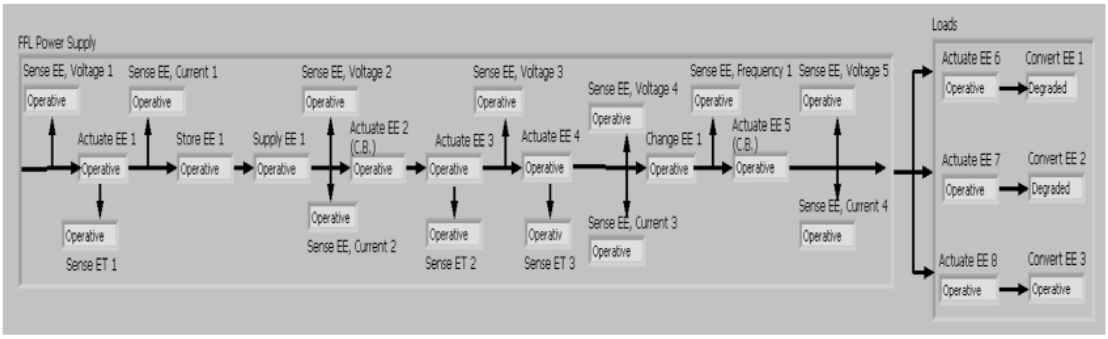


Figure 5.6: The GUI for functional failure reasoning.

Step 3.4: Calculating the Functional Failure Impact for the Set of Scenarios. 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. 5.1. 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 5.5) 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

5.6 Step 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. 5.2. This is shown in Table 5.6 for the two scenarios detailed in the previous section.

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

Table 5.6: 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	0.238	0.238	0.00%	0.119	50.00%	0.119	50.00%

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.

Before presenting a detailed discussion of the results, we would like to emphasize that one should be careful in interpreting the physical meaning of the metrics defined in this study. The metrics introduced in our paper are not defined for traditional risk assessment analysis (such as PRA), which aims at quantitatively assigning risk in a system, but rather for relative, or qualitative evaluation of risk in the system during conceptual design. As a result, the functional failure impact (FFI) and the reduction in risk (RIR) metrics should not be treated as absolute values for quantifying the risk (or risk mitigation) in the system, but as the basis for conducting relative comparisons between competing design alternatives. Similar qualitative interpretation of risk and failure impact is common for other early design analyses. For example, the "risk priority numbers (RPNs)" in failure mode and effects analysis are assigned based on qualitative

assignment of the probability of failure, the probability of detection, and the estimated consequence of a failure all assessed using a linear ordinal scale similar to how we defined the consequential cost factor) C_i.

5.7 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 5.7 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 improve 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 decisions, 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.

5.8 Summary

In this chapter, we introduced a new methodology 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 devel-

Table 5.7: Function Failure Impact (FFI) and the Subsequent Reduction In Risk (RIR) Results

		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%

opment of preventive measures that can effectively and efficiently guard against system failures.

The thirty scenario cases for the design of the electrical power system show how the proposed functional failure reasoning (FFR) methodology can be used to evaluate different conceptual system architectures based on functional failure impact. Several unique characteristics of the develop framework are:

1. First, the framework provides an analytical approach to quantify individual risk of basic functional elements in a system as well as the combined risk resulting from the propagation of functional failures. More importantly, this quantification is derived from system specific function-to-configuration relations integrating the knowledge of which components are used in the system for addressing functional requirements. This is a significant extension to the previously published function-based failure assessment research.

2. Second, the framework provides the designers with a means to determine the level of risk mitigation based on specific architectural design decisions. This is accomplished by computing the direct effect of functional failures and the impact on the overall system safety for different architectures. This paves the way for system level trade-off analysis between architecture, risk mitigation, and cost.

3. Third, the framework provides a means to tackle multiple failures, since it is not built upon the unrealistic single fault assumption. Accordingly, any number of failures can be introduced into the simulation and the framework supports the analysis of multiple failures that affect the system in parallel.

4. Finally, the framework is built upon modular, reusable function-component-behavior models that can be integrated using an industry standard modeling environment. As a result, it allows designers to quickly analyze what components to use in the system, how to configure them, the types and locations of necessary safeguards, and the proper level of system redundancy, all guided by potential functional failures and their impact on overall system performance.

There are several assumptions that the presented method is based upon. These assumptions pose certain limitations that are left to be addressed in future research. For

example, only design decisions targeting system redundancy are tackled in this initial implementation. Such decisions allow the same failure scenarios to be run on the original and modified designs. If, however, more complex design decisions are made governing the addition or removal of a huge number of system components, or the introduction of new technologies, the resulting configuration changes may force a failure scenario to be obsolete for the alternative designs.

Second, the current implementation does not account for the likelihood of different failure scenarios. Certainly, some failures are orders of magnitude less likely than others. In the future, we plan to incorporate failure probabilities of individual components and their failure modes. These can be assessed based on information extracted from past incident and accident logs, or documented component reliability data. This will also enable better representation of scenario probabilities (for example multiple faults are order of magnitude less likely than single-point failures), and the employment of simulation techniques (such as Monte Carlo, discrete event simulations, etc.) that support statistical analysis of probabilistic distributions. Such an approach will improve the current analysis capability, which is limited to a set of critical scenarios (in the presented EPS design, a set of 30 scenarios) that are determined a priori.

Third, the sequence of events to simulate is chosen by the designer. Unavoidably, a designer may miss certain sequence of events that could lead to failures. Exploring the event sequence space automatically for comprehensive coverage of potential failure scenarios is an open area of research left for future studies.

Fourth, this chapter has focused on a design example in which competing design alternatives are differentiated by level of redundancy only. However, the FFR method is not limited to redundancy analysis. In principle, using FFR one may analyze architectures with different component types and connectivity, or architectures that employ different technologies. For example, one may study functional failure impact of different critical scenarios on a typical parallel hybrid and series hybrid car architectures and make architectural considerations accordingly. The challenge in such examples is the fact that the set of critical scenarios may the different between the two designs and therefore the calculation of the reduction of risk (RIR) measure may not be feasible. Nevertheless, designers can still study the functional failure impact (FFI) results of the competing designs and make decisions between different design choices. In the future, we plan on working on examples that show how the FFR technique can be used to analyze

significantly different system architectures.

Finally, the methodology is biased towards achieving optimum risk mitigation. A natural question that follows is: at what cost? Thus, we are interested in expanding the approach to investigate broader costs and benefits of making risk-informed design decisions [158]. That way knowledge from other domains in addition to reliability and risk can be reconciled by including information about the operational consequences of system level failures such as repair, downtime, maintenance cost, etc. In addition, we plan to study the trade-offs between the cost of building the models necessary for the method and the potential benefits for more complex products. Such an integrated approach has the potential to provide significant lifecycle cost savings while designing complex systems.

Chapter 6: Expanding FFIP Analysis with Fault Mode Dependency and Flow State Reasoning

This chapter expands the function-based failure analysis to include new failure scenarios. Specifically, by finding fault modes which may link components in a failure that are not connected in the nominal design. This chapter presents a reasoning approach to find and simulate scenarios where faults propagate outside the nominal system connections allowing for a more comprehensive analysis of potential system failures. The example system in this chapter is the thruster portion of the Reaction Control System (RCS). The content of this chapter was submitted and is in review with the Journal of Engineering Design and is cowritten by David Jensen, Irem Y. Tumer, and Tolga Kurtoglu [166].

6.1 Introduction

Failure analysis in the design stage is a key enabler for generating a robust design. In complex systems, failures often occur as a result of the interaction between subsystems, such as electromechanical and software. Further, a fault can propagate across subsystem boundaries leading to different system level effects based on the specific propagation path. It is the system level effect that determines the significance of a fault. Therefore, identifying the dependencies of a system-level loss to a fault and its propagation path provides a complete failure analysis. When failure analysis also accounts for the probability of that fault occurring it is considered a risk analysis. This type analysis is often done in the validation and late design stage through tools like failure modes and effects analysis (FMEA) and probabilistic risk assessment (PRA) when detailed design information and expert knowledge can be used to determine the interconnectedness of the system. However, as noted by many authors, including Suh [167] and Ullman [115], decisions at the conceptual design stage have a more significant effect on the final outcome of the design. Early methods for dealing with potential failures during design focused on "robust design", where the core concept was to design the system to be insensitive to noise [168]. Other research efforts have focused on moving risk analysis into the con-

ceptual design phase [169, 170, 132, 171, 70, 63, 65, 72]. A common element to each of these different methods for risk analysis is the use of a conceptual system representation for identifying the system-level impact of faults.

A significant challenge for conceptual design-stage risk analysis is using early design representations to determine the expected fault propagation through the system. For most conceptual failure analysis methods, faults are assumed to propagate according to the component connections specified in the conceptual design model. A shortcoming of using these design models for failure analysis was identified in the literature, namely, that the 'as designed' system representation is insufficient for failure propagation analysis for numerous types of failure scenarios [71]. Indeed, the system representation used in the conceptual design stage is based on the expected or nominal behavior. In many possible failure scenarios there can be component interactions that are not part of the intended system behavior. Explosions, impacts, and back flow all represent possible failures that connect system elements that would not be represented in the nominal system models. The motivation of this research is to address how a failure analysis method might analyze failures that propagate along pathways that are not accounted for in the nominal system representation.

The contribution of this chapter is the introduction of a novel approach to determining such failure dependencies. The framework presented here can be applied to develop an automated means of assessing the impact of interacting component failures, providing a means to identifying the consequence of multiple and cascading faults. Using this approach, the analysis is not limited to component interactions that are specified in the nominal-state system representation typically used in design failure analysis. Instead, the analysis searches the space of *potential* component interactions to determine additional system failures. This expansion of the design failure analysis serves to more fully inform designers for risk-based decision making. The presented framework is integrated with a function-based failure propagation analysis tool for early design-stage decision making previously introduced by the authors, namely, the Function Failure Identification and Propagation (FFIP) framework [71, 72, 62, 73, 74].

This chapter specifically presents a method of analyzing interactions and dependencies during the functional design stage that have the potential to result in failures. Section 6.2 presents the foundation of failure analysis within the scope of risk and reliability and the evolution of function-based methods for determining the impact of faults.

Then the details of the Flow State Reasoning framework are presented in Section 6.3. In Section 6.4, this framework is used to modify a function-based failure analysis method and used to analyze a conceptual design for a liquid fueled rocket engine. This is followed in Section 6.5 by a discussion of the results of integrating the proposed framework with a failure analysis method and concluding remarks regarding the application and future direction of this work.

6.2 Background

A common metric for defining system complexity is through the interconnectedness of system elements [172]. For this reason, system design methodologies focus on capturing the dependencies between design elements and across abstraction levels. The main objective in the design stage is turning requirements, customer needs and preferences, and technical constraints into a testable cyber-physical architecture. The functional design approach was developed as a means to enhance the concept generation stage of product design [113]. This method identifies specific functions that a product must accomplish and connects these functions in a block diagram with the energy, material and signal (EMS) flows that are then transformed by the functions. Function flow block diagrams 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 [173, 174]. The Functional Basis [133, 39] was developed in order to avoid the use of designer-specific function and flow descriptions, providing a standard taxonomy for concept design.

The connections in complex system designs between component structures, functions, behavior and the requirements satisfied by each structure have led to the development of formal representation methods to capture this design information. The model-based systems engineering community has used the Systems Modeling Language (SysML) for system design which represents structure, function, and behavior on separate yet connected diagrams [26]. Other approaches have focused on integrating these design features in a single model. The function-behavior-structure (FBS) paths [30] are a formalism for representing function, structure, and behavior of systems. In this method, relations among function, behavior, structure, and processes are utilized to define FBS paths, which are then used to retrieve design information to conduct analogy-based design. FBS could provide designers with analogous nominal operation but was not capable of

presenting failed state system representation. In a similar way, the Function Behavior Representation Language (FBRL) [29] was developed for representing function and behavior with predefined tasks. In this work, functions are defined as conceptual abstractions of a behavior under intended goals, whereas behavior descriptions are specified as relations between input-output parameters of system components.

The Contact and Channel Approach [33, 34] also presents system functionality as tightly linked to the system structure. In this approach a function is achieved through two or more working surfaces pairs (a geometric interface) and a channel and support structure (the volume containing the flow between those interfaces). Implementing this method as a software tool provided the capability of identifying new functionality required based on the chosen physical solution [34]. For example, if a component is used that happens to generate heat, the tool identified a new functionality associated with dissipating that heat. The identification of new energy and material flows is the basis of this chapter as well.

6.3 Flow State Reasoning Framework

The methods reviewed in Section 6.2 all determine component and functional interactions based on a system representation. However, because these methods are applied in the early design stage, the system representations used only captures the nominal state of a system. The nominal state representation reflects the designed intent of the functional realizations and component interactions. A significant issue arises, however, when failure analysis methods use this representation to define component and functional loss dependencies, in that the interactions that occur in the nominal state are not necessarily the interactions that occur in a failed state.

this chapter introduces the Flow State Reasoning (FSR) framework as a way to perform failure propagation analysis to include the effect and propagation paths of failures that cannot be represented in the nominal system representation. The aim of this work is to improve early design-stage failure analysis methods by expanding the scope of failures capable of being analyzed in the early design stage. Several of the methods developed for early design stage failure analysis provide useful information on the effect of system failures that can be used for design decision making. This type of information can be very effective in reducing the cost of redesign and can serve as a valuable metric for

design evaluation.

Figure 6.1 is a graphical overview of how the parts of the proposed framework integrate into an early design-stage failure analysis. The top of Figure 6.1 describes the failure analysis process using the FFIP approach [62]. Starting with a system component and functional representation, a library of generic component behavioral models is used to construct a system behavioral model. Using this system model, a variety of critical scenarios are tested, including single and multiple faults as well as parameter variations. Failure analysis results are reported in terms of impact to system functions as specified in the functional representation. That is, a loss or degradation of function. In the larger context of design, these results are used for identifying more robust designs and identifying weak links within the design. The role of the proposed FSR framework is illustrated on the bottom portion of Figure 6.1. Using expert knowledge, component behavioral models are modified to include fault symptom and fault cause information. This information is added to annotate the failure modes of components. Next, using a domain specific fault relational matrix an algorithm looks for potential connections in the system model that could occur as a result of failure events. These new fault scenarios expand the scope of the failure analysis. The benefit of modifying the failure analysis process with the FSR framework is the consideration of fault propagation that does not necessarily follow the nominal connections.

In order to implement the FSR modifications of the failure analysis a flow-based perspective of fault propagation is used. The following sections detail the basis for this view of fault propagation and how this can be used to reasonably evaluate potential connections within the context of early design failure analysis.

6.3.1 Establishing a Labeling Scheme for Potential EMS Flows

The first assumption of the FSR framework is that system elements, such as functions or components, interact with each other and the environment only through energy, material, or signal (EMS) flows. That is, in order for one element's state or behavior to affect another, there must be a real connection and that connection will be through an EMS flow. The concept of functional modeling is typically used to represent the process of changes to EMS flows and so each function will be connected to the previous and following functions by one or more EMS flows [119, 67]. In component representations

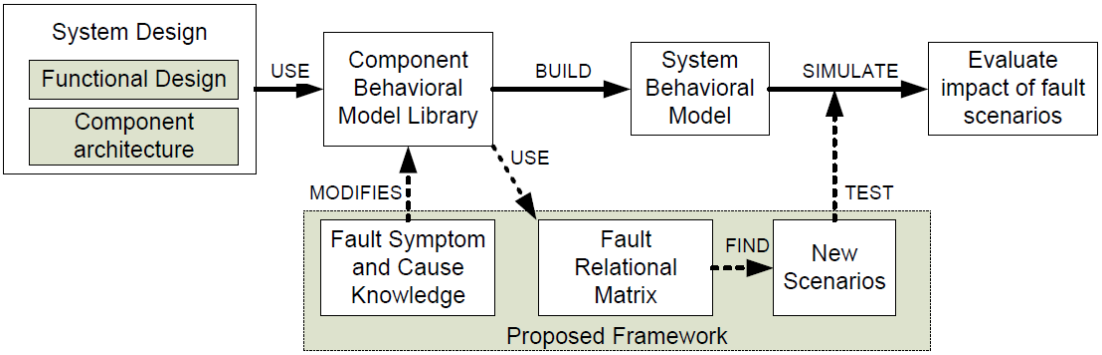


Figure 6.1: Overview of the Flow State Reasoning (FSR) Framework and its role in early design failure analysis.

connectivity is expressed with EMS flows. What follows from this assumption is that for failures that occur in an element and propagate through the system, the fault propagation path will always follow an EMS flow. Explosions, leaks, collisions, and electromagnetic interference are all examples of failures where the actual fault propagation path is along EMS connections that are not represented in the nominal state representation.

Failure events can often lead to unanticipated 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. A conservative approach is to consider that any flow between components and from a component is possible. Also any flow from the environment to the component is possible. It is therefore necessary to distinguish between designed flows and non-designed, or potential flows. Figure 6.2 illustrates the difference between designed flows and potential flows for a simple system. Potential flows are the cause and/or effect of certain failure events. In Figure 6.2 the arcs representing potential flows in the system are simplified to represent multiple EMS flow types.

One solution to the limitation of using the nominal state representation is to simply add all of the potential flows between system elements into the analysis. This faces two challenges. First, for functional representations, it is nonsensical for a function to have input flows that it does not act upon. This implies that potential flows must be represented on real, physical component representations. Secondly, representing all potential flows is confusing and conveys almost no meaningful information to the designers using

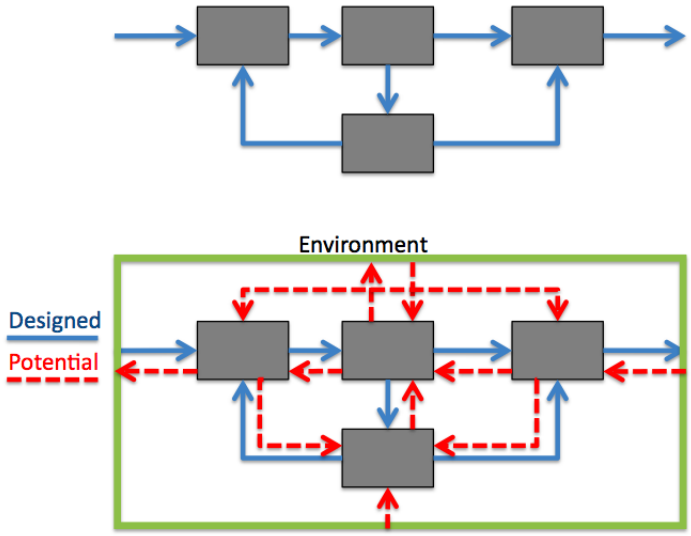


Figure 6.2: *Top*: Nominal-state design system representation. *Bottom*: EMS flows that potentially exist for a system.

the models. We propose using 'Flow State' as a guide for representation. That is, for a particular system state (nominal or failure) a potential flow or nominal EMS flow should be represented on the system model if it is in one of the following states:

- Normal Flow: 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.

This establishes a naming convention to categorizing the types of nominal and potential flows.

6.3.2 Identifying Potential Flows by Mapping Failure Modes

One clear limitation in considering any potential flow between components is the large number of propagation paths that a failure analysis method must investigate. For exam-

ple, consider the abstract system from Figure 6.2. There are seven EMS flows between components and inputs and outputs from the environment in the nominal system representation. However, potentially there can be EMS flows between each component in both directions and from and to the environment and each component. This increases the total number of nominal and potential flow paths from seven to twenty. This indicates that for more complicated systems the number of potential flows to consider must be reduced in order for a failure propagation analysis method to provide meaningful results. We limit the scope of analysis by restricting the investigation of the potential flows to those components that have matching fault causes and fault symptoms. This requires a comprehensive description of a failure with at least one cause and one symptom for each failure mode [175].

Fault causes are defined as events which can trigger a particular failure mechanism in a component. When a failure mechanism is applied to a component, a failure occurs and will result in some evidence. The evidence that a failure has occurred is defined as the symptom of a fault. For example, if the fault cause for a short-type failure in an electronic component is liquid material entering the component and a symptom of a leak-type failure in a valve component is liquid material, then the FSR framework would identify a potential EMS flow between these two components. This potential flow can then be used by a failure propagation analysis method when the valve leak failure is investigated, resulting in an expanded failure analysis. In a similar line of research [176] presented a method of identifying common cause failures. The main difference is that the approach presented in this chapter identifies dependencies between failures.

As a starting point, the FSR framework uses the fault cause-symptom association table found in Figure 6.3 to identify when a potential EMS flow might exist from a failed state. An **X** in this table indicates that the fault symptom has been known to precipitate the associated fault cause. This table was developed based on a generic approach to systems and therefore, for a particular domain, expert judgment should be used to both expand the types of fault causes and symptoms and identify potential dependencies. This approach identifies what potential connections exist in the failed state of a component and limits the scope of potential path analysis to those components with identified fault cause-symptom mappings.

Fault Causes	Related: symptom to cause	Shape Change	Surface Change	Temperature Change	Vibration	Noise	Material Leak	Solid Debris
High mechanical Load		X	x					
Impact					x			x
Open Circuit				x	x	x		
Short Circuit				x	x	x	x	
Surface Damage		x	x				x	x
Liquid Contamination		x			x		x	x
Gaseous Contamination		x			x		x	x
Solid Particle Contamination		x						x
Misaligned		x	x	x	x			
Loose Fit		x	x	x	x			
Tight Fit		x	x	x	x			x
High Temperature				x			x	
Low Temperature				x			x	
Overcurrent				x	x	x		
Overvoltage				x	x	x		
High Pressure		x		x			x	x
Low Pressure		x		x			x	

Figure 6.3: An initial fault cause-symptom mapping for identify potential fault propagation between components.

6.3.3 Extending a Design Stage Failure Analysis

Finally, the naming convention and mappings from the previous two sections must be interpreted with a failure analysis method to evaluate the impact of faults following new

potential fault propagation paths. Specifically, this is accomplished by directing a failure analysis method to evaluate the dependent fault propagation. During a fault scenario, when the failure analysis method triggers the failure mode of a mapped failure, first the single fault propagation is determined. Then the method reevaluates and triggers any related faults as identified in Figure 6.3. There are several factors that allow for tuning of the analysis: 1) The number of cascading failures the analysis tool will consider (one failure leading to another leading to another, etc.); and 2) The number of concurrent failures (one fault symptom relates to multiple fault causes). These tuning factors can be determined based on the fidelity of the model information.

In the following section the application of the FSR framework is demonstrated on a specific failure propagation analysis method. Namely, the Function Failure Identification and Propagation (FFIP) framework [71, 72, 62, 73, 74].

6.4 Application to a Failure Analysis Tool for Concept Evaluation

To illustrate the fundamentals and benefits of the Failure State Reasoning (FSR) framework for failure analysis, the Function Failure Identification and Propagation (FFIP) framework is used in this chapter as the basis for system representation and fault impact simulation. FFIP is a good candidate for applying the framework presented in this chapter because it focuses on capturing failure propagation and relating that propagation to the functional 'health' of a system.

As an illustrative application, we consider the conceptual design of a liquid-fueled rocket engine (LFRE). The goal of LFRE system 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 used. However, for this analysis the software controller is considered outside the boundaries of the system.

6.4.1 Early Design Failure Analysis Using FFIP

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. Figure 6.4 illustrates the different parts of the FFIP framework. Figure 6.4 provides the detail of the failure analysis approach described in Figure 6.1. The FFIP framework uses a set of system representations, and simulations that provide designers with the functional impact of failures. The system is represented with two static models, namely, a functional model and a component configuration model. The functional model contains the sequence of EMS flow transformations that occur in the system to meet the desired system requirements. High level functions are decomposed into subfunctions to a point where they can be implemented with components. Figure 6.5 illustrates the functional model developed to meet the system requirements.

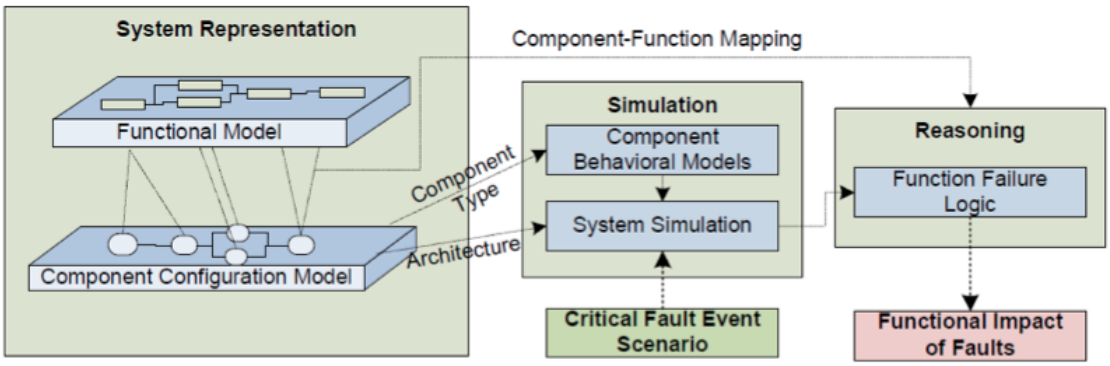


Figure 6.4: Graphical overview of the FFIP framework prior to the integration of the proposed Flow State Reasoning.

The component configuration model contains nodes and arcs, where the nodes represent generalized components and the arcs represent EMS flows. Generalized components are determined based on the possible implementations of functions from the functional model. In this way the functional model and the configuration model can be related by mapping functions to components. The system configuration and components used in this analysis are shown in Figure 6.6.

A simulation is created from these two static system representations. Each general-

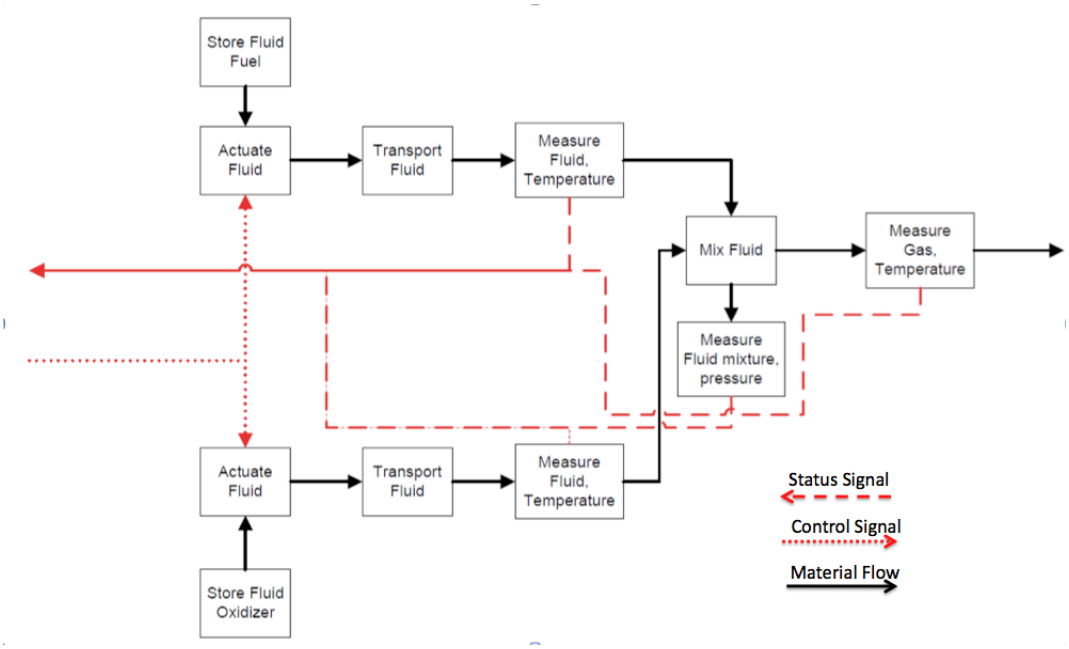


Figure 6.5: Functional model for a conceptual design of a liquid-fueled rocket engine.

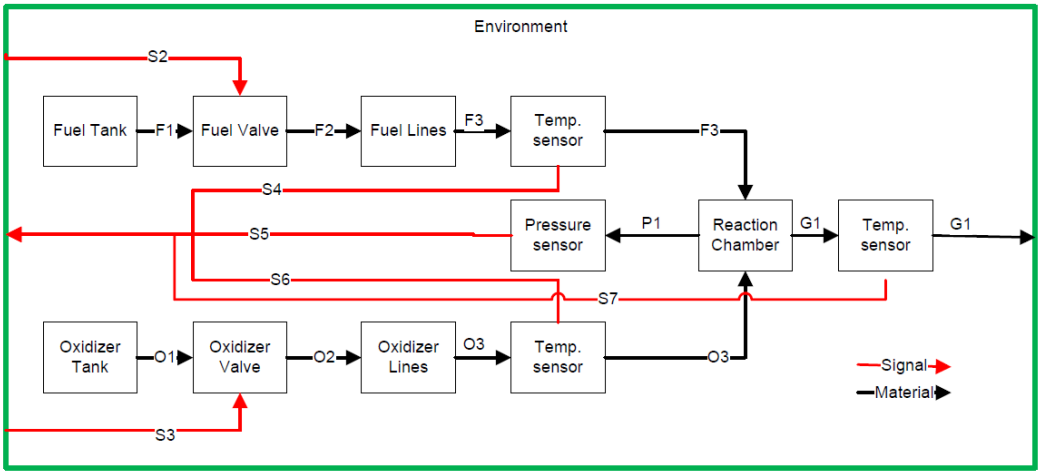


Figure 6.6: Component configuration model for a conceptual design of a liquid-fueled rocket engine.

ized component is represented as a behavioral model (a set of rules) that relates input and output EMS flows based on component mode. In the FFIP analysis framework, the component behavior is represented as the relationship between input and output flows based on component mode. EMS flows are discretized (High, Nominal, Low, ... etc.) and component mode changes affect the output EMS flow levels. Component modes include nominal operating and failure modes. A system simulation is created by combining generalized component behaviors and initiating some input EMS flows. The key part of the simulation is the Function Failure Logic (FFL). For each generalized component a set of logical rules is created to relate the input and output of the component behavioral model to the state of the function (or functions) being realized by that component. Function states are defined as operating, lost, or degraded and reflect the system's ability to carry out that function in the given state. Analysis using this method involves inputting into the simulation a set of critical event scenarios which results in finding the state of all functions in the system as determined by the FFL rules. For example, one critical fault scenario that may interest the designers is the impact of a fuel tank leak. To identify the impact with the FFIP-based simulation the mode of the component "Fuel Tank" is changed to that failure mode. The system behavioral simulation occurs over many time steps. The effect of this change is that first the fuel flow level supplied by the tank is incrementally reduced according to the failure behavior associated with that failure mode. The FFL identifies the functional impact as: the *supply fuel* function is degraded and the *combust fluid* function as degraded. The latter occurs because the behavioral model for the reaction chamber component outputs a nominal flow only when the two input flows are equal.

This analysis provides a designer with the impact of a failure when it propagates through a system in terms of functional losses. As with the other methods mentioned in Section 6.2, FFIP is limited in scope. Specifically, the number of potential failure paths that the method can investigate is limited to the EMS flows in the system's functional and configuration models. The following sections demonstrate how to exapnd the scope of analysis based on the FSR framework.

6.4.2 Reasoning About the State of Flows in a System

As mentioned, the FFIP analysis approach provides the functional impact of faults in terms of the state of each low-level function over the simulation time steps. Fault propagation can be inferred from this based on function effected and how they are connected. Using the previous fuel tank leak example, the propagation path begins at the leak and terminates at the reaction chamber and is assumed to follow the nominal component connections. However, to consider alternative (off-nominal) paths the propagation path needs to be explicit. This is achieved by including a reasoning module to evaluate the state of flows in the system, based on the FSR assumption that all propagation occurs through EMS flows. This reasoning module is termed Flow State Logic (FSL) because it serves a similar to purpose the Function Failure Logic (FFL). Figure 6.7 illustrates an example of the logic rules that the FFL uses to identify function state and the logic that the FSL reasoner uses to identify the state of designed and potential EMS flows in a system.

One difference between FFL and FSL modules is that while FFL is specific for each component implementation of a function there are only two types of FSL. One set of rules is used to evaluate the flows that exist in the nominal connections. This is the "as-designed" FSL which can be added to the simulation reasoning directly. While the other set of rules captures potential flow states and is triggered when the reasoner described below generates a scenario based on a cause-symptom mapping.

6.4.3 Expanded Behavioral Models

To conduct an FFIP analysis no information regarding fault causes or symptoms is necessary, only a knowledge of potential fault modes and their impact on behavior. To conduct the FSR modified analysis, these behavioral models must include this additional failure information. Specifically, the causes and resulting symptoms of faults must also be added as annotations the failure modes. The potential flow connection reasoning requires this information to trigger fault propagations during fault simulations. This information reflects a higher level of necessary system knowledge than is necessary for representation and simulation in the original framework. Figure 6.8 shows an example of this added information to the component behavioral models of the original FFIP models. In this

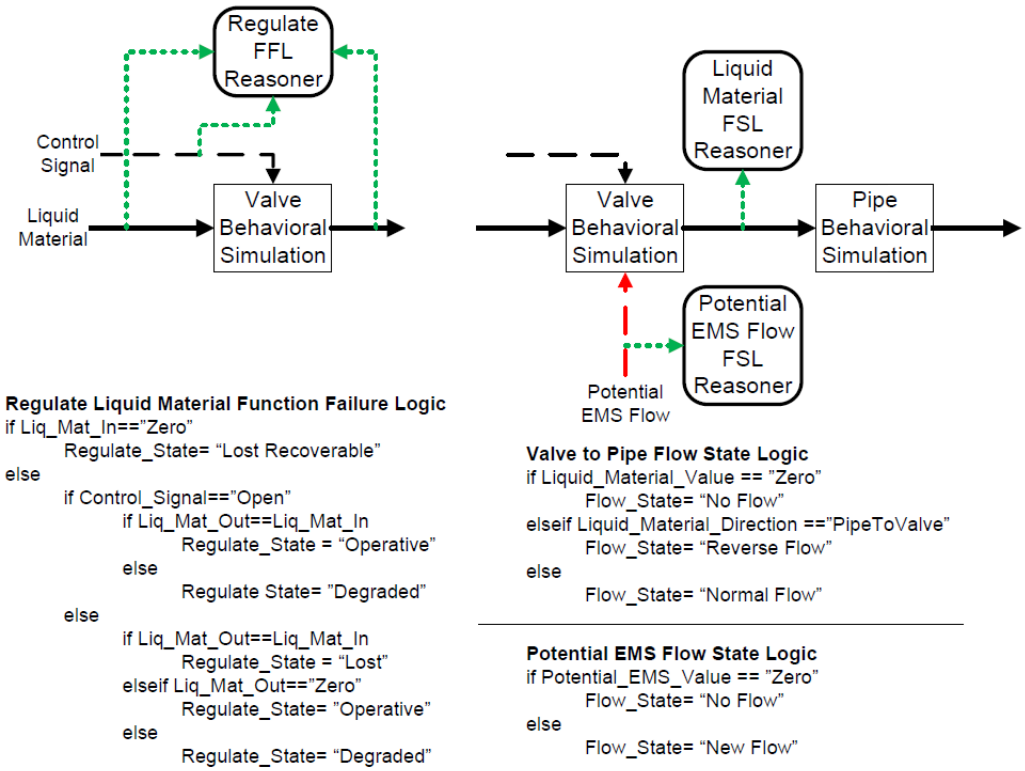


Figure 6.7: *Top*: The FFL reasoner inputs as compared to FSL reasoner inputs. *Bottom*: The logic used in the FFL reasoner as compared to the logic used in the FSL reasoners.

figure, the fault causes and symptoms are only shown for one fault mode. This process is similar to what is done when conducting an FMEA analysis. However, the advantage of this approach is that only immediate causes and symptoms of faults must be identified rather than the system or subsystem level effects. The system and subsystem level effects are determined from simulation. This information is not difficult for a designer who is familiar with the components and the environment it is operating in. For example, it is expected that a designer would know that wear is a cause of certain bearing failure modes and that a number of symptoms such a vibration may occur for that failure mode. With annotations, this information can be added to a component model library and stored for reuse.

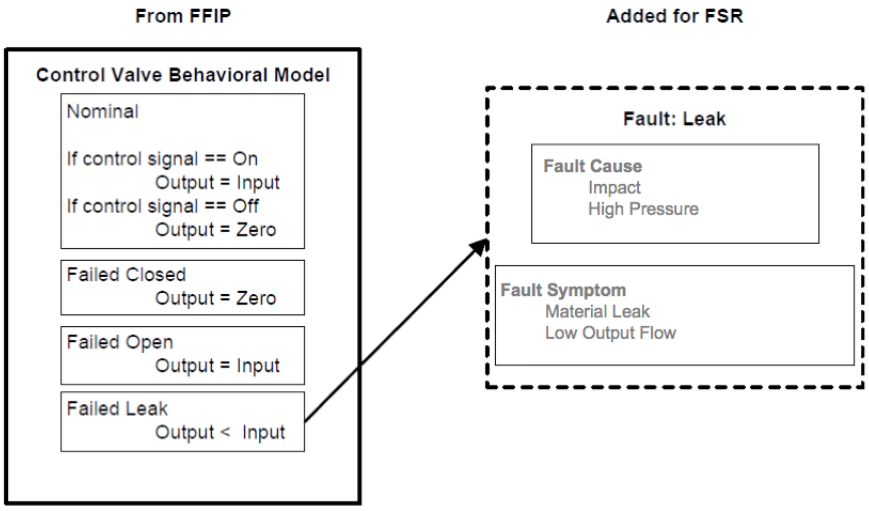


Figure 6.8: An example behavioral model from the original FFIP and the necessary addition of fault causes and symptoms.

6.4.4 Fault Simulation and New Propagation Paths

For this analysis the modeling and analysis tool ModelCenter by Phoenix Integration Inc. [177] was used to simulate the system. ModelCenter provides a means of wrapping component behavior models that may be constructed using different tools (such as Excel, Matlab, or many others) so that they can be compiled together for a system model. Additionally, subsystem models can be combined into an assembly with other subsystem models for more complex systems. For comparison purposes the same set of critical events were given to an unmodified FFIP simulation and an FFIP simulation that was modified based on the FSR framework. The process of fault simulation and functional impact determination using the FFIP framework has been presented in previous research [71, 72, 62]. For clarity, one example fault simulation is demonstrated.

One known failure mode of the fuel line component in the LFRE example is a leak. In this mode the behavioral simulation of the fuel line causes the output fuel to be less than the input fuel. Using the non-modified simulation the whole system is simulated for nominal operation except for the seeded fuel line fault. The system level effect is an unbalanced mixture of fuel and oxidizer in the reaction chamber. The FFL reasoner

identifies the functional impact as a high-level loss of the *Provide Thrust* system function because of the degraded *Transport Fuel* and the lost *Combust Mixture* subfunctions. A similar analysis is executed with the modified simulation. However, the FSR framework provides additional analysis capabilities. The fuel line leak fault is a result of a potential solid material flow to that component from the environment. In the modified simulation, the solid material flow is injected as a fault which causes the fuel line leak. This leak causes the same fuel/oxidizer imbalance in the reaction chamber. Additionally, the fuel that is leaked from the fuel line is now a potential flow in the system. The potential propagation reasoner identifies this fault symptom as matching the fault cause of failure modes in the sensor components. Therefore, the propagation reasoner executes the failure analysis for each sensor fault triggered by the fuel line leak symptom of an extraneous flow. One of these analyses identifies the fuel line temperature sensor component failure. The FFL reasoner for this modified simulation identifies that the same system level functional loss occurs as before, however, the subfunction of *Measure Temperature* is also lost (and similar results the other potentially affected sensor components). Fault propagation paths are created based on the end scenario to component states and the potential flows identified by the Flow State Logic. For example, the FFIP results of the above failure would show that the fuel valve leak failure propagates to the reaction chamber, whereas the FSR modified analysis would show four sets of failure propagations (one for each triggered sensor fault). Each would be similar in that the fuel valve leak fault propagates to the reaction chamber as well as to a sensor, which is the fault path that is identified by the Flow State Logic output.

6.4.5 Discussion of Results

Using only the FFIP analysis framework the functional impact for critical scenarios in the LFRE example can be assessed for 35 unique single faults. These faults represent the total set of known failure modes of the components. The FFIP analysis framework is also capable of evaluating the impact of multiple faults. To demonstrate the impact of the FSR modification of the analysis framework only single faults were considered in the analysis. While the results could be used to make design comparisons or other decisions [72], this analysis does not provide a complete view of potential functional state changes. The benefit of the FSR framework is seen by comparing the unmodified and modified

simulations. Implicit with this result is the propagation path that a fault followed in the behavioral simulation. Table 6.1 shows the functional impact in terms of functions lost or degraded and the fault propagation path for each single fault.

Applying the FSR framework identifies several potential failure propagations based on matching fault causes and symptoms between component failure modes. For example, in the designed system the fuel and oxidizer valves, the fuel and oxidizer lines and the reaction chamber all have leak-like failure modes that output extraneous material flows as a symptom of that failure. Additionally, a fault cause for the sensors to fail is the presence of these types of materials. These potential failure paths as well as others can be identified by the modified analysis and the functional effect of these propagations can be determined using the FSR framework. Table 6.2 shows some of the impacts and propagation paths for these new failures. By limiting the analysis of potential EMS flows to those which can cause a failure in the system components the modified simulation was able to investigate the effect of 27 additional potential flows. 11 potential EMS flows were identified as coming from the environment to components and 16 potential EMS flows were identified from components with leak type failure modes to components that can be affected by the material that was leaked. The potential flows from components to the environment were not considered.

As illustrated in Section 6.4.4, a significant benefit of modifying FFIP with the FSR framework is seen in the ability to more fully analyze cascading failures. The ability to simulate multiple faults and determine functional impact is one of the main benefits of the FFIP method. With the modifications from the FSR framework multiple fault simulation occurs when failures in one component cause a fault in another component. This cascading fault approach is a logical means of assessing multiple fault scenarios. Previous methods have used a random selection of components and expert knowledge to determine which components to use in a multiple fault failure simulation [72]. The result of multiple fault simulations are affected by both the component fault choice and the relative timing of those faults. Therefore, the FSR framework provides the sequencing and identifies the components affected in a cascading failure that causes multiple faults.

The LFRE conceptual design is a fairly simple system and it helps to illustrate the benefits of the FSR framework. However, it also imposes three limitations. First, the functional effect of a leak type failure is simply the combination of the leak failure and the effect of the sensor failure. This result occurs because the valve control is not considered

Table 6.1: Functional losses found by the original FFIP simulation.

Component Failure	**Functions Affected**	**Effect**
Fuel Tank Empty	Store Liquid Fuel	lost
	Actuate Liquid Fuel	lost recoverable
	Transport Liquid Fuel	lost recoverable
	Combust Mixture	lost
Fuel Tank Leak	Store Liquid Fuel	degraded
	Combust Mixture	lost
Fuel Tank Over Pressurized	Store Liquid Fuel	degraded
	Combust Mixture	lost
Oxidizer Tank Empty	Store Oxidizer	lost
	Actuate Oxidizer	lost recoverable
	Transport Oxidizer	lost recoverable
	Combust Mixture	lost
Oxidizer Tank Leak	Store Oxidizer	degraded
	Combust Mixture	lost
Oxidizer Tank Over Pressurized	Store Oxidizer	degraded
	Combust Mixture	lost
Fuel Valve Stuck Closed	Actuate Liquid Fuel	lost
	Transport Liquid Fuel	lost recoverable
	Combust Mixture	lost
Fuel Valve Stuck Low	Actuate Liquid Fuel	degraded
	Combust Mixture	lost
Stuck High	Actuate Liquid Fuel	degraded
	Combust Mixture	lost
Fuel Valve Leak	Actuate Liquid Fuel	degraded
	Combust Mixture	lost
Oxidizer Valve Stuck Closed	Actuate Oxidizer	lost
	Transport Oxidizer	lost recoverable
	Combust Mixture	lost
Oxidizer Valve Stuck Low	Actuate Oxidizer	degraded
	Combust Mixture	lost
Oxidizer Valve Stuck High	Actuate Oxidizer	degraded
	Combust Mixture	lost
Oxidizer Valve Leak	Actuate Oxidizer	degraded
	Combust Mixture	lost
Fuel Line Blocked	Transport Liquid Fuel	lost
	Combust Mixture	lost
Fuel Line Leak	Transport Liquid Fuel	degraded
	Combust Mixture	lost
Oxidizer Line Blocked	Transport Oxidizer	lost
	Combust Mixture	lost
Oxidizer Line Leak	Transport Oxidizer	degraded
	Combust Mixture	lost
All Temperature Sensor Failures	Sense Temperature	lost
All Pressure Sensor Failures	Sense Pressure	lost
Reaction Chamber Leak	Combust Mixture	degraded

Table 6.2: Example set of new failures investigated by adding the FSR logic to the FFIP simulation.

Failure	**Functions Affected**	**Effect**
New EMS Flow from Environment to Components		
Solid Material to Fuel Valve	Actuate Liquid Fuel	degraded
	Combust Mixture	lost
Solid Material to Oxidizer Line	Transport Oxidizer	lost
	Combust Mixture	lost
Electrical Energy to Pressure Sensor	Sense Pressure	lost
Component Failures Caused by Component Leak Failure Modes		
Fuel Line Leak to Pressure Sensor	Transport Liquid Fuel	degraded
	Combust Mixture	lost
	Sense Pressure	lost
Reaction Chamber Leak to Temperature Sensor	Combust Mixture	degraded
	Sense Temperature	lost
Fuel Valve Leak to Oxidizer Temperature Sensor	Actuate Liquid Fuel	degraded
	Combust Mixture	lost
	Sense Temperature	lost

in this analysis. The control subsystem actuates the valves according to sensor inputs. This means that if a failure affects the sensors, such as in the leak cases shown, then that fault may also propagate through the control subsystem and back to the initiating component like a leaking valve. The control subsystem was not included in this analysis for two reasons. First, without a control system to mitigate the effect of a failure, fault propagation through the hardware is more readily apparent. Secondly, preliminary research of including the control subsystem in the analysis has shown that the results are most useful for optimizing the control system [71]. Using these results to develop better prognostics and health management strategies is an area of future work for this research [71]. The second limitation from using this simple conceptual design is that all potential flow induced faults that were investigated occur as initiating faults. The consequence is that the functional impact found for a new solid material flow causing a blocked flow in the fuel line is identical to simply investigating the blocked flow failure mode of the fuel line. However, this is not generally the case for more complex systems. The advantage of using the FSR framework to enhance the analysis is seen when failure induced EMS flows occur as part of the fault propagation. For example the functional impact of some component failure leading to the generation of a new solid material

flow that then causes the fuel line blockage is likely to be different than the addition of the functional impact of the two failures investigated individually. Finally, the third limitation imposed on this analysis was that components were specified such that the failure induced EMS flows did not propagate to nominally connected components. For many types of potential EMS flows the propagation to nominally connected components is likely. For example, vibrations rarely affect a single component without propagating to other physically coupled components.

Despite these limitations, the example system demonstrates the benefits of using the FSR framework to enhance an early design-stage failure propagation analysis. The limitation imposed by using the designed system representation to model fault propagation is overcome with the addition of potential EMS flows to the nominal state representation. Further, the mapping of fault causes and fault symptoms using behavioral models that include nominal, failed state and potential flow environment behavior provides insight into additional system failures that might have been missed if this were part of a more complex system.

6.5 Summary

This chapter addresses the limitation of using nominal system state representation based design models in the analysis of failure propagation. This chapter introduced a framework that can be used to enhance failure analysis methods to include the effect of potential energy, material, and signal (EMS) flows that may occur in a failed state. Identifying the dependencies between component faults and functional states is critical for design stage failure analysis. The Flow State Reasoning (FSR) framework was introduced to address this challenge. First a labeling scheme was created to describe potential and nominal EMS flows in a system representation. Second, the FSR framework was shown to map component fault causes to failure symptoms to facilitate identifying which of all the potential flows that were identified might actually exist as a result of a failure. When the fault cause of a component failure is also the symptom of a different component failure, then a failure analysis method that is enhanced by integrating with this framework will investigate the impact of potential EMS flows between these components during a failure.

Further investigation is necessary to find the best approach to optimizing the po-

tential connection reasoning. One limitation to this approach is the inability to specify the timing of fault propagation. Fault propagation is shown sequentially without an assessment of delay between fault causes and effects both along potential and designed propagation paths. Further, it may be advantageous to investigate failure from a top down perspective, to find all potential faults and fault propagation paths that lead to a specified end functional state.

The FSR framework is designed to be used to guide the modification of design stage failure analysis methods by including faults and failure propagation along potential EMS flows the can occur in a failed state. However, the philosophy behind the FSR framework is also applicable to other reliability methods. The two-fold philosophy behind the FSR framework is, first, that all components are connected with potential EMS flows, and second, that some failures may propagate through a system along these flows which are not generally represented in system models. For example, FMEA is a standard reliability analysis for several domains. When an FMEA is generated with software that is automatically or manually created with expert input, the assessment of the effect of failures can also be limited by the system representation that is being used. Because many real failures do not propagate through a system according to designed component connections, failure analysis should take into account the potential dependencies that can result from a failure. Risk analysis methods that begin with this type of thorough failure analysis have the potential to provide designers and other decision-makers with the information necessary for designing and operating safety-critical, complex systems.

Chapter 7: A Safety Function Failure Reasoning Method

The previous two chapters presented a quantified evaluation of the effect fault scenarios on system functions and demonstrated a method for identify fault dependent scenarios. Together these two form an analysis of the fault behavior under a large set of potential scenarios. This analysis is capable of identifying the state of component-level functions during a failure simulation by using the function-to-component mappings and function failure logic. Thus, this approach provides a design evaluation of functional robustness or functional reliability (if failure mode probabilities are considered). However, this approach cannot provide designers an evaluation of how "safe" a design is. As discussed in Section 2.3, safe designs and reliable designs are not the same goal. Therefore, this chapter presents the reasoning method for evaluating the state of safety functions as defined in Chapter 4. This approach uses a system simulation built following the design process developed in Chapter 4 and uses a set of logical rules to identify if a safety function is lost or inoperative. The result of implementing this approach is an analysis of the ability of a design to remain in a safe state during critical faults scenarios, thus enabling safety-based design decision making.

7.1 Background

The approach presented in this chapter is based on the design methodology presented in Chapter 4. This approach follows the Systems-Theoretic Accident Model and Processes (STAMP) [134] which describes the occurrence of an accident as possible through the violation of a safety constraint. Further, Leveson et al. developed a design stage methodology based on the STAMP model called the Systems-Theoretic hazard and Process Analysis (STPA) guide and evaluate designs for safety [84]. As detailed in Section 4.1.3, the STPA method guides the designer to generate control structures to identify safety constraints. 14 types of general ways the control structure can fail serve to guide the system designers in identifying potential failure scenarios. Leveson offers a metric of safety evaluation for a design based on how difficult a scenario is to mitigate, however,

the probability of that scenario occurring is said to be impossible to determine [84]. In contrast to the STPA approach of guiding the designer to identify scenarios this chapter presents a method of simulating scenarios and using reasoners to identify the control structure violations.

7.2 Safety Function State Reasoning

As mentioned in section 4.1.3, the STAMP approach presents four ways that a safety constraint can be violated. To repeat these are [134]:

1. A control action required for safety is not provided or is not followed.
2. An unsafe control action is provided that leads to a hazard.
3. A potentially safe control action is provided too late, too early or out of sequence.
4. A safe control action is stopped too soon (for continuous or non-discrete control actions).

As part of the STPA methodology for identify scenarios where one of these unsafe control actions may occur, Leveson presents 14 types of control structure failure [84]. These 14 factors are illustrated in Figure 7.1. We propose that by mapping the implementation of the safety function to the system architecture that a reasoning approach can be applied to the system simulation to identify the *state* of the safety function. The state of the safety function is either "operating" or "lost". The safety function is "lost" when any of the 14 factors exists. Therefore, we have interpreted a set of the 14 factors into rules that can be applied to the system simulation to reason on the state of the safety function. A safety function being lost does not mean that the associated mishap will occur. Rather it indicates that the barrier against the transition to the mishap state is lost. Ultimately this will serve as a means of evaluating design safety by identifying designs where the barriers to mishap states are more difficult or more unlikely to be lost.

In the FFIP approach, the state of functions is reasoned about using Function Failure Logic (FFL) on the parameter values in the behavioral simulation (see Chapter 5). In the same way, the state of the safety function state can be reasoned on by evaluating the behavior of components and the value of parameters in the system simulation. In Table 7.1 the rules for identifying when a safety function is lost are presented for the general

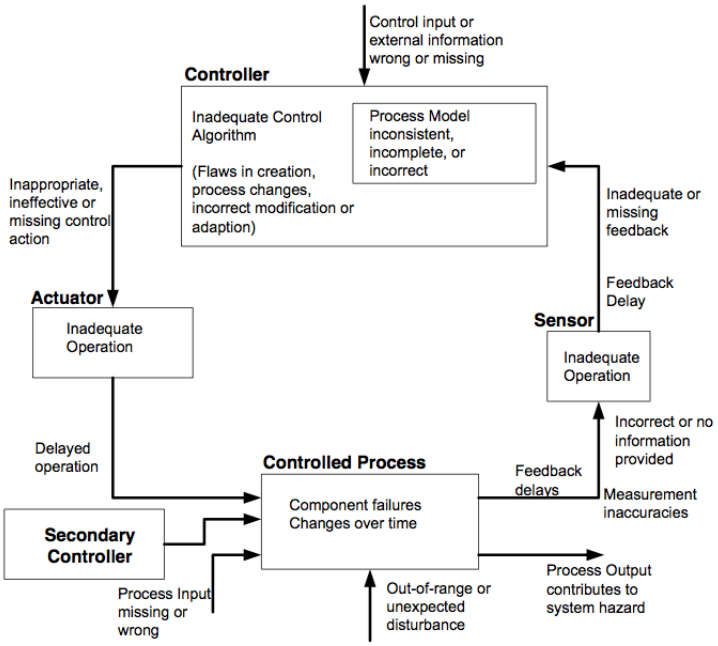


Figure 7.1: Generalized factors that are used to identify potential safety constraint violation scenarios in the STPA methodology [84].*Repeated for clarity.*

case (matching Figure 7.1). These rules are separated into two classes, flow based rules and component based rules. A flow based rule is reasoned on based on the flow between components where a component rule is based on the state of a component.

As can be seen in Table 7.1, the rules for identifying safety function state depend on a variety of variables. For example the first and second rules look at the value of the control signals in and out of the controller. Using the behavioral modeling in the FFIP approach, the state of flows are defined in qualitative intervals. In the case where there is no value, as might happen with a controller failure, the output flow of the behavioral model is NaN (or not a number). Alternatively, the Flow State Reasoning (FSR) logic would identify this flow as *No Flow* (see Chapter *ChatperFSR*). Rules 3, 6, and 8 utilize part of the simulation parameter, namely the simulation time to evaluate safety function state. These rules identify an X that is greater than the expected simulation change time. This value is dependent on the modeling tool chosen. For example, some tools can

simultaneously change each state in the component at the same time step (simulation time models real time) while other tools complete one state change for each time step (simulation time represents processing time). For the latter type of tool, the X would depend on the number of components in the system while X with the other type of tool would be a very small number. Finally, the component type rules look at the FFL for the functions implemented by the control structure components to identify the state of the safety function.

7.3 Example Application

To demonstrate the application of the safety function state rules in a system we utilize the safety-guided design presented in Chapter 4. Chapters 5 and 6 provide the details of the electrical power and thruster subsystems as well as the simulation and failure scenario identification process. As depicted in Figure 4.9, the safety function for this example prevents the system state change where the orbital trajectory is permanently lost. While there is a large set of system failure scenarios that will not result in the loss of this safety function, we present two example scenarios where the safety function is found to be lost by applying these rules to the system simulation.

7.3.1 Case 1: Safety Function Control Structure Component Failure

In this critical scenario we consider a component failure mode in one of the components used for implementing the safety function control structure. Considering broad failure rates can be identified for early component failures [178], this type of scenario can provide a first-pass reliability estimate of the safety function control structure. For example, hydraulically driven valves exhibit a failure mode where a pressure leak causes slower valve movement. One design decision that might be made is whether to use hydraulic valves or electromechanical actuators for controlling fuel and oxidizer valves in the thruster subsystem. Electromechanical actuators do not exhibit this leak-caused slow behavior failure mode. As illustrated in Figure 4.9, the valves are mapped to the actuator part of the specified safety function control structure.

In the system simulation a scenario is tested by initializing the system in a nominal state and triggering one or more mode changes. For this example, the fuel valve mode

Table 7.1: Safety Function State Reasoning Rules

#	**Safety Function Control Structure Fault Type**	**Simulation Rule Interpretation (safety function lost if:)**
Flow Based Rules		
1	Control inputs wrong or missing	Control_signal_in == NaN (not a number)
2	Inappropriate, ineffective or wrong control action	Control_signal_out == NaN
3	Delayed actuator operation	ΔT is greater than X between change in Signal and change in Actuator state
4	Process Input missing or wrong	Process_in_Flow==*Zero*
5	Unidentified or out of range disturbance	FSR recongizes *New Flow* (see Chapter 6)
6	Actuator Feedback delay	ΔT is greater than X between change in Process state and change in Signal
7	Incorrect or no information provided	Sensor_signal_in==NaN
8	Sensor Feedback delay	ΔT is greater than X between change in Sensor state and change in Signal
9	Inadequate or missing feedback	Signal==NaN
Component Based Rules		
10	Proccess model inconsistent, incomplete or incorrect	Computation FFL==*Degraded* or *Lost*
11	Inadequate Actuator operation	Actuator FFL==*Degraded* or *Lost*
12	Component Failures and changes over time	Process FFL==*Degraded* or *Lost*
13	Inadequate sensor operation	Sensor FFL==*Degraded* or *Lost*

changes from nominal to the slow behavior failure mode. The thruster controller actively opens the fuel and oxidizer valve to result in equal flow states. When the controller directs the fuel valve to open to the same state as the oxidizer valve (a change from half open to full open), the component behavior specified for the fuel valve in that failure mode results in a delay action. Because this is an actively controlled system, the controller response is to reduce the oxidizer valve opening to match since it appears to the controller that the fuel valve cannot be changed (the response is too slow). The result is that oxidizer valve responds but now the fuel vale has changed and the result is an unequal mixture. The FFL reasoning in this simulation reports that the function of the thruster *to combust* the mixture is degraded due to the incorrect mixture. While the FFL provides the functional impact of the scenario, the safety function state rules identify the impact to system safety.

When the safety function state rules described in Table 7.1 are applied to this scenario simulation it is clear that rule 3 will identify the safety function as being lost. This means, that the property of the system which would inhibit the loss of orbital trajectory is jeopardized by this component failure mode. In a design decision process, a comparison of electromechanical actuator failure mode scenarios and hydraulically driven valves can be conducted and the result using the safety function rules can provide designers with an estimate of the safer design.

7.3.2 Case 2: Fault-Dependent Failure Scenario

Following the component annotation process described in Chapter 6, the failure mode causes and symptoms are included in this analysis. The result of applying the FSR algorithm is an identification that there is a potential fault-based connection between the fuel lines in the thruster subsystem and inverter in the electrical power system. This is found because one failure mode of the fuel lines is a leak, which exhibits the symptom of a material leak. Additionally, one of the potential causes identified for the electrical short failure mode for the inverter in the electrical power subsystem is liquid contamination. Using the cause-symptom mapping found in Figure 6.3, a failure scenario is identified where the fuel line failure mode is triggered and subsequently the inverter failure mode is triggered.

The result of this scenario is that inverter failure causes a loss of electrical power

to the controller and the thruster system. This represents a complete loss of operation and the FFL reasoner identifies the functional impact as *Lost, No Flow* for most of the functions in this system. Multiple safety function state rules indicate the loss of the safety function. For example, the component based rules all indicate a loss of the safety function based on the component FFLs.

This scenario demonstrates that components in different subsystems may interact in certain failure scenarios and that the result of this interaction may result in a lose of the safety function. Further, neither of these components that interacted where part of the identified safety function control structure. In the first scenario it would be possible to identify a loss to the safety control structure by a loss of one of its components, the loss from this scenario required propagating failed behavior through the system.

7.4 Summary

In this chapter we have developed the rules for identifying the state of the safety function when the safety-guided functional design process of Chapter 4 is used. Further, using these rules we have identified the impact in terms of safety of component failure scenarios and fault propagation through system behavioral simulation. Specifically, we present 13 rules that evaluate the flow values, the simulation time, and the component FFL to identify is a safety function is lost. In this work, when a safety function is lost it implies that the barrier to a mishap state transition is lost and does not necessitate a resulting accident state.

The presented approach uses simulation to arrive at the factors which may lead to an accident. In contrast, the STPA approach presented factors as a guide for safety engineers to identify potential scenarios. While the same factors are used, some aspect of those factors cannot be evaluated through the current early design behavioral simulation. Specifically, a set of the identified factors include “inappropriate” or “inaccurate” signals. At the abstraction level implemented for the system behavioral model, these evaluations are not possible. Thus the safety function rules presented in this chapter will not identify these issues. Therefore, this approach is not intended to replace the expert identification and review of potential scenarios but rather augments and expands the capability of the expert-based approach with behavioral simulation.

This chapter is the conclusion of the analysis aspect of the safety-centric design

decision enabling framework. This chapter and the two previous chapters have presented a means of quantifying the impact of failures in terms of their functional effect at a high abstraction level. This enables fault simulation and analysis of cyberphsycial systems early in the design process and indicates that design decisions can be made based on the quantified effect of that simulation. Secondly, it has been shown that fault dependent scenarios can be identified and simulated and that the analysis of the cyberphsycial system is not limited to the nominal connected fault propagation assumptions. Finally, this chapter completes the analysis by showing how the simulation behavior can be evaluated to reason on the state of the operation of safety functions in the system. The presented analysis approach bridges the gap between hazard analysis and component fault analysis to provide designers with an early assessment of design safety. What has yet to be addressed is how this analysis can be used for design decision making.

The design decision making process is not as simple as choosing one design over another based on a single evaluation. Rather, it is a multi-objective satisfaction process where designs have multiple competing attributes, even within the scope of failure and safety evaluations. Therefore, the next chapter presents several means of understanding the analysis results. It is through these views of the system that safety-centric design decision making is enabled.

Chapter 8: Clustering Failure Analysis Results to Enable Design Decision Making

This chapter demonstrates three techniques for comparing and clustering the results of a function based failure analysis. This work provides alternative methods for using the results of fault simulations which have been performed based on the approach of the last three chapters. Specifically, this chapter illustrates how to group scenario results to address highest risk, a probabilistic classification scheme to abstract system failure behavior, and cluster results in terms of similar functional effect. These three methods are demonstrated on the analysis of an electrical power systems which represents the electrical subsystem of the safety guided framework. The content for the chapter was accepted to the conference of the American Society of Mechanical Engineering Computers and Information in Engineering Conference and was cowritten by David Jensen, Christopher Hoyle, and Irem Y. Tumer [179].

8.1 Introduction

Risk analysis has the greatest impact on system design when it can be incorporated into the early design-stage and be used as a decision making tool. In this capacity risk can become an attribute of the design and be used in architecture and component selection. The challenge of risk assessment at the design stage is the lack of refined system information. Traditional methods of failure and risk analysis rely on statistical failure data and apply methods where expert knowledge of the system is needed to know the impact and path of fault propagation. For this reason, risk assessment traditionally occurs at the validation stage of a well refined design, where specific component failure probabilities and likely fault propagation paths can be defined. However, to achieve the benefits of early risk-based decision making several methods of failure analysis using the abstract functional approach have been developed. While some of these design-stage methods use historic failure rates associated with the component types or functions to identify risk [64, 65], others have used a behavioral approach to determine the potential

impact of failures [72, 63, 69]. The benefit of using a behavioral approach is the ability to simulate fault propagation and identify the effect of a fault within the context of the designed system.

Early design-stage failure analysis is a powerful decision making tool allowing designers to make changes to decrease the risk of single, multiple or cascading faults [72]. However, the functional approach also enables a high degree of failure characterization. Understanding the ways in which a system design can fail provides designers with the information to develop more robust alternatives. Further, design iteration driven by risk must address the overall response to failures and not focus on only single fault risk improvement.

The Function Failure Identification and Propagation (FFIP) framework is one of the methods for assessing the functional impact of faults in the early design stage [62]. The results of using an FFIP-based analysis of a design is an evaluation of the state of each function in the system in response to a simulated failure scenario. In previous work these results have been used to evaluate the consequence of different fault scenarios for a system design [71, 180, 181] and for making design decisions based on fault consequence [72]. In contrast, this chapter focuses on different methods of evaluating a large set of failure scenarios to explore a design and develop design decisions that reduce overall system risk.

Specifically, this chapter demonstrates three different methods of evaluating and grouping the results of simulated failure scenarios and identifies how each method provides insight into the failure characteristics of a design. The first method evaluates the cost of failures and generates groups such that the designer can prioritize failures and make design changes to reduce the risk of those failures. The second method uses a statistical analysis method, Latent Class Analysis, to identify the emergent failure behavior categories that a design exhibits. Groupings from this approach are used to characterize a designs based on high-level failure behavior. Finally, an evaluation of functional-impact similarity generates groups of failure results based on the similarity to a particular functional state of interest. These groupings can be used to identify identical and similar faults to ensure that designers address all the faults that might lead to that state or a very similar one. Overall, the goal of this work is to provide a framework to understand and explore potential failures in the early design stage, providing designers with the ability to make risk-informed design changes to improve system reliability.

8.2 Background

In the previous chapter, the failure analysis process was detailed. In this section a short summary of the failure analysis method is provided as well as a background on the statistical bases for one of the clustering methods employed in this approach. The focus of this current chapter is the development of the clustering approach applied to the results of analysis using the Safety-Centric Function Failure Identification and Propagation (FFIP) framework.

The Function Failure Identification and Propagation (FFIP) framework [62, 73, 71, 72, 74, 182, 180, 183, 181] was introduced 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 to identify failure propagation paths by mapping component failure states to function 'health'. This approach uses simulation to determine fault propagation and fault effect, thus providing the designer with the possibility of analyzing component and interaction failures and reasoning about their effects on the rest of the system. The two main advantages of the FFIP method are: 1) a functional abstraction which allows it to be used in complex systems employing both software and physical components; and, 2) a simulation-based approach allowing analysis of multiple and cascading faults.

An FFIP analysis begins with a functional representation of a system and utilizes the mapping of functions to components in a component structural representation. A system simulation is built following the structural representation. The nominal and faulty behavior of generic components are stored as state machines in a component library. Each state represents a behavioral mode of the component where the qualitative intervals (high, low, etc.) of the input flow attributes are converted to output flow attributes. For example, in the nominal mode of a fuel line the input flow level of fuel is the same as the output. However, in the blockage fault mode the output flow level is reduced to zero. Finally, the main contribution of the FFIP approach is the Function Failure Logic (FFL) reasoner which relates the input and output attributes of the component simulation to the expected change for the function mapped to those components. The result of an FFIP analysis is an evaluation of the health status of each function in the system. There are four potential health states for a function defined as:

1. Healthy: The function affects the flow as intended
2. Degraded: The function affects the flow differently than intended
3. Lost: The function does not affect the flow
4. No Flow: There is no flow for the function to act on (usually due to an upstream failure)

A failure event, which is the triggering of one or more component transitions, is simulated and the functional impact as identified by the FFL reasoner is recorded. Each event scenario simulated produces one record. These records will be used by the clustering approaches proposed in this chapter.

8.2.1 Latent Class Analysis

The social sciences have used the concept of latent classes since the 1950s [184]. Manifest (or observed) variables are the data of empirical studies. A latent variable is one not directly tested but is nevertheless correlated to observations of the manifest variables. Structural equation modeling is a set of methods for identifying these latent variables in the underlying structure of data [185]. If the latent variable is continuous then methods such as factor analysis and multivariate mixture estimation can be used to find this structure. However, if the latent variables have discrete categories then the structure fits a latent class model [186]. As an example, survey questions on personal views of several political topics can form the parameters of a statistical model. Latent class analysis (LCA) on the survey data could be used to identify subgroups into which the respondents are classified. Groups identified within the data would likely correspond to labels like "conservative","liberal", etc. There are three main results from performing an LCA. First, each data point has a probabilistic membership to each class of the latent variable (e.g., the respondent's likely political leaning). Secondly, each discrete variable state is correlated to a latent class (e.g., liberals have a high probability of answering affirmatively to question three.) The final component of the LCA output is class membership percentages for the entire data set (e.g., 40% conservative, etc.)

Formally, the latent class model is based on the concept that the probability of observing a specific pattern ($\mathbf{Y}$) of manifest variable states $\mathbf{y}$, denoted $P(\mathbf{Y} = \mathbf{y})$, is a

weighted average of the C class-specific probabilities $P(\mathbf{Y} = \mathbf{y}|X = x)$, where X is a latent variable with C number of classes. Weighting with the proportion of that class to the latent variable $P(X = x)$ results in Equation 8.1.

$$P(\mathbf{Y} = \mathbf{y}) = \sum_{x=1}^{C} P(X = x)P(\mathbf{Y} = \mathbf{y}|X = x) \tag{8.1}$$

Further, the manifest variables within a class, Y_l are assumed to be locally independent. Therefore, Equation 8.2 defines the probability of observing a pattern in the L manifest variables within a class.

$$P(\mathbf{Y} = \mathbf{y}|X = x) = \prod_{l=1}^{L} P(Y_l = y_l|X = x) \tag{8.2}$$

As with K-Means (or C-Means) data clustering, algorithms for implementing LCA use expectation maximization for a predefined number of groups. Therefore, LCA must be executed iteratively in order to identify the correct number of classes for the latent variables. Identifying the goodness of fit of the latent class model is typically accomplished by examining the Akaike Information Criterion (AIC). This metric is an estimate of the information entropy (information lost) when a statistical model is used to describe reality. The AIC formulation modifies the log-likelihood estimation by the number of parameters, punishing over-fitting models. The objective in checking goodness of fit with AIC is to find the minimum of Equation 8.3, where $\mathbf{K}$ is the number of parameters and $\mathbf{L}$ the likelihood function for the statistical model.

$$AIC = 2\mathbf{K} - ln(\mathbf{L}) \tag{8.3}$$

LCA was chosen as a clustering method over other clustering methods because the manifest variables in the results are the discrete states of each function in the system. Additionally, the hypothesis of this work is that the failure behavior of a system is also categorical. This categorical system-level failure is the latent variable in our analysis. The discrete (and ordinal) nature of the variables rules out other multivariate mixture models.

8.3 Overview and Comparison of Methods

While the analysis of individual failure scenarios of interest was shown to be useful in design alternative selection [72] and scenario exploration [180], the following three methods focus on evaluating overall system failure characteristics for a large set of scenarios. The first method uses an evaluation of the consequence of a fault scenario and identifies other scenarios with similar consequence. The second method use Latent Class Analysis to identify the high-level failure behavior of the system. Finally, the third method identifies functionally identical and similar fault scenarios using a similarity relation metric. The following sections discuss each method in detail and Table 8.1 provides a comparison and summary of results for each method.

Table 8.1: Comparison of the Three Methods for Fault Simulation Clustering

	Consequence Method	**LCA Method**	**Similarity Method**
How to find the groups:	Cost metric is used to evaluate the consequence of a failure	LC regression is applied to the data	Similarity metric and the distance between fault scenarios is calculated
What the groupings mean:	Levels of impact	Discrete, emergent patterns of failure behavior	Faults that effect the system in identical and similar ways
How to use the groupings:	Prioritize faults to address	Characterize system failure behavior	Identify similar faults to states of interest

8.4 Example Application

To demonstrate the clustering approaches applied to function failure analysis results, we perform an FFIP analysis on a design concept of an electrical power system (EPS). This EPS example is an early design-stage model that uses a battery to provide power for a set of AC and DC loads. This example is based on the design of the Advanced Diagnostic and Prognostic testbed located at the NASA Ames Research Center [164]. In

previous work various potential design architectures were compared using a quantified interpretation of the FFIP results [72]. This work focused on evaluating the benefit of different potential redundancies through comparing the reduction in cost of failure events, where the cost is the sum for all the function's values (as identified by the designer) multiplied by the cost of that function being in the identified state. While this example system is complicated enough to identify the effects of fault propagation, the FFIP analysis has been demonstrated on a more complicated system (nuclear power generation [180, 181]). Figures 8.1 and 8.2 illustrates a functional and component view of the system.

The effect of different component fault modes is identified for the EPS using a simulation of the system built by connecting component models created with the Stateflow toolbox in Matlab Simulink. A scenario is simulated where one or more faults is triggered and the resulting changes in system dynamics are allowed to propagate. The output of each simulation is a health status of each function identified in Figure 8.1. Using a Matlab script, a large set of scenario results is generated; first simulating each component fault mode as a single fault scenario and then two fault combinations. Three or more fault scenarios can also be generated in the same manner. For this system, simulating every possible combination of two faults is not computationally expensive. However, for more complex systems there are three possible ways for guiding the scenario selection and simulation process. First, expert knowledge can provide direction on the components that are likely to negatively interact and have known fault causation. An alternative to this approach is simulating fault modes based on the relationship between causes and symptoms of faults as shown in [71]. Finally, the clusters generated using the approach demonstrated in this chapter may provide direction in fault modes that should be simulated together in an iterative approach.

Figure 8.3 illustrates a snippet of the function failure analysis results for the scenarios. In this figure each row is a separate scenario and the columns correspond to the resulting identified state of the system functions. For the clustering analysis three sets of scenarios were generated. The first set of results tested each failure mode of each component resulting in 97 simulations. The second and third set of scenarios tested two fault scenarios. The difference between these last two sets was a reversal of the order in which the faults where tested (e.g., battery fault then relay fault, and reversed order in the third set). For the three sets this generated 677 fault simulation records.

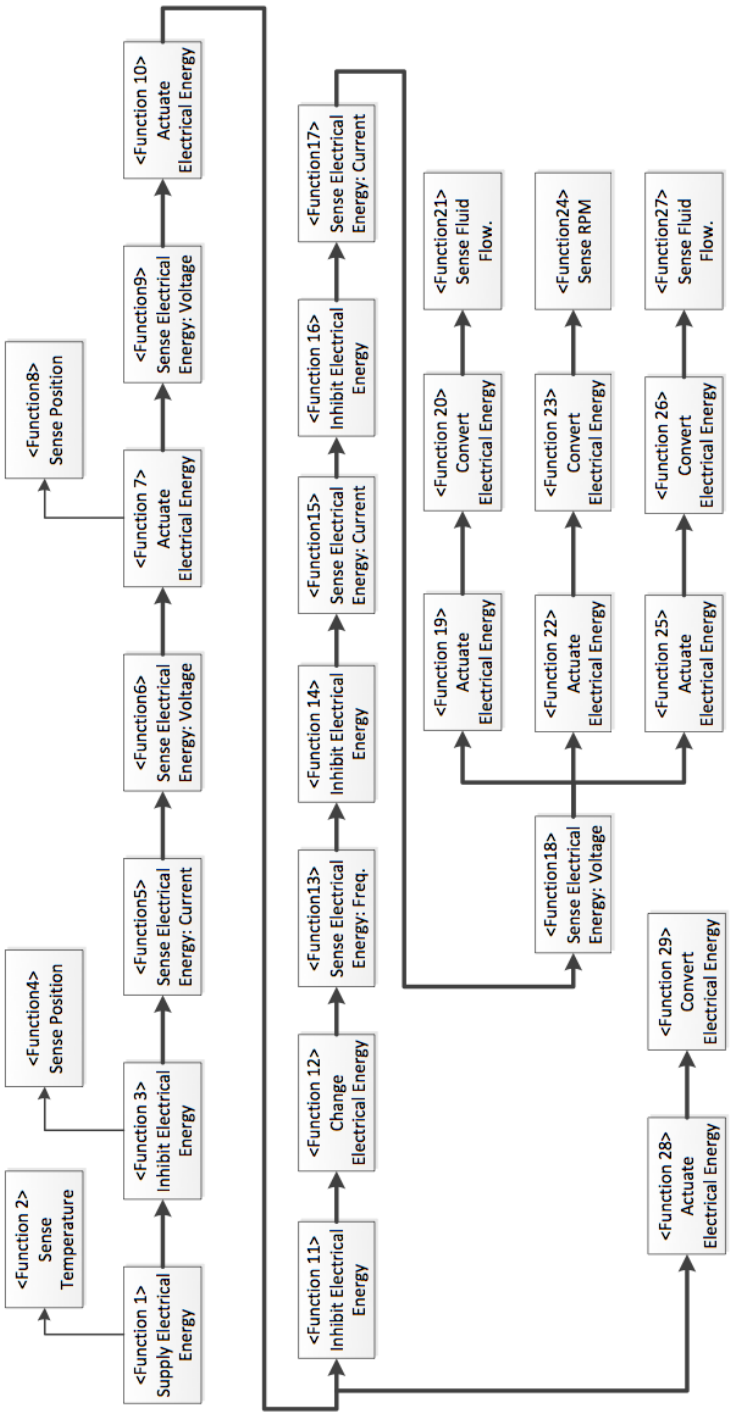


Figure 8.1: Functional model of the electrical power system (EPS).

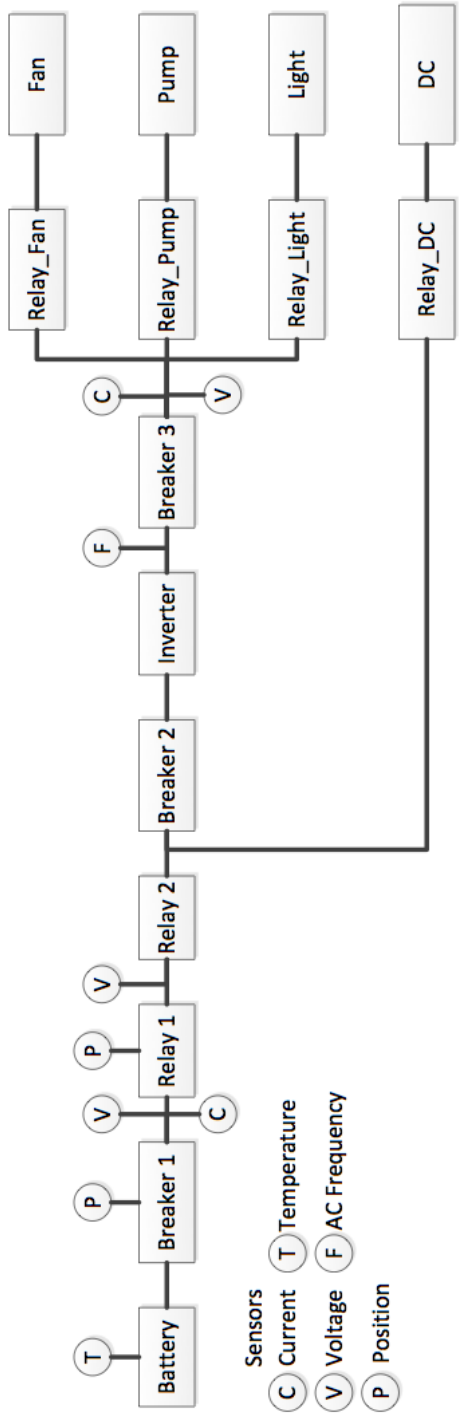


Figure 8.2: Component model of the Electrical Power System (EPS).

fault id	Function 1	Function 2	Function 3	Function 4	Function 5	Function 6	Function 7	Function 8	Function 9	Function 10	Function 11	Function 12	Function 13	Function 14	Function 15	Function 16	Function 17	Function 18	Function 19	Function 20	Function 21	Function 22	Function 23	Function 24	Function 25	Function 26	Function 27	Function 28	Function 29
1	3	1	4	1	1	1	4	1	4	1	4	3	1	4	1	4	1	1	4	4	1	4	4	1	4	4	1	4	4
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	4	1	4	1	1	1	4	1	4	1	4	3	1	4	1	4	1	1	4	4	1	4	4	1	4	4	1	4	4
4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	4	1	4	1	1	1	3	1	4	1	4	3	1	4	1	4	1	1	4	4	1	4	4	1	4	4	1	4	4
6	4	1	4	1	1	1	4	1	3	1	4	3	1	4	1	4	1	1	4	4	1	4	4	1	4	4	1	4	4

1 =Operating
2=Degraded
3=Lost
4=No Flow

Figure 8.3: An example of the functional state output from simulating several failure scenarios. Each row represents the final functional impact of the component-level functions.

8.5 Consequence and Risk Metric

In previous research the functional "cost" of a failure event was determined by multiplying the cost of the state by that function's importance to the designer [72]. This concept of cost or consequence was applied to the failure scenario results. Through previous analysis of the example system certain failure behavior has been observed. For example, this system is not currently modeled with a controller. Therefore, failure events with only sensor failures have very little functional impact on the system. While the function of that sensor may be lost or degraded, no other functionality in the system is lost. Further, some failures result in power loss or open circuits, causing many of the functions to be at a state of "No Flow." The consequence of the system being in a specific functional state is dependent on the value of a particular function. Therefore, failure scenario consequence is considered requirement specific. That is, if a requirement for the EPS is to maintain operation of the loads, then the value of those load functions and the consequence of failures that effect them are high. For this analysis, all losses were considered equally undesirable. Therefore each function has identical worth and the functional consequence is reduced to the sum of the functional states. Since "Lost" was accounted as *3* and "No Flow" as *4*, this implies that failures that result is a loss of power to components so that they cannot operate are worse than component failures. This perspective is based on the relative ease of replacing faulty components for this

system and the value in robust designs that allow continuous operation.

The results of the 677 scenarios is provided in Figure 8.4. A clear pattern can be seen in this data indicating at least three typical costs of failure for this system. The sum of a scenario could theoretically be anything within the range of 29 (every function "Healthy") to 116 (every function "No Flow"). However, it is clear that the emergent behavior of the combined component behaviors leads to a typical range of cost values. These three ranges observed are:

1. 29 through 45
2. 46 through 60
3. 61 and higher

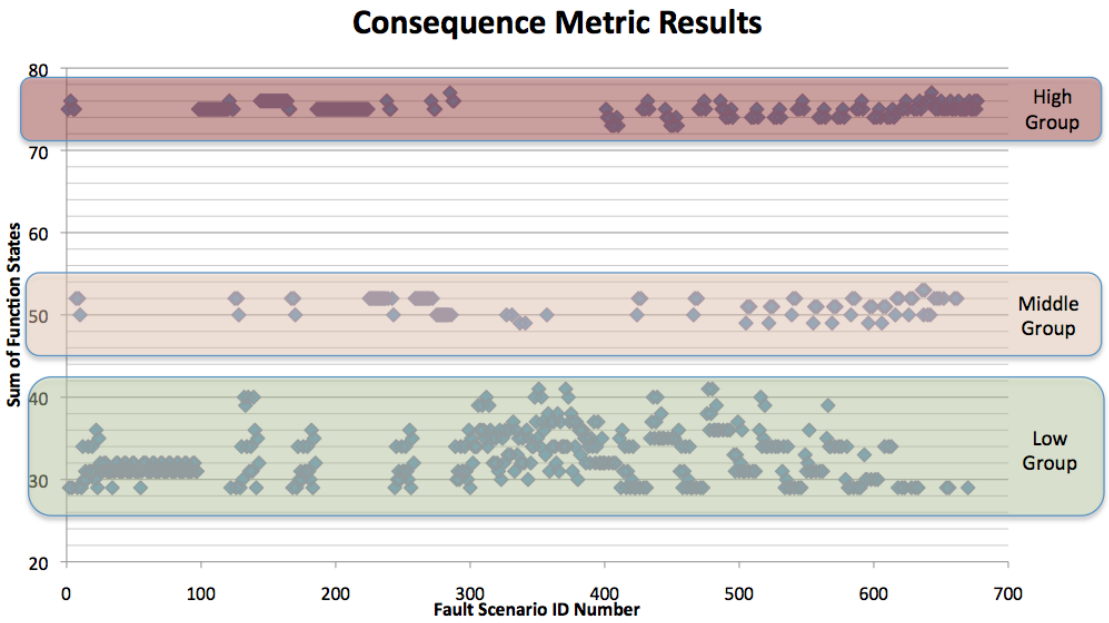


Figure 8.4: Representing the cost of each of the 677 failures simulated results in three clear groupings for this system.

The top level groups found through this consequence method are those with the highest negative effect on system functionality. These results could be multiplied by

either a qualitative or estimated probability to identify high risk failures. Designs that reduce the impact of these specific faults and reduce fault impact overall are preferred.

8.6 Latent Class Analysis

The second method of evaluating the failure results is focused on identifying patterns of failure behavior. For this method a Latent Class Analysis (LCA) was performed on the 677 fault simulation results using the package poLCA [187, 188] for the statistical software tool R [189]. The poLCA package treats the manifest variables as nominal. Although the function's state is ordinal (that is, state 2 "degraded" is more like state 1 "healthy" than either state 3 or 4) the same multinomial distribution is assumed for the expectation maximization algorithm [190]. In order to avoid local maxima, the poLCA classification algorithm is executed 10 times for each specified number of classes. The 10 iterations are performed assuming a predefined number of classes of the latent variable (system failure type). The correct number of classes is identified as the LC model with the lowest Akaike Information Criterion (AIC). Figure 8.5 shows the change in AIC as the number of classes ranges from two through seven. In this analysis, the AIC flattens out after four classes. AIC is a minimum at five classes after which the AIC begins to increase due to over-fitting in the model.

Once the correct latent class model is identified, there are three desired outputs from the LCA. The first output is a set of conditional probability tables for each manifest variable. These tables identify the probability of finding a manifest variable at a specific state for each category of the latent variable. In the context of this analysis, this indicates that if a failure event is of a particular class of system failure then the function is likely to be in a specific state (healthy, degraded, etc.) The second output uses these probability tables to identify the posterior probability of a scenario belonging to each class of the latent variable. This is the output used for the probabilistic classification of the failure events. Finally, the proportion of each classification is reported. This leads to the identification of the class with the largest membership of failure events. A Matlab script was also created to sort and label the failure scenarios according to each class. A subset of the first seven failure scenario results for each class are provided in Figure 8.6. For comparison, each failure event was also categorized according to the consequence groupings found using the previous method and those results are under the

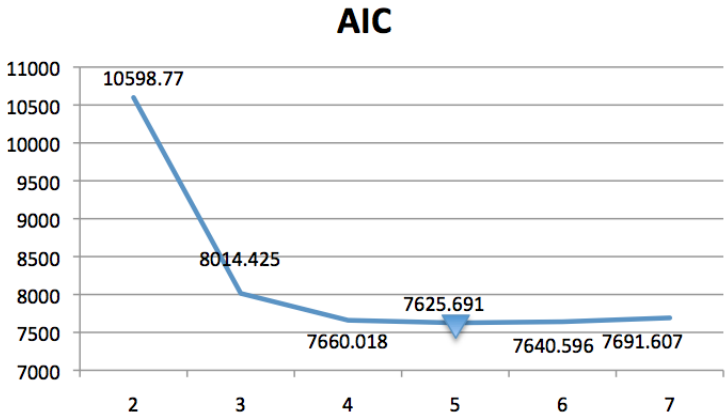


Figure 8.5: The Akaike Information Criterion (AIC) as a function of the number of classes for the latent class model applied to the function failure simulation data. Minimum is at 5 classes of the latent variable of "System Failure."

columns called "S-Group" in Figure 8.6. What can be seen from Figure 8.6 is that that S-groupings and the clustering resulting from the LCA method do not identify the same groupings. This can be explained in that the LCA method is identifying patterns of failure.

The meaning of the different classes found using an LCA are not directly found but must be inferred from the resulting groups. For example, all of the single fault sensor failures where grouped together in class 5. These faults all have minimal functional effect on the system. The faults in class 2 also have only a few negative functional effects, however these faults led to failures in 2 of the load branches. Many of the faults in class 4 led to functional losses in all of the AC loads and class 3 had failures that affected the main power supply and resulted in a loss of power for the entire system. In this way the classes can take on meaning for the designer based on the failures that are grouped together. While the simple summing approach provides a sense of the consequence of a fault, the LCA clusters highlight the higher-level patterns of failure behavior.

LCA Class 1

P(1)	Fault ID	Components	Modes	S-Group
1	677	DC_Relay_Mode_&_DC_Mode	1 1	1
1	676	Breaker4_Mode_&_DC_Mode	2 1	1
1	675	Breaker4_Mode_&_DC_Relay_Mode	2 1	1
1	674	Breaker4_Mode_&_DC_Mode	1 1	1
1	673	Breaker4_Mode_&_DC_Relay_Mode	1 1	1
1	672	Light_Mode_&_DC_Mode	2 1	1
1	671	Light_Mode_&_DC_Relay_Mode	2 1	1
1	669	Light_Mode_&_Breaker4_Mode	2 1	1
1	668	Light_Mode_&_DC_Mode	1 1	1
1	667	Light_Mode_&_DC_Relay_Mode	1 1	1
1	666	Light_Mode_&_Breaker4_Mode	1 2	1

LCA Class 2

P(2)	Fault ID	Components	Modes	S-Group
1	662	Light_Relay_Mode_&_Breaker4_Mode	1 2	1
1	660	Light_Relay_Mode_&_Light_Mode	1 2	1
1	652	Pump_Mode_&_Light_Relay_Mode	2 1	1
1	649	Pump_Mode_&_Breaker4_Mode	1 2	1
1	647	Pump_Mode_&_Light_Mode	1 2	2
1	645	Pump_Mode_&_Light_Relay_Mode	1 1	1
1	642	Pump_Relay_Mode_&_Breaker4_Mode	1 2	1
1	640	Pump_Relay_Mode_&_Light_Mode	1 2	1
1	638	Pump_Relay_Mode_&_Light_Relay_Mod	1 1	1
1	637	Pump_Relay_Mode_&_Pump_Mode	1 2	1
1	636	Pump_Relay_Mode_&_Pump_Mode	1 1	1

LCA Class 3

P(3)	Fault ID	Components	Modes	S-Group
1	493	Relay1_Mode_&_Breaker4_Mode	1 2	3
1	491	Relay1_Mode_&_Light_Mode	1 2	3
1	488	Relay1_Mode_&_Pump_Mode	1 2	3
1	485	Relay1_Mode_&_Fan_Mode	1 3	3
1	484	Relay1_Mode_&_Fan_Mode	1 2	3
1	483	Relay1_Mode_&_Fan_Mode	1 1	3
1	482	Relay1_Mode_&_Fan_Relay_Mode	1 1	3
1	481	Relay1_Mode_&_Breaker3_Mode	1 2	3
1	480	Relay1_Mode_&_Breaker3_Mode	1 1	3
1	479	Relay1_Mode_&_Inverter_Mode	1 1	3
1	478	Relay1_Mode_&_Breaker2_Mode	1 2	3

LCA Class 4

P(4)	Fault ID	Components	Modes	S-Group
1	613	Fan_Mode_&_Breaker4_Mode	1 2	1
1	611	Fan_Mode_&_Light_Mode	1 2	1
1	608	Fan_Mode_&_Pump_Mode	1 2	1
1	593	Fan_Relay_Mode_&_Fan_Mode	1 1	1
1	580	Breaker3_Mode_&_Fan_Mode	2 1	1
1	576	Breaker3_Mode_&_Breaker4_Mode	1 2	2
1	574	Breaker3_Mode_&_Light_Mode	1 2	2
1	571	Breaker3_Mode_&_Pump_Mode	1 2	2
1	568	Breaker3_Mode_&_Fan_Mode	1 3	2
1	567	Breaker3_Mode_&_Fan_Mode	1 2	2
1	566	Breaker3_Mode_&_Fan_Mode	1 1	2

LCA Class 5

P(5)	Fault ID	Components	Modes	S-Group
1	97	Voltage3_Mode	6	1
1	96	Voltage3_Mode	5	1
1	95	Voltage3_Mode	4	1
1	94	Voltage3_Mode	3	1
1	93	Voltage3_Mode	2	1
1	92	Voltage3_Mode	1	1
1	91	Current3_Mode	6	1
1	90	Current3_Mode	5	1
1	89	Current3_Mode	4	1
1	88	Current3_Mode	3	1
1	87	Current3_Mode	2	1

Labels

P(1): Probability of belonging to Class 1
Fault ID: Numeric ID tag for fault records
Components: Component(s) where a fault is triggered (in order)
Modes: The numeric label for the fault mode(s) of each component
S-Group: Grouping based on summing the failure modes numbers

Figure 8.6: A selection of some of the failure simulations classified in each of the five classes of the latent variable "System Failure."

8.7 Functional Similarity Metric

The third method of grouping uses a metric of functional similarity to compare all of the scenarios to a specific functional state of interest. The objective of this method is to guide designers when they know of a particular fault or undesired system functional state to identify other scenarios with exact and similar effects. To determine the similarity between two scenarios, a distance is determined between the states of each function. This is analogous to the hamming distance except that the ordinal nature of the function states is taken into account. Classical clustering algorithms such as k-means also use a similar concept of distance to group objects. The differences between this approach and hierarchal clustering are: 1, the distance scale is adjusted to recognize the functional meaning of the states and weighted by the importance of the function; 2, clustering algorithms aim to find the clusters that all data can be fit into, whereas this approach bases the cluster on the distance from a point. In this approach, ordinal classifications are used to determine the distance as specified in the relation matrix in Table 8.2. In this relational matrix we specify that there is no functional difference in a function being "Lost" and that function being inoperative in a state of "No Flow."

Table 8.2: Relational Matric for identifying the distance between function states.

State	Healthy	Degraded	Lost	No Flow
Healthy	0	1	2	2
Degraded	1	0	1	1
Lost	2	1	0	0
No Flow	2	1	0	0

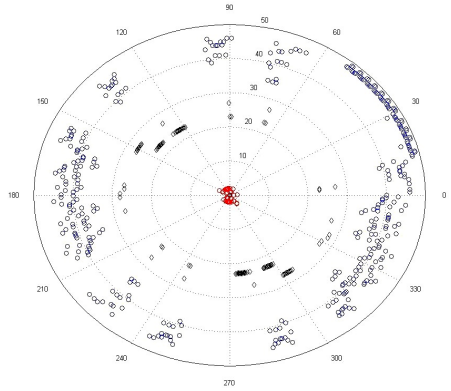
A functional similarity algorithm was generated for analysis of these results using the relational matrix in Table 8.2. The total distance was calculated by summing over each function state evaluation. Here again, a weighting for functional importance could be incorporated. However, for this analysis each function was given equal importance. System states that are functionally identical are collected in an 'Exact List'. All other states are mapped using polar coordinates where the point's radial position is the calculated distance and the angular position is indexed to the fault identification number. Figure 8.7 illustrates three maps comparing all of the tested faults to scenarios with specific component faults. These maps help identify distinct groups of functional similarity.

Based on the nearest cluster, the 'Similar List' is generated from those failure scenarios within the central grouping. Figure 8.8 illustrates a selection of the Similar and Exact list for the scenario where the fault of "Inverter Failed Off" was simulated. For any functional similarity map generated there can be different groupings (based on radial distance), for example there are 2 groups in Figure 8.7b and 3 in both Figures 8.7a and 8.7c.

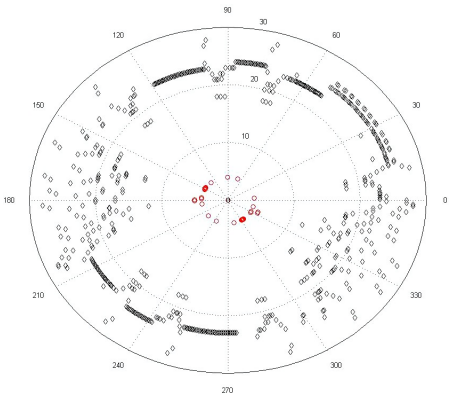
Generally it is the close proximity faults that are of interest to the designer. This group, along with the identical matches, can assist designers in identify the risk of certain faulty states. When using Fault Trees, expert knowledge is needed to identifying all the possible low-levels faults and combinations of faults that might lead to a top-level event. In contrast, the functional similarity method identifies all of the scenarios that have the same impact as well as all the scenarios that are similar enough that designers should be aware of them. Thus this method is most useful when designers are aware of a particular faulty system state and want to explore the design to identify multiple ways of reaching that or a similar state. For example, Figure 8.8 provides some of the similar and exact matching scenarios as the inverter failure. In this case the designer knows that one hazard leading to the undesired state of all the AC loads being lost is caused by the inverter failure. What is identified through this analysis is that (along with several other faults) a fault in the breaker next to the inverter will also lead to this or fault in the breaker before the inverter is similar in effect to be of concern. In this way, functional similarity groupings help designers explore failure behavior and evaluate multiple causes for losses.

8.8 Summary

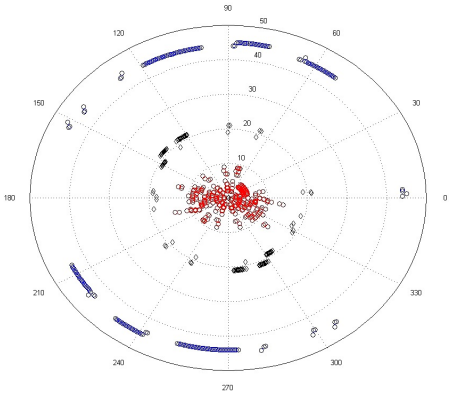
This chapter proposed three different approaches for evaluating the results of a function-based failure analysis method in the early design stage. These different approaches each offer different perspectives on a system design's potential failure behavior and can be used to provide insight to designers for design decision making. In contrast to other methods which focus on single faults or single failure scenarios, the goal of this work is to characterize a design's overall failure behavior. Specifically, three methods were proposed in this chapter for analyzing and grouping failure analysis results to provide this design insight. Table 8.1 compared the three approaches and the findings when



(a) Similarity map for fault "Battery Disconnected."



(b) Similarity map for fault "Inverter Failed Off."



(c) Similarity map for fault "Pump Seized."

Figure 8.7: Similarity maps created by comparing a state to all other listed states for functional similarity.

Functionally Similar to Invert Failed Off			
Component 1	Mode	Component 2	Mode
'Breaker3_Mode'	1		
'Battery1_Mode	[2	Breaker3_Mode'	1]
'Breaker1_Mode	[2	Breaker3_Mode'	1]
'Breaker2_Mode	[2	Breaker3_Mode'	1]
'Breaker3_Mode	[1	Fan_Relay_Mode'	1]

Functionally Identical to Inverter Failed Off			
Component 2	Mode	Component 2	Mode
'Breaker2_Mode'	1		
'Inverter_Mode'	1		
'Battery1_Mode	[2	Breaker2_Mode'	1]
'Battery1_Mode	[2	Inverter_Mode'	1]
'Breaker1_Mode	[2	Breaker2_Mode'	1]

Figure 8.8: Part of the similarity and exact lists of scenarios with respect to the inverter fault.

applied to an electrical power system design.

Grouping based on consequence provides designers with clusters of high to low consequence. This evaluation of consequence is based on the designer's value of functions and is requirement specific. This leads to the identification of critical scenarios based on their effect. Although not demonstrated here, generic component failure rates have been collected [191] and this information can be used to identify a failure probability over a certain operating time. With this information designers can identify critical failures based on their risk (consequence x probability). This approach would require using either statistical independence assumptions or a priori knowledge of the fault correlation for multiple fault scenarios. However, this approach is still advantageous over traditional risk analysis methods such as PRA because the fault propagation is identified through simulation instead of a specified path.

The second approach to grouping proposed in this chapter used Latent Class Analysis (LCA) to reason about the high-level failure behavior of the system. The LCA method identifies patterns in the failure analysis results and correlates those patterns to an underlaying latent variable of "system failure." The result of using this method is a characterization of the system-level failure behavior into discrete categories. Additionally, this method generates probability tables for observing a function state within a category of system failure. While the categories are complex and can be difficult to apply labels to (as is typically done when conducting an LCA), the overall system can be characterized as having distinct failure behavior. For large complex systems where complete multiple fault simulation across multiple subsystems would be computationally prohibitive, this method allows for a probabilistic approximation of a subsystem. That

is, the subsystem could be replaced with a model containing the failure modes (classes) identified with LCA and the output would be a probabilistic evaluation of the state of each function in that subsystem based on the subsystem failure category. Further, since the LCA method produces probabilistic tables of function state for a class, the classification of new failure scenarios can be predicted without rerunning the LC regression algorithm.

Finally, the functional similarity grouping method proposed in this chapter compares the set of function failure scenario analysis results to a particular system functional state of interest. A relational matrix is used to define the distance between the ordinal function states of "Healthy", "Degraded", "Lost", and "No Flow". Based on the distance for each function a total distance is determined between each failure scenario and the state of interest and this distance can further be represented on a scalar plot. This approach allows for the identification of scenarios that result in similar and identical functional effects. This method can be used when the designer knows of an undesired system state and wants to assure that design changes address all the potential scenarios for reach that or a similar state. The three methods of grouping the scenario data presented in this chapter address different design concerns as mentioned above. Alternative methods of grouping the results may be necessary to address additional concerns as they are identified.

The focus of this chapter has been on different methods of exploring design-stage function failure analysis results. The objective of this has been to aid designers in making risk-informed design decisions. The vision of this work is that design information gained by identifying what failure scenarios to address, the overall failure behavior of the system, and similar or identical failures effects will guide designers to make better design decisions.

Chapter 9: Special Issues: Large Complex Systems and the Analysis Sandbox

The content of this chapter has been accepted to the Journal of Computing and Information Science in Engineering and is based on published conference paper co-written by David Jensen, Nikolaos Papakonstantinou, Seppo Sierla, and Irem Y. Tumer [180].

9.1 Introduction

9.1.1 Challenges for Complex System Design

The design of large complex systems faces numerous challenges due to the complexity of the design information content, unknown interactions between subsystems, and the need to integrate expertise of designers or teams of designers from different domains. Specifically, we recognize three challenging tasks that arise in the design stage for these systems: 1) Codesign of the multiple domains of technology; 2) Determining the effects of emergent behavior; and, 3) Determining risks across the system from fault propagation. The method presented in this chapter is aimed at addressing the challenges in these tasks.

The term codesign is most often used to refer to technologies that require close integration of electrical hardware and software systems as found in mechatronics and consumer electronics [192, 193]. However, the technical challenges and the general solution approach are also applicable to the integration of electromechanical and human control systems. Specifically, it is challenging to represent necessary system design information across technical domains at a similar and relatable abstraction level. This representation is necessary for the development of interfaces and interaction behavior. While it is advantageous to develop different subsystems concurrently, often in the design stage different subsystems are at various levels of design refinement. Formal model representation languages, such a s SysML [105], serve to implement a model-based design approach to address this challenge. That is, high-level system models may be composed of multiple

representations to capture design information across domains even when subsystems are at different levels of design refinement.

Emergent behavior is a unique aspect of systems where they exhibit behavior that is more complex than the sum of the behavior of their constituent parts. Along with the emergent behavior, which is intended and is used to implement the desired functionality of the system, unforeseen behavior may also occur. In this work, the focus in on emergent behavior that may compromise the reliability or safety of the system. Emergent behavior is defined as degradation or loss of functionality of a subsystem that is caused by poorly chosen design parameters in other subsystems. Our goal is to provide methods and tools for studying how changing several such parameters impacts the occurrence of emergent behavior.

Finally, the third challenging task is concerned with evaluating systems designs for risk and safety. Traditional approaches focus on identifying faults and their probabilities of occurrence and consequence. For large systems this can create certain statistical limitations for risk analysis. For example, in complex systems faults are rarely independent. Further, the consequence of a component fault depends on the connection that the component has with the rest of the system. This is evidenced by the increase in failures that affect systems that deploy both computational and physical elements. Therefore a risk assessment of complex systems should be based on a failure analysis that accounts for fault propagation across domains and subsystems.

9.1.2 Prior Work

A function failure analysis approach was developed and presented in prior work to identify the functional losses and determine their effects through downstream propagation. This method was developed to help address the three challenging designs tasks presented in the previous section. Specifically, the Function Failure Identification and Propagation (FFIP) framework was developed to capture the effects of complex system interactions early in the design stage and presenting the effects and propagation of faults in terms of functional losses [62, 72, 182, 71]. The FFIP design-stage analysis framework was developed to identify the system-wide functional effect of component failures, even when the fault propagation paths cross the boundaries of electronic, mechanical and software subsystems [182, 180].

To facilitate the codesign challenge, the FFIP method uses a functional model to represent multiple domain subsystems at the same abstraction level. Further, these functions are linked to generalized components configuration representation. The FFIP method is a model-based simulation approach and behavior associated to these components can be at differing levels of design refinement. The state-based simulation of nominal and faulty behavior also assists in identifying emergent behavior early in the design stage. Finally, the FFIP analysis method consists of triggering a fault and recording the fault propagation and system impact. These results can then be used to evaluate the risk of potential system designs without independence assumptions and across the different technical domains of the system.

Thus, FFIP is applied at the early concept design phase using an abstract system representation to eliminate unreliable designs before costly design commitments are made. High fidelity system models are not available at this phase, therefore the results of an FFIP analysis are strongly affected by specific modeling choices in the component model and the sequence and timing of critical event scenarios.

9.1.3 Objectives and Contributions

In this chapter, the FFIP framework and supporting tools are extended significantly to better address the challenges of emerging behavior. While high fidelity simulation at an early phase is not possible, the results should not be totally dependent on specific model parameter values. The simulation approach presented in this chapter permits varying the values of key design parameters and the timing of critical events; simulation results reveal the impact of the variations on the emergent behavior. Previous research has expressed values for energy, material and signal (EMS) flows with an enumeration such as [zero, low, nominal, high], so that component behavioral models determine output flow values according to input flow values and the current nominal or faulty state of the component. This approach is not sufficiently fine-grained for capturing how several parameter changes might either reinforce or weaken each other in causing emergent behavior, so the framework needs to be modified to support continuous flow values in order to describe feedback loops. This extension of the FFIP approach to system representation addresses the first challenge identified earlier by allowing for more detailed subsystem behavior, while maintaining the equal abstraction system level representation

necessary for codesign.

In order to systematically investigate the effects of different parameter values and different timing of critical events, it will be necessary to run the FFIP simulation for each combination of parameter values, so the number of simulation runs grows exponentially as more parameters are brought into the scope of the study. In order for this to be feasible, a user interface is needed for specifying the parameters and the ranges in which they are varied. The second goal of this chapter is then to propose a generic and scalable algorithm for automatically running the complete set of simulation runs implied by the user's choices. The algorithm and user interface are implemented and interfaced to the Matlab/Simulink based FFIP framework, producing Excel output for each run, which can be further sorted or filtered according to the health status of a function of interest specified by the user. The paper demonstrates the use of these tools for identifying the most relevant parameters for hazards and for focusing their values to the interesting range in subsequent series of automatic simulation runs. With these changes, the failure analysis can provide better results for risk assessment early in the design stage.

9.2 Background

The method presented in this chapter uses an early-design stage system representation and behavioral simulation to evaluate risk in functional terms. A variety of methods and techniques have been developed for risk assessment, system information sharing, and simulating expected system performance, discussed next.

9.2.1 Risk Assessment in Complex System Design

This research fundamentally aims to reduce or eliminate the risks of malfunctions and failures. Many methods exist in practice and in research literature to enable risk mitigation. Failure Modes and Effects Analysis (FMEA) is used in safety critical industries for safety analysis. It is based on past experience and defines the effects of a single failure at a time for a number of component failure modes. These effects can be identified by using the Fault Tree Analysis (FTA) method [76]. Some research has used FMEA for multiple failures but if detailed design documents are used the combinations of failures grow beyond a manageable size in complex systems [194]. Functional models can be used with

FMEA to describe more clearly the effects upon the functions of complex systems [195]. The Functional Failure Analysis (FFA) is a method similar to FMEA but at a functional level of the system design [196]. Although attempts at partially automating this process have been made, FMEA still remains a manual and laborious process [197]. HAZard and OPerability (HAZOP) is a qualitative method that can be performed on the functional model of a complex system. System functions are combined with a set of keywords with the aim to identify all possible failure modes [198]. This method also requires significant manual effort, although there are attempts to automate parts of it by using knowledge databases [199]. Probabilistic Risk Assessment (PRA) [79] is a quantitative method that can identify weaknesses and vulnerabilities in complex systems. It requires a detailed design and the failure rate information of the components used. Therefore, it can only be applied in the latter phase of the system design process [200].

In general, a high level of system behavioral and interaction knowledge is required to perform these assessment methods. Therefore these methods are most advantageous for verifying the safety of a well-refined design. The goal of this work is to utilize risk assessment early in the design-stage to help designers make design decisions, before a refined design has to be completed.

9.2.2 Complex System Design Representation

The primary method for conveying the relationship between elements in the design-stage of complex systems is through abstract engineering design models of functionality and interface requirements. These models range from detailed function and behavior information to loose hierarchical artifact relationships. These approaches aim at capturing the complexity of the system through a single (or set) of static representations. Examples include: block diagrams based on function logic [201], a historical design repository approach proposed as a framework to store component information and relate the elements of information to each other [202], and other function-based representations that support search and modeling processes of conceptual design [203]. These approaches utilize the concept of function to connect multiple domain system elements at the same abstraction level. Other research has explored languages for describing product functionality and relating it to product behavior [29] and using these languages to reason about designs [30]. Systems Modeling Language (SysML) is a formalism for representing many differ-

ent aspects of system design including function, behavior and structure. Several recent works have focused on extending the static system representation of SysML for dynamic system simulation in other tools [204]. Finally, several researchers have introduced representation as part of methods of tackling design time failure analysis. Examples include: a function-failure design method [64] which uses functional models and historical failure data to map the functionality of a system, a Bayesian network analysis tool for evaluating properties of function structures using dependencies between flows and functions [60], and a risk based decision-making and cost-benefit analysis method [159] to assess the risk of integrating health management capabilities in aerospace systems at the system-level design stage.

This research builds on the functional modeling approach to design representation. These functional representations are also mapped to generic component and behavioral representations. The presented approach leverages the above modular approach to design, where system functions, structure and behavior are created from combining lower-level models.

9.2.3 Simulation-Based Design

This research utilizes behavioral simulation of models at multiple abstraction levels with increasingly higher fidelity, including models that do not require component geometry details to be implemented at the architecture level. Simulation-based design methods require the capability of specifying detailed input design parameters and using them to obtain a model response. This simulation process is called a forward model, which system designers use to account for the effects of variability in the input and design parameters on the model response, thereby incorporating uncertainty into the design process. Common techniques for forward model analysis include sampling techniques to propagate uncertainty (e.g., Monte Carlo simulation), and model approximation methods such as response surface models and meta-modeling [205, 206]. These methods support reliability-based and robust design optimization techniques. Reliability-based methods [207] estimate the probability of system response based upon specified probability distributions but no effort is made to minimize variation, whereas robust design [208] strives to minimize the effects of variation.

Simulation based techniques have also been used to assess risk factors in a design

and analyzing the effects of faults in a system. For example, directed graphs and Multi-Signal Flow Graphs [140] are used in many domains to analyze dependencies and fault propagation and model cause-effect dependencies.

At the early design stage, the detailed specifications of component geometry and system topology required by some methods are usually not available. To address this problem, we expand our prior work to address simulation and reasoning in early design [182, 72, 209, 130].

9.3 Methodology

The Function Failure Identification and Propagation (FFIP) framework was developed to capture the effect of complex system interactions early in the design stage and presenting the effect and propagation of faults in terms of functional losses [62, 72, 182, 71]. The simulation and reasoning approach in FFIP has its roots in qualitative physics [141] and qualitative reasoning [130, 143, 142]. FFIP utilizes a finite state representation of system behavior, and performs reasoning based on qualitative relationships between functional and behavioral models of system components. Prior to this research, FFIP has been demonstrated on relatively simple example systems to demonstrate different aspects of the methodology. This chapter presents how the FFIP simulation and analysis can be improved to better meet some of the challenges found in the design of large complex systems [182, 180].

This section introduces the FFIP methodology by a simple example, so that the reader may understand how the results in the later sections were obtained. The example is a simplified subprocess from the pulp and paper industry (see Figure 9.1). White liquor is heated and supplied to a digester in order to enable a chemical reaction necessary to pulp production. The white liquor tank has a continuous flow of incoming and outgoing liquor, and the tank serves the purpose of intermediate storage and heating of the liquor. The desired functionality is expressed as a functional model, according to the functional basis defined in [133, 119] (see Figure 9.2). Failure propagation analyses with FFIP use the functional model to identify loss of functions that are not necessarily associated with failed components.

The functional model is not simulated, since simulation requires information about components, their connections and internal behavior. This information is captured in

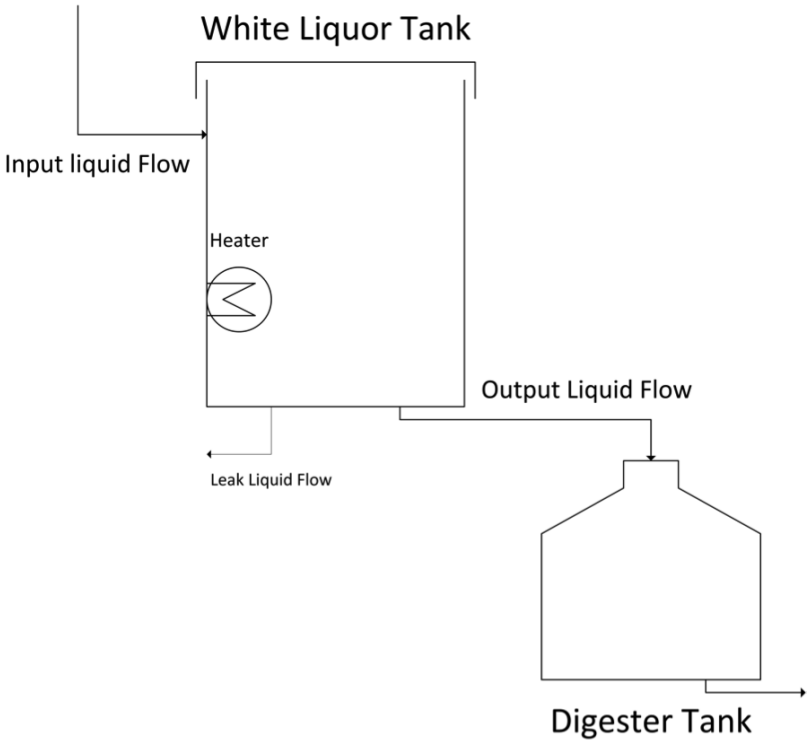


Figure 9.1: Piping and Instrumentation Diagram of the example process.

Figure 9.2: Functional model of the example process.

a configuration flow graph (CFG), which has been implemented in the Simulink tool (see Figure 9.3). The CFG and functional model have the same flows between functions and components, following the taxonomy defined in [133]. This makes it possible for a Function Failure Logic (FFL) to passively observe how abnormal flow levels propagate in the simulated CFG, and to use this information to determine if a function defined in the functional model (Figure 9.2) has been degraded or lost. The FFL for the Supply liquid material (white liquor) function is shown in Figure 9.4; it compares the input and output flows of the white liquor tank shown in Figure 9.3.

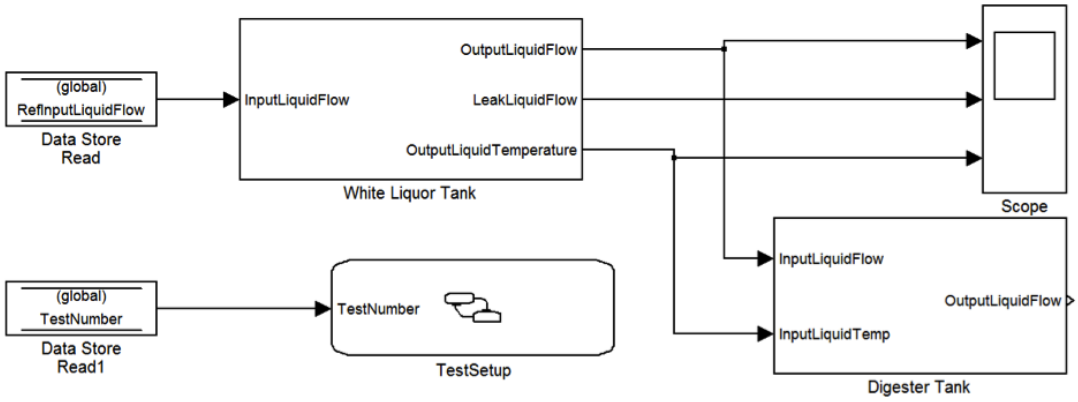


Figure 9.3: Configuration flow graph of the example process.

The relationship between input and output flows of a component in a CFG is defined by a behavioral model (BM). The BM for the white liquor tank is shown in Figure 9.5. Statecharts are used in behavioral modeling, and a state is defined for each nominal and failed mode of the component. In this case there is one failed mode: the tank is leaking. Critical events may be injected to the simulation at any time, and these cause mode changes (e.g. the leakFailure event triggers a transition to the TankLeaking state.)

In earlier versions of the FFIP framework, flow levels were described with an enumeration [zero, low, nominal, high], but this approach is insufficient for our boiling water reactor case study, in which several positive and negative feedback loops affect a single flow. Would two feedback loops simply cancel each other out, if one of them increases a flow and the other decreases it? How can the effect of varying design parameters be captured in the simulation? In order to describe feedback loops in early phase designs,

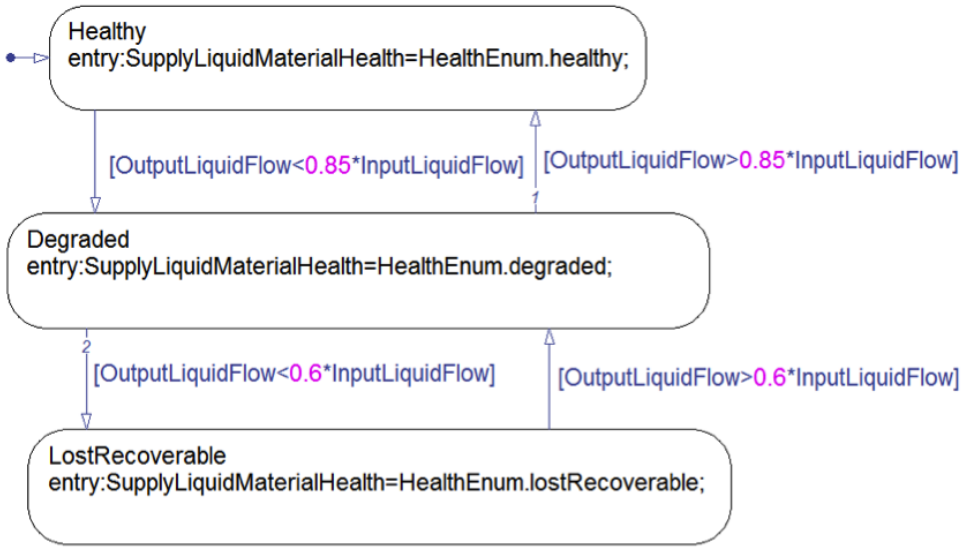


Figure 9.4: Function Failure Logic for the Supply liquid material (white liquor) function.

flow levels may be any value in the range [0..10] in this simulation. First order linear difference equations are used to relate input and output flows to each other. Consider the state Nominal in Figure 9.5. Since the behavioral model is executed by a fixed-step solver, the tank level is a sequence, with one value for each simulation step. The first line of code in the nominal state is a linear difference equation that relates the current and previous elements in the sequence. The behavioral model is thus a system of first order linear difference equations relating the input flows, output flow and components internal variables (such as the tank level). The coefficients may either be fixed numbers or parameters that may be changed between successive simulation runs; an example of the latter is leakSize in the Leaking state.

Since during early phase design detailed component dimensions are not known and the range [0..10] is used for flow levels, the results obviously depend on how the model is parameterized. One objective in this article is to observe how parameter changes in the behavioral models affect the emergent behavior. These parameters may either be design parameters, timing of critical event scenarios or parameters of faults. Changes in values of two parameters are studied in this example. RefInputLiquidFlow in Figure 9.3 is the

Figure 9.5: Behavioral model of the white liquor tank component.

reference liquid flow that should go through the white liquor tank and the digester in normal operation of the process; it is a design parameter. In Figure 9.5 Leaking state, leakSize is a parameter of a fault.

The methodology used in this chapter is to define a number of values of interest for the parameters to be varied and to systematically perform FFIP simulation to identify those combinations of parameter values that result in degradation or loss of functions. The choice of values is done by an analyst who understands the application domain. The flowchart in Figure 9.6 illustrates the methodology in the case of two parameters, and additional parameters can be handled by adding a nested loop for each new parameter. Figure 9.7 displays the graph that is obtained in the flowchart when parameter 1 is RefInputLiquidFlow, Sp1 is [5.0, 10.0, 20.0], parameter 2 is leakSize and Sp2 is [0.5, 2.0, 3.5]; each curve in the graph is the trend for the tank level in one simulation run. The output of Function Failure Logic for the simulation in Figure 9.7 is shown in Table 9.1. The Transmit Thermal Energy function is lost when the tank level drops below the level of the heating element. The status is lost recoverable, since the function can return to healthy when the level rises above the heating element. The Convert Thermal Energy to Chemical function, which is associated with the digester component, may fail due to fault propagation from another component, in this case a leak in the white liquor tank.

9.4 Case Study: Analysis of a Boiling Water Reactor Design

The proposed approach is applied to a Boiling Water Reactor (BWR), focusing on its steam outlets. The BWR uses the thermal energy from nuclear fission inside the reactor core to produce high-pressure steam. The steam is guided via pipes and valves to steam turbines to provide motion to electrical generators. The steam flow to the turbine is controlled by a pressure control valve, such that the pressure inside the reactor vessel is kept steady. Fuel rods are arranged into assemblies, which are submerged under water. The space between the fuel rods forms flow channels through which the coolant pumps circulate water, which acts as both coolant and moderator. As a coolant, it removes thermal energy from the core, and a greater flow rate reduces the void fraction, which is the proportion of steam in the coolant. As a moderator, water slows neutrons, and the number of thermal neutrons is increased when the void fraction is decreased.

Several feedback loops affect the energy production at the core. An increase of

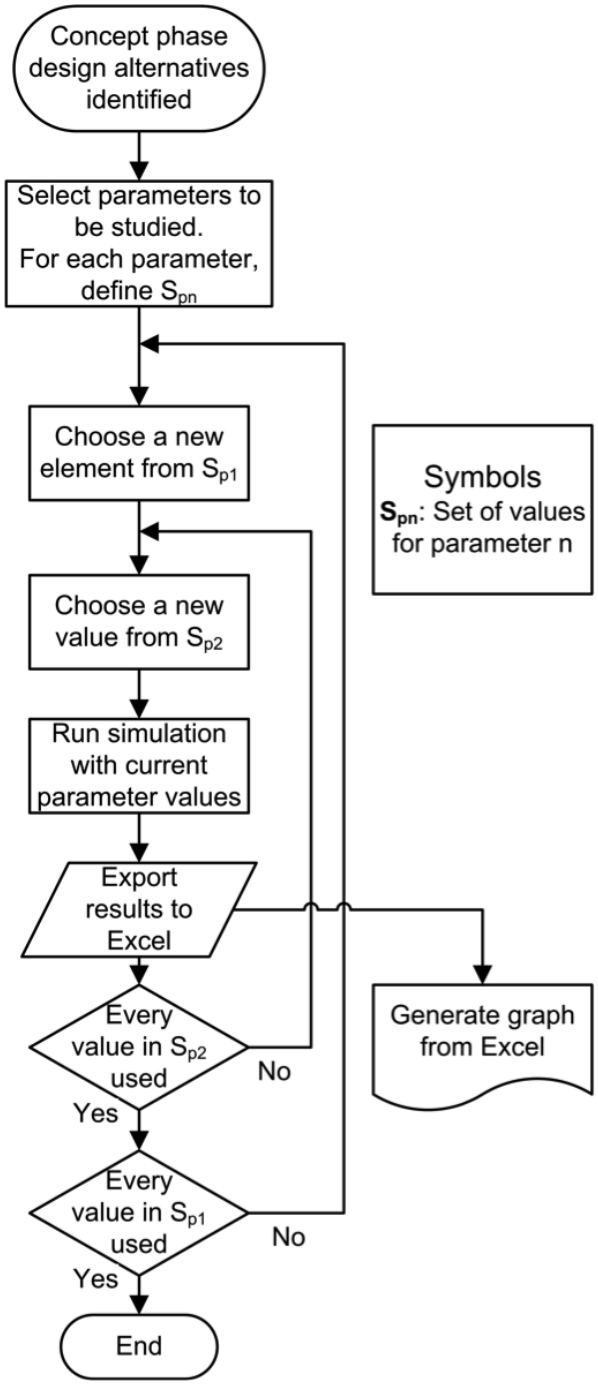


Figure 9.6: A flowchart describing how every combination of parameter values is simulated systematically to determine those combinations of parameter values that result in degradation or loss of functions.

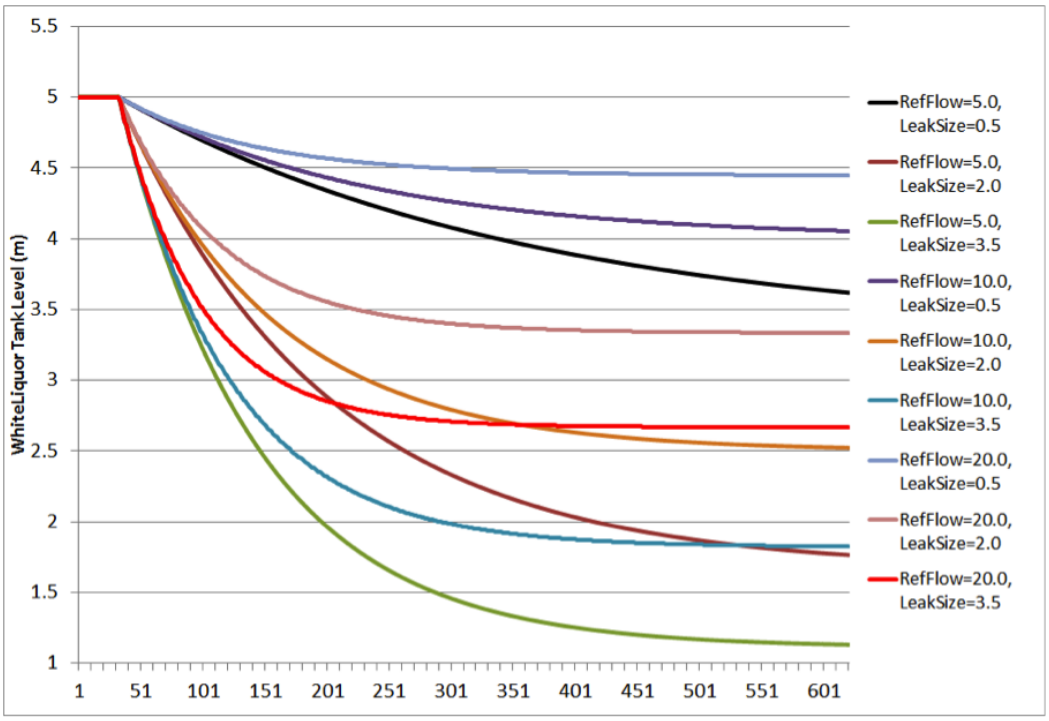


Figure 9.7: The result of performing the procedure in Figure 9.6 when parameter 1 is RefInputLiquidFlow and parameter 2 is leakSize.

Table 9.1: The Output of Function Failure Logic for the Simulation in Figure 9.7

	Supply Liquid Material Health	Transmit Thermal Energy Health	Convert Thermal Energy To Chemical Health
RefFlow=5.0, LeakSize=0.5	Degraded (t232)	Healthy	Degraded (t232)
RefFlow=5.0, LeakSize=2.0	Degraded (t76)	LostRecoverable (t416)	Degraded (t76), Lost Recoverable (t416)
RefFlow=5.0, LeakSize=3.5	Degraded (t57), LostRecoverable (t113)	LostRecoverable (t197)	Degraded (t57), LostRecoverable (t197)
RefFlow=10.0, LeakSize=0.5	Degraded (t310)	Healthy	Degraded (t310)
RefFlow=10.0, LeakSize=2.0	Degraded (t78), LostRecoverable (t234)	Healthy	Degraded (t78)
RefFlow=10.0, LeakSize=3.5	Degraded (t58), LostRecoverable (t123)	LostRecoverable (t292)	Degraded (t58), LostRecoverable (t292)
RefFlow=20.0, LeakSize=0.5	Healthy	Healthy	Healthy
RefFlow=20.0, LeakSize=2.0	Degraded (t83)	Healthy	Degraded (t83)
RefFlow=20.0, LeakSize=3.5	Degraded (t59), LostRecoverable (t162)	Healthy	Degraded (t59)

pressure in the reactor vessel decreases the void fraction and increases power output. A decrease of coolant flow rate has the opposite effect, but insufficient flow can cause a heat transfer crisis from fuel rods to coolant due to steam buildup; this can lead to fuel damage due to overheating of the rods. The Doppler Effect can also reduce the neutron flux. This is a physical phenomenon in which neutron absorption (without fission) by the 238U and 240Pu isotopes increases when the fuel temperature increases [210].

A reactor protection system is present which can initiate an emergency shutdown of the reactor, referred to as a scram. Some conditions that can trigger a scram are related to the neutron flux exceeding a threshold value, the coolant flow rate being below a threshold value or the external electrical power coming from the power grid being lost. Maintaining coolant flow at all times is necessary to prevent steam buildup at the core, since this may cause a heat transfer crisis and damage to the fuel rods due to overheating. However, a BWR has positive feedback between reactor power and coolant flow, so a controlled decrease of flow in emergency situations is necessary. During scram, a common design in a BWR is to drive coolant pumps down to minimum rotations per minute gradually by a ramp. The slope of this ramp and the need for emergency power for the pumps are design parameters of interest in this case study.

There are some known hazards to maintaining acceptable pressure and flow in the reactor vessel that a BWR design must mitigate. The pressure control valve in the steam outlet pipe is driven by a sophisticated controller that must react rapidly to changes. If the pressure control subsystem closes the valve, it notifies the turbine protection subsystem, which relieves the pressure by opening the dumper valve. However, due to the sophisticated nature of the pressure control subsystem, unintentional closure of the valve due to software malfunction is possible, resulting in no notification and a pressure shockwave back to the reactor vessel.

9.4.1 Abstractions: Functional Model and Configuration Flow Graph

The first step in analyzing the early design of the BWR is identification of desired functionality. The processes inside a BWR are expressed through functions. These functions are aggregated and connected to create a structure based on the flows of energy, material and signals (EMS) affected by each function. This function structure is known as a functional model (FM). The Functional Basis [133] is used as a taxonomy for naming the functions and the flows. The scope of the case study is restricted to acute situations that might require emergency shutdown of the reactor. The model is not intended for studying residual heat removal, which is a process that can take months, involving additional physical phenomena and technical systems that are not included in this model.

A configuration flow graph (CFG) is produced by selecting components to implement the functions, and connecting them with the same EMS flows as in the functional model. Table 9.2 contains the direct mapping between some of the functions and components for a BWR design used in the FFIP analysis. The components include mechanical, electrical and software parts and can contain components hierarchically to obtain the level of granularity needed to support mapping between the CFG and FM. In Figure 9.8, the top level CFG is provided, and the internals of the reactor component are shown in Figure 9.9.

The reason for having separate a FM and CFG is that, due to fault propagation, a component failure may result in degradation or loss of functionality mapped to another component. For example, a leak in a pipe (a faulty component state) can cause a degraded functional state for a store liquid function, even though the tank component

Table 9.2: Mapping of System Functions to Components

Function	Component
Transmit thermal energy	Flow channels and Fuel Rods
Condition radioactive energy	Flow channels
Supply electrical energy	Power supply rail
Transport liquid material	Coolant pumps
Process status signals	Reactor protection
Inhibit radioactive energy	Control rods
Regulate and guide gas material	Pressure control valve
Process status signals	Pressure control

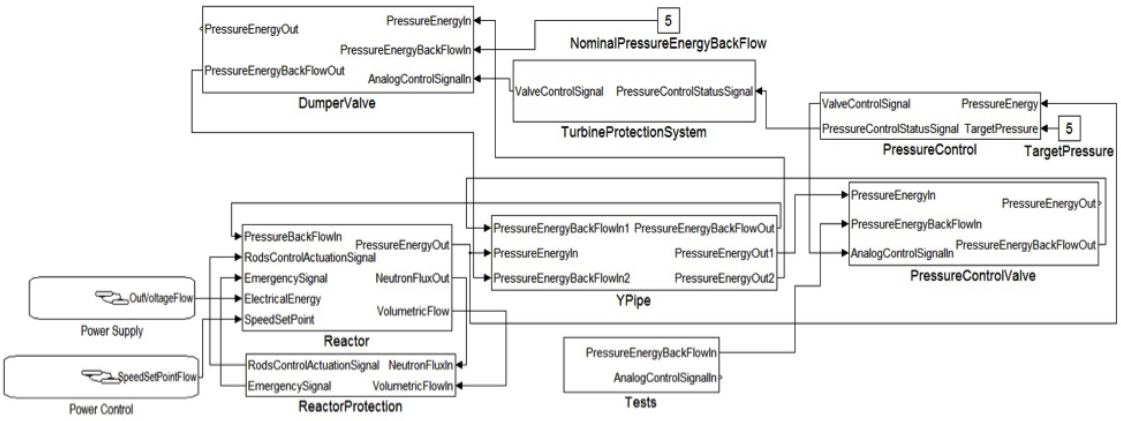


Figure 9.8: Top level CFG model for boiling water reactor core and its steam outlets.

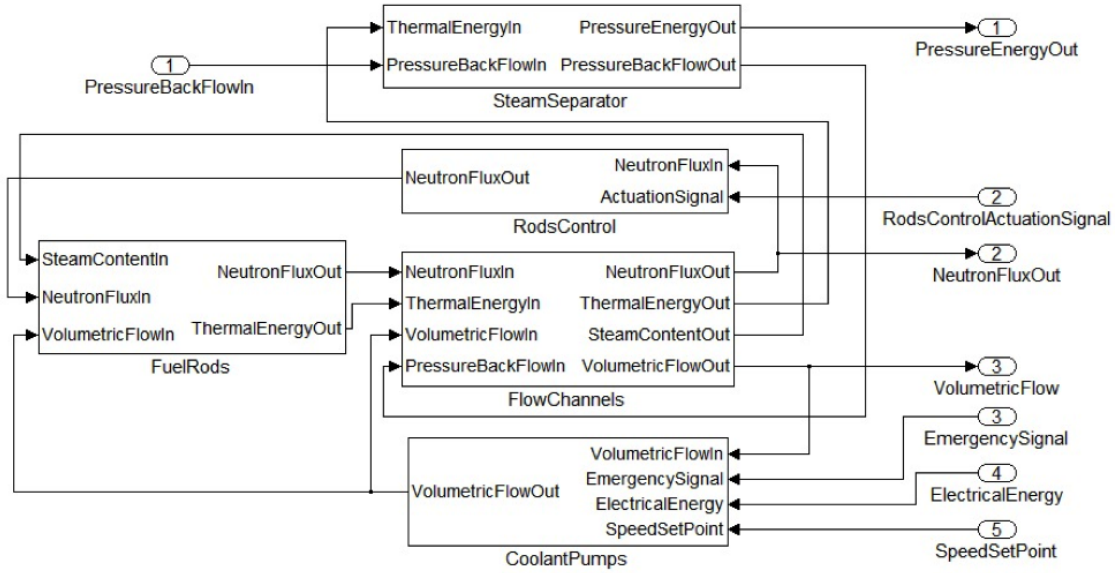


Figure 9.9: Internals of the Reactor component in Figure 9.8.

mapped to this function is in a nominal state. In the analysis, the CFG is simulated, critical events are inserted as faulty component states, and a logic relating function to failures, namely, the Functional Failure Logic (FFL), is used to observe component input and output flows and determine the health of the related function. This is possible because the FM and CFG use the same EMS flows.

9.4.2 Behavior and Function-Failure Logic Reasoning

The behavior of the components is implemented using the Simulink environment and the Stateflow package, which are part of the Matlab tool. Stateflow models are used to represent hybrid behavior. There is a state for each nominal and faulty component mode, and within these states there can be continuous behavior. An example of behavior logic for the pressure control valve component is demonstrated in Figure 9.10. State transitions are triggered by critical events that are inserted into the simulation. The behavioral models should be constructed by experts familiar with the application domain, the physical phenomena and the impact of various parameter value changes on these

phenomena and design alternatives.

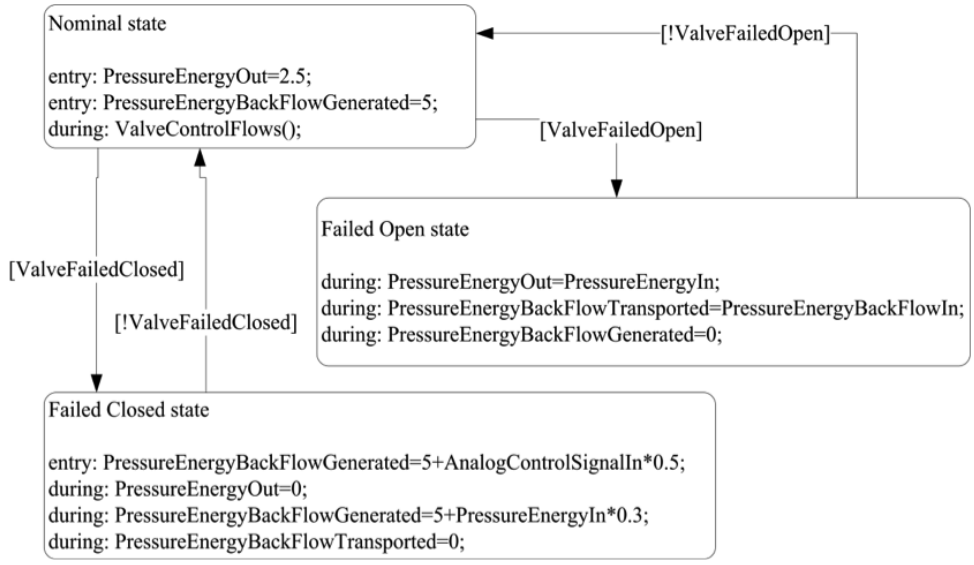


Figure 9.10: Behavioral logic for the PressureControlValve component in Figure 9.8.

The FFL reasoner code for the "transmit thermal energy" function, which is partly implemented by the fuel rods, is presented in Figure 9.11. The FFL reasoner determines the health of this function by the value of the temperature parameter during the simulation. The values of 5.5, 6 and 7 for entering the health status "degraded", "lost recoverable" and "lost" respectively are based on qualitative descriptions for the Temperature levels. The value of 5 is defined as nominal.

9.5 Automation framework and user interface

Figure 9.12 shows a screenshot the user interface for specifying a set of FFIP simulation scenarios to be generated and run automatically. After specifying the Simulink.m file containing the parameter, the user can select any of these parameters and then specify the limits of its range and the interval between samples. From the selections in the "choices made" box, the heat transfer coefficient (HTC) will be given values [1, 3, 5] in separate simulations. Similarly, the time delay between the injected pressure control valve closure malfunction and the injected power rail failure (Delay) will be one of the

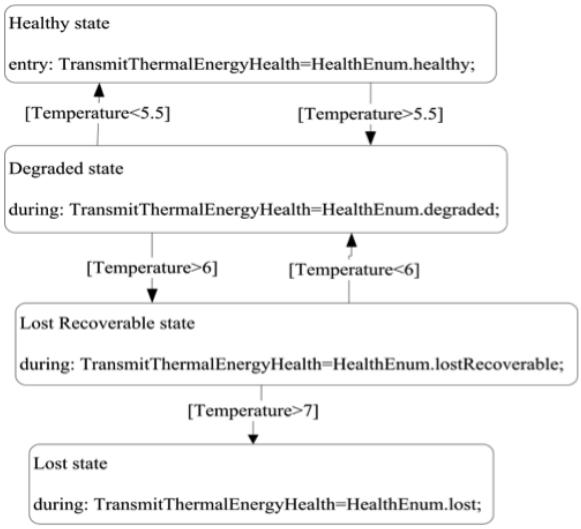


Figure 9.11: Stateflow chart of the FFL reasoner for the "Transmit thermal energy" function in Table 9.1

following [0, 40, 80]. The slope of the ramp for driving the coolant pumps to minimum rotations per minute (Slope) is one of: [-0.01, -0.015, -0.02]. The simulation will be run with each combination of parameter values, resulting in 27 simulation runs. For each of these runs, the value of one signal is logged for each step of the discrete time simulation; in Fig. 12 the function health of "Transmit thermal energy" from fuel rods to coolant is selected. The process is repeated so that the Temperature of the fuel rods is logged, resulting in 27 temperature curves and information of the worst case health status of each curve; in the next section, the health status is used to group the curves onto separate charts for readability.

The algorithm for generating the set of parameterized FFIP simulations is shown in Figure 9.13. The information entered through the user interface is stored into a list, with one element for each parameter; in this case the parameters are HTC, Delay and Slope. A CreateLoop function is called, which iterates through the list in such a way that every combination of parameters is covered. This is accomplished by a recursive function call shown with a bold dashed arrow. This design ensures that the No-branch of decision 1 is taken exactly once for each combination of parameter values, resulting in an entry in the Matlab script for running the FFIP simulation with those values. Recursion was

used to improve readability and compactness of code [211, 212].

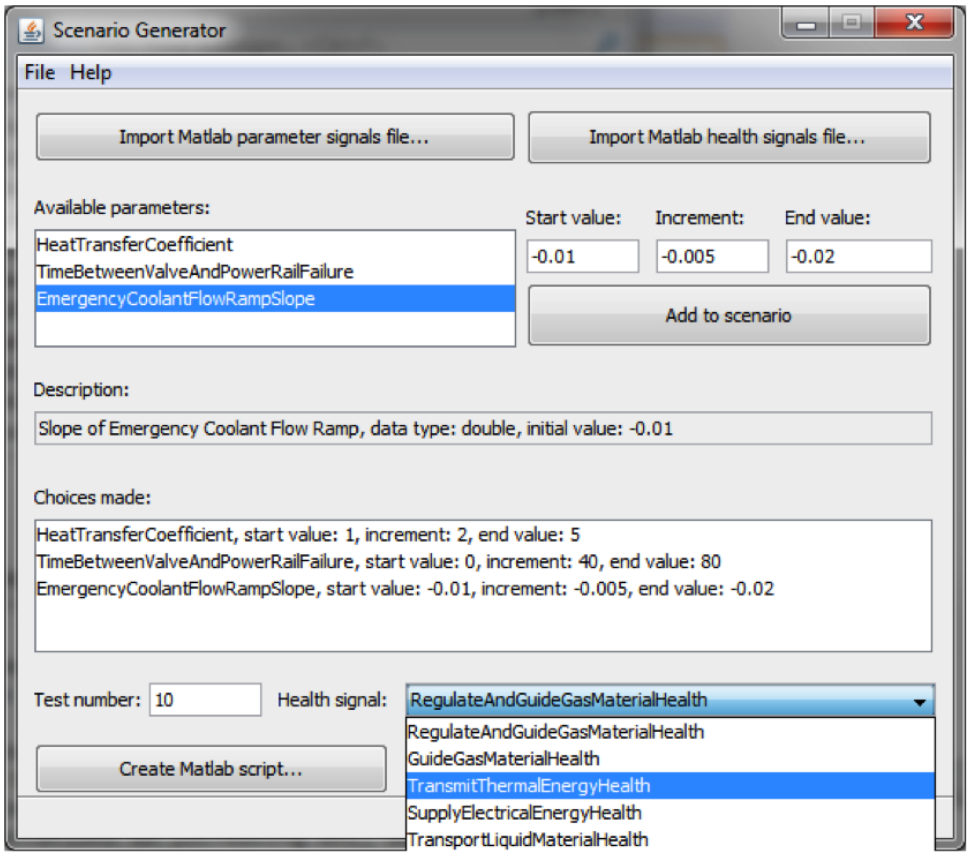


Figure 9.12: User interface for specifying a set of FFIP simulation scenarios.

9.6 Simulation Results

In the initial critical event scenario, emergency power for coolant pumps was cut off before the pressure shockwave occurred. In the user interface, three different values are selected for HTC, the slope for driving coolant pumps to minimum rotations per minute (Slope) and the time delay between the unintentional pressure control valve closure and power rail failure (Delay). The medium value is the designer's best estimation of a realistic design parameter, while the low and high limits are given the most extreme values that

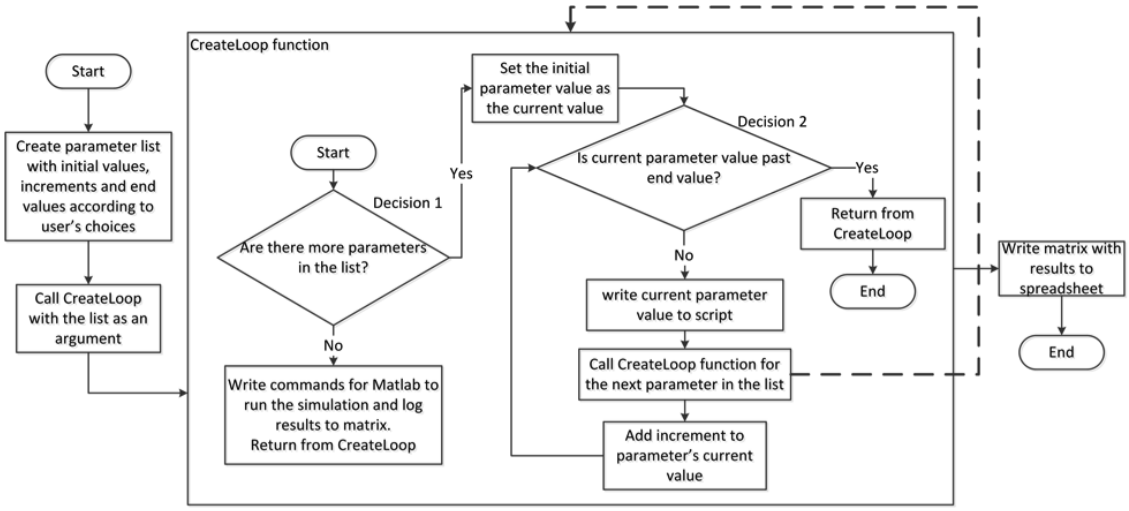


Figure 9.13: Algorithm for generating the set of parameterized FFIP simulations.

are estimated to be feasible. This first phase of the simulation will result in 27 fuel rod temperature curves, one for running the simulation with each combination of parameter values. The selected function of interest is thermal energy transfer from fuel rods, so the Excel output for each simulation run includes the lowest functional health status of this function during the simulation. Accordingly, the curves for fuel rod temperature are grouped in three graphs: the simulation runs during which this function's health was always healthy (Figure 9.14), the runs during which the health was never worse than degraded (Figure 9.15) and the runs during which this function was lost (Figure 9.16). When several curves were overlapping and visually indistinguishable on the graph, the parameter values corresponding to them are shown in square brackets in the legend; for example, in Figure 9.16, the temperature curves for HTC=2 and Delay=0 were identical for all values of Slope.

It is clear that better performance was obtained when the power rail failure occurred later, since coolant flow was maintained for a longer time. This indicates that from the perspective of fuel rod health, steam removal from the core is more important than an acute drop in reactor power, which could be achieved by stopping the pumps. A very low value for the HTC indicated stability problems regardless of the other values, and this is a physical design constraint for this reactor type. A very high value for HTC provided

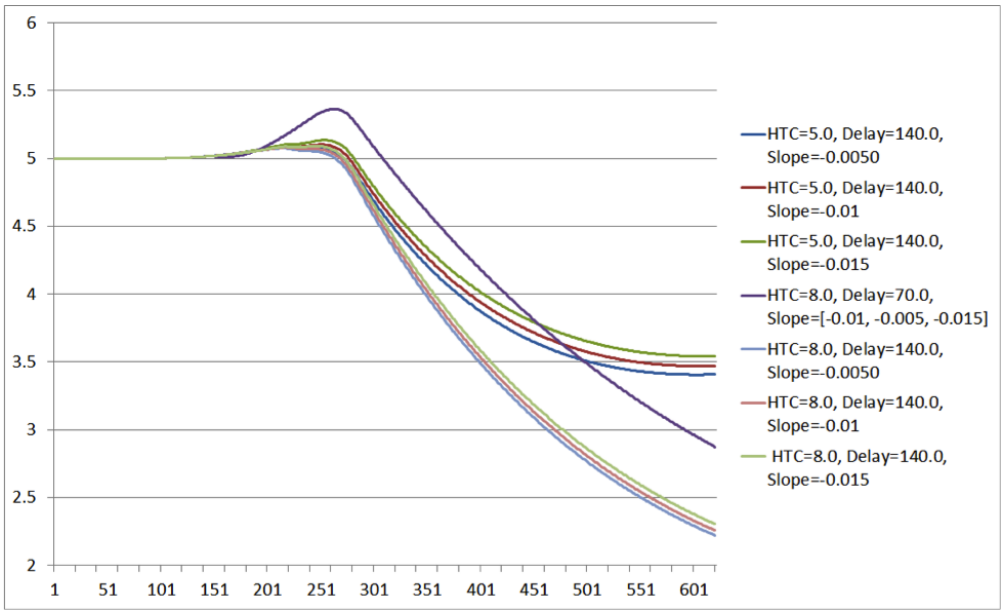


Figure 9.14: Temperature of fuel rods in first phase FFIP simulation scenarios with parameter values resulting in healthy FFL verdicts.

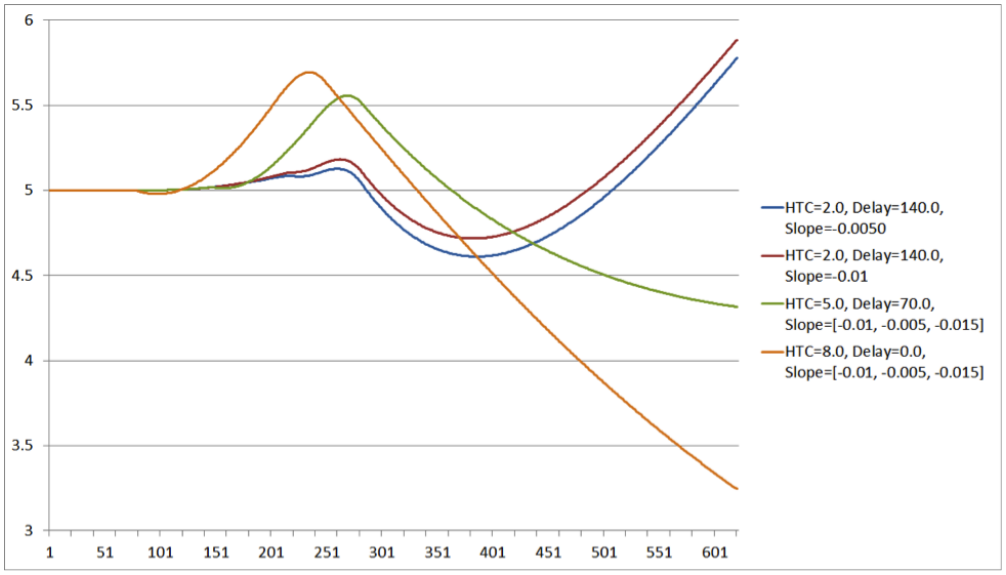


Figure 9.15: Temperature of fuel rods in first phase FFIP simulation scenarios with parameter values resulting in degraded FFL verdicts.

consistently stable behavior, although a temporary overheating of fuel rods is possible; however, it might be physically impossible to realize this design. With medium values of HTC, fuel rod cooling depends primarily on the availability of power for the coolant pumps and secondarily on having a smaller slope for the ramp that drives down the rotations per minute of the coolant pumps. Based on this analysis, it is concluded that the availability of power for coolant pumps must be ensured with emergency power, so a different critical event scenario, in which the emergency power is not cut off, is selected as the basis for the next set of simulation runs. In this case, the time delay between the shockwave and power rail failure does not significantly influence the results, so this parameter is excluded from further study.

The second phase of FFIP simulation focuses on the combinations of HTC and ramp slope, resulting in 9 automatically executed simulations. From Figure 9.17 it is clear that when emergency power is present, stable performance can be achieved even with lower HTC values when a gentler slope is used for the ramp. However, a longer ramp implies a slower decrease of reactor power during emergency shutdown, so a third simulation run is performed by narrowing the parameter values to the range in which these two parameters

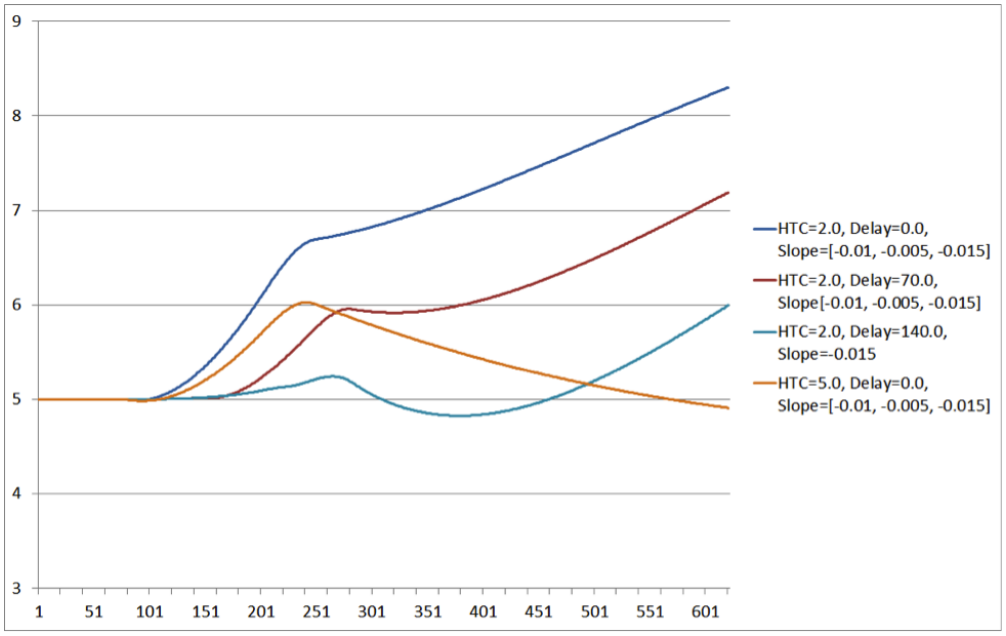


Figure 9.16: Temperature of fuel rods in first phase FFIP simulation scenarios with parameter values resulting in lost FFL verdicts.

are likely to be optimized. The results in Figure 9.18 are given as a starting point for detailed design, which will refine the conceptual design through specific decisions on component types, subsystem design and control algorithm design. One decision that emerges already from the simulation of the BWR system design is that availability of electric power for coolant pumps must be ensured by emergency power supplies even when the power rail is lost.

In every scenario on Figure 9.18, the fuel rod cooling is performed satisfactorily during the acute situation requiring emergency shutdown of the reactor. The simulation time is set to a value that is of interest for the emergency shutdown safety function. It is unclear if the temperature would remain below nominal if the simulation time were to be increased. In the case study, it was stated that the model covers only emergency shutdown functionality and is not aimed at studying residual heat removal, which has a much longer time constant; this would require additional detail to the system model and a different set of simulation runs. The emergency shutdown of the reactor is used in this chapter to demonstrate the application of FFIP automation framework.

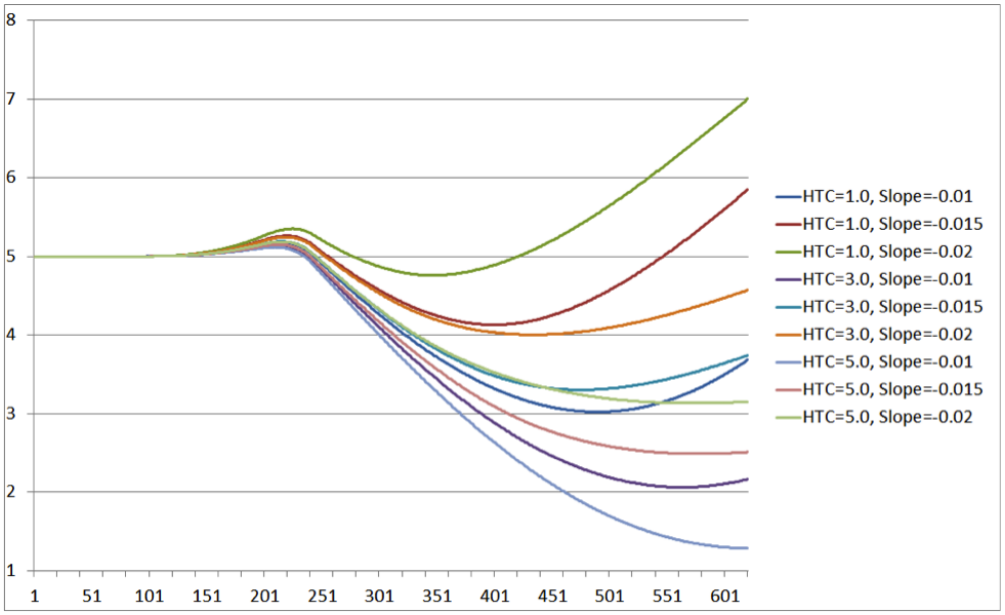


Figure 9.17: Temperature of fuel rods in second phase FFIP simulation: emergency power present.

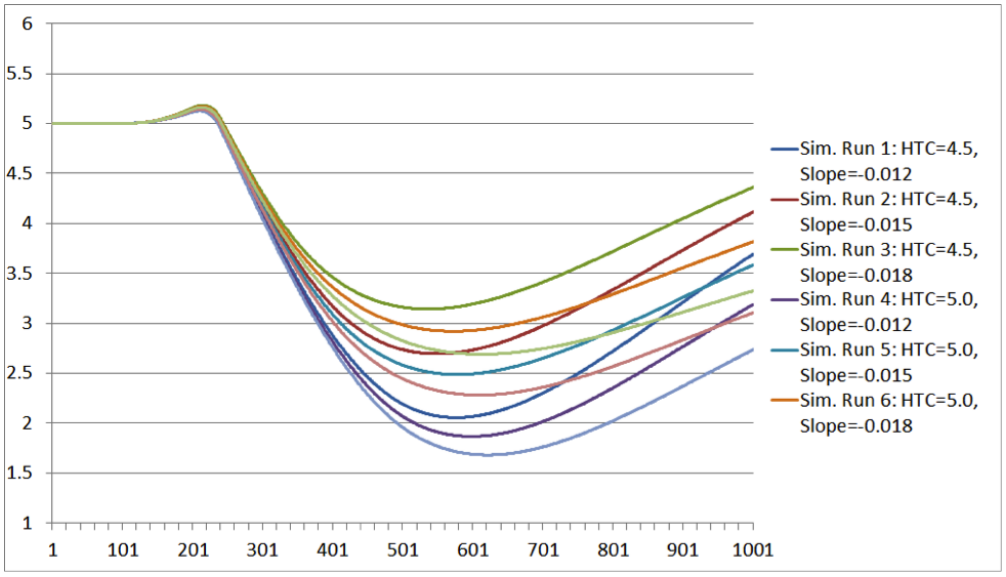


Figure 9.18: Temperature of fuel rods in third phase FFIP simulation scenarios with narrowed ranges for parameter values.

9.7 Conclusions and Future Work

The design and failure analysis of large complex systems presents numerous challenges. While the FFIP framework developed previously was intended to meet some of these challenges, the extensions presented in this chapter greatly increase the ability of this analysis approach to be used as a tool in the early concurrent design and risk assessment processes. The extensions to the FFIP framework presented here enable the study of the reliability of a range of design alternatives for achieving the specified functionality. For alternatives that can be expressed in terms of parameters of component behavior or timing of critical events, the scalability of the approach and supporting tools to a larger number of simulation runs is only limited by computing power and human capability of understanding large data sets. The latter problem is mitigated by the possibility of filtering and sorting the simulation runs in terms of the health status of a specific function, but further research on user interface design will improve the feasibility of the approach for complex industrial-scale systems. The behavior of electromechanical

components is described with first-order linear approximations in this case study, but they are implemented as Simulink blocks, so the approach permits more sophisticated modeling if it is deemed appropriate in the early concept design phase

The investigation of a range of designs under a range of critical event scenarios is restricted to alternatives that can be expressed by varying values of parameters. The availability of emergency power could be controlled manually by selecting different simulation scenarios, but currently it is not possible to cover more fundamental differences such as different types or placement of sensors or entirely different software algorithms. This can be achieved with further work on integrating a feature modeling capacity to the configuration flow graph, so that every valid configuration of the feature model will undergo FFIP simulation in order to filter out unreliable alternatives. The automated FFIP simulation presented in this chapter solves many algorithmic and technical challenges related to the generation and simulation of every valid configuration.

Chapter 10: Conclusions

This research presented a design, simulation and analysis method for enabling safety-centric design decision making. In this research, a novel method representing the safety property of system as safety functions was introduced and incorporated into the functional system design process. The goal of explicitly including safety functions is to allow behavioral simulation and model-based reasoning to be used to relate component faults to system level hazards and potential mishaps. A major focus of this work was the process of identifying and simulating these fault scenarios. Finally, while simulating each scenario provided an evaluation of impact in terms of component and safety function states, effective design decision making requires analysis over many scenarios. Additionally, we present three methods of grouping the scenario results to provide different views of the system to the designer.

To demonstrate the methodology of this research we considered the design of a spacecraft maneuvering system. Chapter 4 described how the functional system design method is augmented to include the safety property of the system. Specifically, when a desired hazard-to-mishap transition mitigation property was identified a safety function is implemented. Using the presented process, the safety function was mapped to the components architecture that followed a functional decomposition approach. This explicit mapping using models allows for a system simulation of component behavior to be used to reason about the effect on safety as well as functional impact. To demonstrate the detail of this early design analysis of the complex system we considered the subsystems of the maneuvering system. Chapter 5 used the electrical power subsystem to demonstrate that through component fault simulation the functional effect could be quantified for making design selection decisions. Recognizing that complex systems often fail in complex ways, Chapter 6 illustrated using the thruster subsystem an approach for identifying novel fault scenarios. That is, in addition to modeling single failure modes, this chapter showed how mapping between known fault symptoms and fault causes can be used to find potential cascading failure scenarios. Finally, the analysis part of this research is wrapped up in Chapter 7. In this chapter we presented the reasoning approach that

can use the entire system behavior simulation to identify any potential loss in the safety function specified from Chapter 4. The final phase of this research is presented in Chapters 8 and 9. First, it is clear that designers of complex systems would like to evaluate system performance over a large set of scenarios. However, the function-based analysis of this work can provide designers with more than simple ratings for a design. Instead, a shopping or design exploration approach is adopted. To that end, Chapter 8 presented three methods of exploring the analysis of the electrical power subsystem by clustering the scenarios simulation results. The first method identified and ranked scenarios for risk. The second method abstracted the system-level failure behavior into probabilistic functional effects for new scenarios or large system modeling. The third method grouped scenario results based on weighted similar functional effect to a specified system functional state. Finally, to demonstrate that design can benefit from analysis exploration, Chapter 9 presents an analysis sandbox approach that was used for understanding the effect of a critical scenario on a boiling water nuclear reactor.

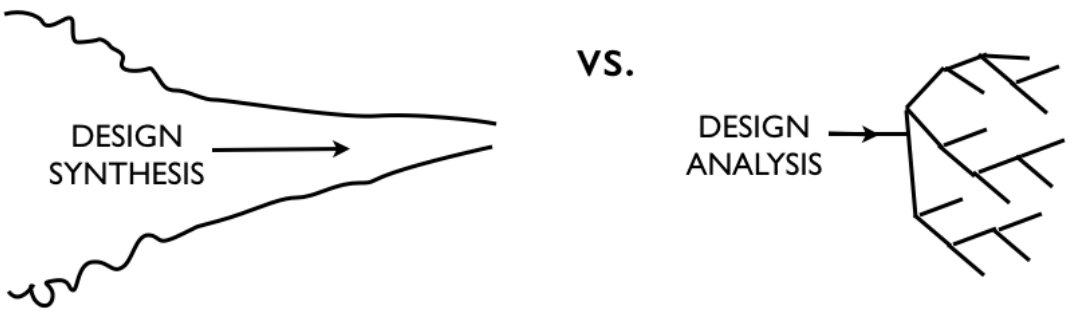


Figure 10.1: Distinguishing synthesis from analysis in the context of design decision making.

Throughout this research the design process has been identified as a decision making process where this research aims to aid that decision making process. However, decision theory supports the idea that decision making (and thus design) is a synthesis activity [213]. A major focus of this work was on analysis. In contrast to synthesis, analysis is a process of identifying all the constituent parts of the problem. The difference between synthesis and analysis are illustrated in Figure 10.1. From the decision theory perspective

detailed failure analysis serves to rate a design for selection among competing options (as synthesis activity). While this research does serve that purpose and provides a means of evaluating designs for safety in a way not previously possible, this work also presented a different role for analysis in general. That is, the process of failure analysis was used as a sandbox for exploring a design, providing engineers with a deeper understanding of the problem and the design solution. Further, this activity is useful in the design context for identifying novel solutions. Therefore, this research indicates that analysis might have a greater role in driving the design process if there is flexibility in following the traditional V-diagram approach to system design. In the broader context of engineering design, this research highlights the role of analysis in supporting the synthesis process. That is, the process of narrowing down options from a potential design space is can be guided not just by the functional goals but the safety concerns as well. Further, iterative loops of concept analysis exploration can identify unexplored dimensions of the design space. This indicates that a model of design where instead of a linear process of design definition, concept analysis is integrated to regularly provide concept assessment and a tool for knowledge expansion.

In many areas, such as energy and transportation, complex systems have a growing presence in modern life. The risk in capital investment and human life when these systems become unsafe can be catastrophic. Further, the nature of these systems leads to complex and emergent failure behavior. This work begins to address how safety and risk can direct design from the earliest stages towards safe system operation. While no system can be free from all potential failure, a focus on safety can change how unexpected failures affect the system. Many products are designed to fail into a safe mode. A design process focused on increasing reliability but not on the system property of safety may miss design solutions that lead to fail-safe concepts. Additionally, a design process focused on hazard avoidance may not properly prioritize those hazards and may miss potential triggers for those hazards without an assessment of the cause and propagation of failures. The framework developed in this research captures both perspectives and can lead to improved system design.

Bibliography

[1] Suh, N., 2005, *Complexity: theory and applications*, Oxford University Press, USA.

[2] Lee, E. A., 2008, "Cyber physical systems: Design challenges," Tech. Rep. UCB/EECS-2008-8, EECS Department, University of California, Berkeley, URL `http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-8.html`.

[3] Checkland, P., 1999, "Systems thinking," Rethinking management information systems: An interdisciplinary perspective, pp. 45–55.

[4] Sage, A., 1992, *Systems engineering*, John Wiley & Sons, New York.

[5] Smith, B., 1964, *The Rand Corporation: case study of a non-profit advisory corporation*, Ph.D. thesis, Harvard University.

[6] Goode, H. and Machol, R., 1957, *System engineering: an introduction to the design of large-scale systems*, McGraw-Hill.

[7] Hall, A., 1962, *A methodology for systems engineering*, Van Nostrand.

[8] Military Standard (MIL-STD 499B), 1993, "Systems engineering management," Department of Defense.

[9] Sage, A. and Armstrong, J., 2000, *Introduction to systems engineering*, John Wiley & Sons, New York.

[10] Blanchard, B. and Fabrycky, W., 1990, *Systems Engineering and Analysis*, Prentice Hall Upper Saddle River.

[11] Buede, D., 2009, *The engineering design of systems: Models and methods*, John Wiley & Sons, New York.

[12] Chestnut, H., 1967, *Systems engineering methods*, John Wiley & Sons, New York.

[13] Wymore, A., 1967, *A mathematical theory of systems engineering: The elements*, John Wiley & Sons, New York.

[14] Wymore, A., 1993, *Model-based systems engineering*, CRC Press, Inc. Boca Raton, FL, USA.

[15] Hawryszkiewycz, I., 1991, *Introduction to systems analysis and design*, Prentice-Hall, Inc. Upper Saddle River, NJ, USA.

[16] Wallace, R., Stockenberg, J., and Charette, R., 1987, *A unified methodology for developing systems*, Intertext Publications, Inc.,/McGraw-Hill, Inc. New York, NY, USA.

[17] Chapman, W. L., Bahill, A. T., and Wymore, A. W., 1996, *Engineering modeling and design*, CRC Press, London, UK.

[18] DARPA, October, 2010, "Adaptive Vehicle Make," Overview Slides, URL `http://tinyurl.com/896tgpl(accessedMarch,2012)`.

[19] Augustine, N., 1997, *Augustine's laws*, AIAA (American Institute of Aeronautics & Astronautics).

[20] Henley, E. and Kumamoto, H., 1981, *Reliability engineering and risk assessment*, vol. 193, Prentice-Hall Englewood Cliffs (NJ).

[21] Ericson, C., 2005, *Hazard analysis techniques for system safety*, John Wiley and Sons.

[22] Farmer, F., 1967, "Siting criteria: A new approach." Tech. rep., United Kingdom Atomic Energy Authority, Risley, Eng.

[23] Pahl, G., Beitz, W., and Wallace, K., 1996, *Engineering design: a systematic approach*, Springer Verlag.

[24] Van Bossuyt, D., Tumer, I., and Wall, S., 2012, "A Case for Trading Risk in Conceptual Design Trade Studies," Research in Engineering Design, **in review**.

[25] Lamassoure, E. S., Wall, S. D., and Easter, R. W., 2004, "Model-based engineering design for trade space exploration throughout the design cycle," *Proceedings of the Space 2004 Conference and Exhibit*, AIAA 2004-5855.

[26] Weilkiens, T., 2007, *Systems engineering with SysML/UML: modeling, analysis, design*, Morgan Kaufmann.

[27] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W., *Object oriented modeling and design*, Prentice Hall, West Nyack, NY.

[28] Papalambros, P. Y. and Wilde, D., 2002, *Principles of optimal design: modeling and computation*, Cambridge University Press.

[29] Sasajima, M., Kitamura, Y., Mitsuru, I., and Mizoguchi, R., 1996, "A representation language for behavior and function: FBRL," Expert Systems with Aplications, **10**(3-4), pp. 471–479.

[30] Qian, L. and Gero, J., 1996, "Function-behaviour-structure and their roles in analogy-based design," Artificial Intelligence in Engineering Design, Analysis and Manufacture, **10**, pp. 289–312.

[31] Otto, K. N. and Wood, K. L., 2001, *Product Design: Techniques in reverse engineering and new product development*, Prentice Hall.

[32] Ulrich, K. and Eppinger, S., 1995, *Product design and development*, McGraw-Hill.

[33] Matthiesen, S., 2002, "A contribution to the basis of the element model: Working surface pairs and channel and support structures on the correlation between layout and function of technical systems," Institute for Product Development, Technical University Karlsruhe, Karlsruhe, Germany.

[34] Albers, A., Braun, A., Sadowski, E., Wynn, D. C., Wyatt, D. F., and Clarkson, P. J., 2011, "System architecture modeling in a software tool based on the contact and channel approach (C&C-A)," Journal of Mechanical Design, **133**(10), 101006.

[35] Hansen, C. and Andreasen, M., 2002, "Two approaches to synthesis based on the domain theory," Engineering Design Synthesis, **356**, pp. 93–108.

[36] Antonsson, E. and Cagan, J., 2001, *Formal Engineering Design Synthesis*, Cambridge University Press.

[37] Chakrabarti, A., 2002, *Engineering design synthesis: understanding, approaches, and tools*, Springer Verlag.

[38] Bracewell, R. and Sharpe, J., 1996, "Functional descriptions used in computer support for qualitative scheme generation-" schemebuilder"," Artificial Intelligence for Engineering Design, Analysis and Manufacturing, **10**, pp. 333–346.

[39] Stone, R., Wood, K., and Crawford, R., 2000, "Using Quantitative Functional Models to Develop Product Architectures," Design Studies, **21**, pp. 239–260.

[40] Ulrich, K. and Seering, W., 1989, "Synthesis of schematic descriptions in mechanical design," Research in Engineering Design, **1**(1), pp. 3–18.

[41] Altshuller, G. and Rodman, S., 1999, *The innovation algorithm: TRIZ, systematic innovation and technical creativity*, Technical Innovation Ctr.

[42] Bohm, M. and Stone, R., "A natural language to component term methodology: Towards a form based concept generation tool," *Proceedings of the ASME 2009 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*.

[43] Hubka, V. and Eder, W., 1996, *Design science*, Springer.

[44] Kurtoglu, T. and Campbell, M., 2009, "Automated synthesis of electromechanical design configurations from empirical analysis of function to form mapping," Journal of Engineering Design, **20**(1), pp. 83–104.

[45] Thoft-Christensen, P. and Baker, M., 1982, *Structural reliability theory and its applications*, Springer Berlin et al.

[46] Carter, A., 1986, *Mechanical reliability*, vol. 1, Macmillan London.

[47] Bain, L. and Engelhardt, M., 1991, *Statistical analysis of reliability and life-testing models: theory and methods*, vol. 115, CRC.

[48] Kececioglu, D., 2002, *Reliability engineering handbook*, vol. 1, DEStech Publications, Inc.

[49] Distefano, S. and Xing, L., 2006, "A new approach to modeling the system reliability: dynamic reliability block diagrams," *Reliability and Maintainability Symposium, 2006. RAMS'06. Annual*, Ieee, pp. 189–195.

[50] Modarres, M., Kaminskiy, M., and Krivtsov, V., 1999, *Reliability engineering and risk analysis: a practical guide*, vol. 55, CRC.

[51] Xue, J. and Yang, K., 1995, "Dynamic reliability analysis of coherent multistate systems," Reliability, IEEE Transactions on, **44**(4), pp. 683–688.

[52] Pukite, P. and Pukite, J., 1998, *Markov Modeling for Reliability Analysis*, Wiley-IEEE Press.

[53] MIL-STD-1629A, *Procedures for Performing Failure Mode, Effects, and Criticality Analysis*, Department of Defense.

[54] Rugina, A., Kanoun, K., and Kaâniche, M., 2008, "The adapt tool: From aadl architectural models to stochastic petri nets through model transformation," *Dependable Computing Conference, 2008. EDCC 2008. Seventh European*, IEEE, pp. 85–90.

[55] Barkai, J., 1999, *Automatic Generation of a Diagnostic Expert System From Failure Mode and Effects Analysis (FMEA) Information*, Society of Automotive Engineers, 400 Commonwealth Dr, Warrendale, PA, 15096, USA.

[56] Price, C., Pugh, D., Wilson, M., and Snooke, N., 1995, "The flame system: Automating electrical failure mode and effects analysis (fmea)," *Reliability and Maintainability Symposium, 1995. Proceedings., Annual*, IEEE, pp. 90–95.

[57] Russomanno, D., Bonnell, R., and Bowles, J., 1993, "Functional reasoning in a failure modes and effects analysis (FMEA) expert system," *Reliability and Maintainability Symposium, 1993. Proceedings., Annual*, IEEE, pp. 339–347.

[58] Taguchi, G. and Organization, A. P., 1986, *Introduction to quality engineering: designing quality into products and processes*, The Organization Tokyo.

[59] Maass, E. and McNair, P., 2009, *Applying Design for Six Sigma to Software and Hardware Systems*, Prentice Hall.

[60] Wang, K. and Jin, Y., 2002, "An analytical approach to function design," *14th Int. Conference on Design Theory and Methodology IDETC/CIE2002*, pp. 449–459.

[61] Smith, J. and Clarkson, P. J., 2005, "Design concept modelling to improve reliability." Journal of Engineering Design, **16**(5), pp. 473 – 492.

[62] Kurtoglu, T. and Tumer, I. Y., 2008, "A graph-based fault identification and propagation framework for functional design of complex systems," Journal of Mechanical Design, **130**(5).

[63] Huang, Z. and Jin, Y., 2008, "Conceptual Stress and Conceptual Strength for Functional Design-for-Reliability," *Proceedings of the ASME Design Engineering Technical Conferences; International Design Theory and Methodology Conference*.

[64] Stone, R. B., Tumer, I. Y., and VanWie, M., 2005, "The Function Failure Design Method," Journal of Mechanical Design, **14**, pp. 25–33.

[65] Grantham-Lough, K., Stone, R. B., and Tumer, I. Y., 2009, "The risk in early design method," Journal of Engineering Design, **20**(2), pp. 144–173.

[66] Hata, T., Kobayashi, N., Kimura, F., and Suzuki, H., 2000, "Representation of functional relations among parts and its application to product failure reasoning," International Journal for Manufacturing Science and Production, **3**(2/4), pp. 77–84.

[67] Stone, R. B., Tumer, I. Y., and Stock, M. E., 2006, "Linking product functionality to historical failures to improve failure analysis in design," Research in Engineering Design, **16**(2), pp. 96–108.

[68] Grantham-Lough, K., Stone, R. B., and Tumer, I. Y., 2008, "Implementation Procedures for the Risk in Early Design (RED) Method," Journal of Industrial and Systems Engineering, **2**(2), pp. 126–143.

[69] Krus, D. and Grantham Lough, K., 2007, "Applying function-based failure propagation in conceptual design," *Proceedings of the ASME Design Engineering Technical Conferences; International Design Theory and Methodology Conference*.

[70] Clarkson, P., Simons, C., and Eckert, C., 2004, "Predicting change propagation in complex design," Journal of Mechanical Design, **126**, p. 788.

[71] Jensen, D., Tumer, I. Y., and Kurtoglu, T., 2009, "Flow State Logic (FSL) for analysis of failure propagation in early design," *Proceedings of the ASME Design Engineering Technical Conferences; International Design Theory and Methodology Conference*.

[72] Kurtoglu, T., Tumer, I. Y., and Jensen, D., 2010, "A Functional Failure Reasoning Methodology for Evaluation of Conceptual System Architectures," Research in Engineering Design, **21**(4), pp. 209–234.

[73] Jensen, D., Tumer, I. Y., and Kurtoglu, T., 2008, "Modeling the propagation of failures in software-driven hardware systems to enable risk-informed design," *Proceedings of the ASME International Mechanical Engineering Congress and Exposition*.

[74] Jensen, D., Tumer, I. Y., and Kurtoglu, T., 2009, "Design of an Electrical Power System using a Functional Failure and Flow State Logic Reasoning Methodology," *Proceedings of the Prognostics and Health Management Society Conference*.

[75] Leveson, N., 1995, *Safeware: System Safety and Computers*, Addison-Wesley Publishing Co.

[76] Vesely, W. E., Goldberg, F. F., Roberts, N., and Haasi, D. F., 1981, *The Fault Tree Handbook*, US Nuclear Regulator Commission, NUREG0492, Washington, D.C.

[77] Lee, W., Grosh, D., Tillman, F., and Lie, C., 1985, "Fault tree analysis, methods, and applications a review," Reliability, IEEE Transactions on, **34**(3), pp. 194–203.

[78] Greenfield, M. A., 2000, "Nasas use of quantitative risk assessment for safety upgrades," *IAAA Symposium*.

[79] Stamatelatos, M. and Apostolakis, G., 2002, *Probabilistic Risk Assessment Procedures Guide for NASA Managers and Practitioners v 1.1*, NASA, Safety and Mission Assurance, Washington, D.C.

[80] Mosleh, A., Groen, F., Hu, Y., Zhu, D., Najad, H., and Piers, T., 2007, *Simulation-Based Probabilistic Risk Analysis Report*, Center for Risk and Reliability, University of Maryland.

[81] Rausand, M. and Hoylanc, A., 2004, "System reliability theory: Models and statistical method¿," Annals of Statistics, **6**, pp. 701–726.

[82] Dulac, N. and Leveson, N., 2005, "Incorporating safety into early system architecture trade studies," *Proceedings of the International conference of the system safety society.*

[83] Pereira, S., Lee, G., and Howard, J., 2006, *A System-Theoretic Hazard Analysis Methodology for a Non-advocate Safety Assessment of the Ballistic Missile Defense System*, Citeseer.

[84] Leveson, N., 2011, "Engineering a safer world," MIT Press.

[85] Perrow, C., 1984, *Normal accidents: Living with high-risk technologies*, Princeton University Press.

[86] Morgan, M. and Morrison, M., 1999, *Models as mediators*, Cambridge University Press.

[87] Hodges, W., 1993, *Model theory*, Cambridge Univ Press.

[88] Mukherjee, A. and Samantaray, A., 2001, "System modelling through bond graph objects on SYMBOLS 2000," Simulation Series, **33**(1), pp. 164–170.

[89] Karnopp, D., Margolis, D., and Rosenberg, R., 2006, *System dynamics: modeling and simulation of mechatronic systems*, vol. 3, John Wiley & Sons New Jersey.

[90] Peterson, J., 1981, *Petri net theory and the modeling of systems*, Prentice Hall PTR Upper Saddle River, NJ, USA.

[91] Jensen, K., 1996, *Coloured Petri nets: basic concepts, analysis methods, and practical use*, Springer Verlag.

[92] Menzel, C. and Mayer, R., 1998, "The IDEF family of languages," Handbook on architectures of information systems, pp. 215–249.

[93] Long, J., 2002, "Relationships between common graphical representations in system engineering," Vitech white paper, Vitech Corporation, Vienna, VA.

[94] Russell, S. and Norvig, P., 2003, *Artificial intelligence: a modern approach*, Prentice Hall Englewood Cliffs, NJ.

[95] Sage, A., 1995, "Systems engineering and systems management for reengineering," Journal of Systems and Software, **30**(1), pp. 3–25.

[96] Chen, P., 1976, "The entity-relationship modeltoward a unified view of data," ACM Transactions on Database Systems (TODS), **1**(1), p. 36.

[97] Harel, D., 1987, "Statecharts: A visual formalism for complex systems," Science of computer programming, **8**(3), pp. 231–274.

[98] Taylor, F., 2005, *The principles of scientific management*, 1st World Library.

[99] Larsen, P., Plat, N., and Toetenel, H., 1994, "A formal semantics of data flow diagrams," *Formal aspects of Computing*, Citeseer.

[100] Lano, J., 1990, "The N2 Chart," System and Software Requirements Engineering, pp. 244–271.

[101] Alford, M., 1977, "A requirements engineering methodology for real-time processing requirements," IEEE Transactions on Software Engineering, pp. 60–69.

[102] Long, J., 2000, "Relationships between Common Graphical Representations Used in System Engineering," Proceedings of SETE.

[103] Fowler, M., Scott, K., et al., 1998, *UML distilled*, Addison-Wesley Boston.

[104] Kartson, D., Balbo, G., Donatelli, S., Franceschinis, G., and Conte, G., 1994, *Modelling with generalized stochastic Petri nets*, John Wiley & Sons, Inc. New York, NY, USA.

[105] Object Modeling Group, "Sysml version 1.2," URL `http://www.omgsysml.org/`.

[106] Ahmad, S., 2007, *Analyzing Suitability of SysML for System Engineering Applications*, Master's thesis, Saleem Zubair Ahmad School of Engineering Blekinge Institute of Technology, Box 520 SE 372 25 Ronneby Sweden.

[107] Sasajima, M., Kitamura, Y., Mitsuru, I., and Mizoguchi, R., 1996, "A representation language for behavior and function: Fbrl," Expert Systems with Aplications, **10**(3-4), pp. 471–479.

[108] Qian, L. and Gero, J., 1996, "Function-behaviour-structure and their roles in analogy-based design," Artificial Intelligence in Engineering Design, Analysis and Manufacture, **10**, pp. 289–312.

[109] DeLoach, S., 1999, *Multiagent systems engineering: A methodology and language for designing agent systems*, Storming Media.

[110] Acquisti, A., Sierhuis, M., Clancey, W., and Bradshaw, J., 2002, "Agent based modeling of collaboration and work practices onboard the international space station," *Proceedings of the Eleventh Conference on Computer-Generated Forces and Behavior Representation*, vol. 16.

[111] Chandrasekaran, B. and Josephson, J., 2000, "Function in device representation," Engineering with computers, **16**(3), pp. 162–177.

[112] Umeda, Y., Tomiyama, T., and Yoshikawa, H., 1995, "Fbs modeling: Modeling scheme of function for conceptual design," *Proc. of the 9th Int. Workshop on Qualitative Reasoning*, pp. 271–8.

[113] Pahl, G. and Beitz, W., 1996, *Engineering Design: A Systematic Approach*, Springer-Verlag, London, UK.

[114] Komoto, H., D'amelio, V., Echavarria, E., Tomiyama, T., et al., 2008, "A review of function modeling: Approaches and applications," Artificial Intelligence for Engineering Design, Analysis and Manufacturing, **22**(2), pp. 147–169.

[115] Ullman, D. G., 2003, *The mechanical design process*, McGraw-Hill.

[116] Welch, R. and Dixon, J., 1992, "Representing function, behavior and structure during conceptual design." *4 th International Conference on Design Theory and Methodology*, pp. 11–18.

[117] Deng, Y.-M., Britton, G., and Tor, S., 2000, "Constraint-based functional design verification for conceptual design," CAD Computer Aided Design, **32**(14), pp. 889–899.

[118] De Kleer, J. and Brown, J., 1984, "A qualitative physics based on confluences," Artificial intelligence, **24**(1-3), pp. 7–83.

[119] Stone, R. B. and Wood, K. L., 2000, "Development of a functional basis for design," Journal of Mechanical Design, **122**(4), pp. 359–370.

[120] Keuneke, A. and Allemang, D., 1989, "Exploring the no-function-in-structure principle," Journal of Experimental & Theoretical Artificial Intelligence, **1**(1), pp. 79–89.

[121] Bohm, M., Stone, R., and Szykman, S., 2005, "Enhancing virtual product representations for advanced design repository systems," Journal of Computing and Information Science in Engineering, **5**(5), p. 360.

[122] Bryant, C., Stone, R., Greer, J., McAdams, A., Kurtoglu, T., and Campbell, M. A., 2007, "A function-based component ontology for system design," *Proc. of the International Conerence on Enginering Design.*

[123] Umeda, Y., Takeda, H., Tomiyama, T., and Yoshikawa, H., 1990, "Function, behaviour, and structure," Applications of artificial intelligence in engineering V, **1**, pp. 177–194.

[124] Umeda, Y., Ishii, M., Yoshioka, M., Shimomura, Y., and Tomiyama, T., 1996, "Supporting conceptual design based on the function-behavior-state modeler," Artificial Intelligence for Engineering Design, Analysis and Manufacturing: Aiedam, **10**(4), pp. 275–288.

[125] Umeda, Y. and Tomiyama, T., 1997, "Functional reasoning in design," IEEE expert, **12**(2), pp. 42–48.

[126] Tomiyama, T., Umeda, Y., and Yoshikawa, H., 1993, "A cad for functional design," CIRP Annals-Manufacturing Technology, **42**(1), pp. 143–146.

[127] Yoshioka, M., Umeda, Y., Takeda, H., Shimomura, Y., Nomaguchi, Y., and Tomiyama, T., 2004, "Physical concept ontology for the knowledge intensive engineering framework," Advanced Engineering Informatics, **18**(2), pp. 95–113.

[128] Gero, J., 1990, "Design prototypes: a knowledge representation schema for design," AI magazine, **11**(4), p. 26.

[129] Gero, J. and Kannengiesser, U., 2007, "A function–behavior–structure ontology of processes," AI EDAM: Artificial Intelligence for Engineering Design, Analysis, and Manufacturing, **21**(04), pp. 379–391.

[130] Forbus, K., 1984, "Qualitative process theory," Artificial intelligence, **24**(1-3), pp. 85–168.

[131] Keuneke, A., 1991, "Device representation-the significance of functional knowledge," IEEE expert, **6**(2), pp. 22–25.

[132] Umeda, Y., Tomiyama, T., Yoshikawa, H., and Shimomura, Y., 1994, "Using functional maintenance to improve fault tolerance," IEEE Expert: Intelligent Systems and Their Applications, **9**(3), pp. 25–31.

[133] Hirtz, J., Stone, R., McAdams, D., Szykman, S., and Wood, K., 2002, "A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts," Research in Engineering Design, **13**, pp. 65–82.

[134] Leveson, N., 2004, "A new accident model for engineering safer systems," Safety Science, **42**(4), pp. 237–270.

[135] Hutcheson, R., McAdams, D. A., Stone, R. B., and Tumer, I. Y., 2007, "Function-based systems engineering," *International Conference on Engineering Design*.

[136] Patterson-Hine, A., Narasimhan, S., Aaseng, G., Biswas, G., and Pattipati, K., 2005, "A review of diagnostic techniques for ishm applications," *1st Integrated Systems Health Engineering and Management Forum*.

[137] De Kleer, J. and Kurien, K., 2003, *Fundamentals of model-based diagnosis*, Safe Process.

[138] Qualtech Systems Inc., 2010, "Teams tool," URL `http://www.teamqsi.com/`.

[139] DSI International Inc., 2012, "eXpress," URL `http://www.dsiintl.com/WebLogic/Products.aspx`.

[140] Deb, S., Pattipati, K. R., Raghavan, V., Shakeri, M., and Shrestha, R., 1995, "Multisignal flow graphs: a novel approach for system testability analysis and fault diagnosis," IEEE Aerospace and Electronics Systems Magazine, **10**, pp. 14–25.

[141] Weld, D. and de Kleer, J., 1987, *Readings in qualitative physics*, Morgan Kauffman.

[142] Kuipers, B., 1986, "Qualitative simulation," Artificial intelligence, **29**(3), pp. 289–338.

[143] Struss, P., 1988, "Mathematical Aspects of Qualitative Reasoning," Int. J. Artificial Intelligence in Engineering, **3(3)**, pp. 156–169.

[144] Patton, R., Frank, P., and Clark, R., 1998, *Fault Diagnosis in Dynamic Systems: Theory and Applications*, Prentice Hall, Hertfordshire, UK.

[145] Dvorak, D. and Kuipers, B. J., 1989, "Model based monitoring of dynamic systems," *Proceedings of IJCAI*.

[146] Williams, B. C. and Nayak, P. P., 1996, "A model-based approach to reactive self-configuring systems," AAAI, pp. 971–978.

[147] Kurien, J. and Nayak, P., 2000, "Back to the future with consistency-based trajectory tracking," AAAI, pp. 370–377.

[148] Korbicz, J., Koscielny, J., and Kowalczuk, Z., 2004, *Fault diagnosis: models, artificial intelligence, applications*, Springer Verlag.

[149] Giarratano, J. C. and Riley, G. D., 2004, *Expert Systems: Principles and Programming*, PWS, Boston, MA.

[150] Yairi, T., Kato, Y., and Hori, K., 2001, "Fault Detection by Mining Association Rules from House-keeping Data," *Proceedings of the SAIRAS*.

[151] Berenji, H., Ametha, J., and Vengerov, D., 2003, "Inductive Learning For Fault Diagnosis," *Proceedings of the 12th IEEE International Conference on Fuzzy Systems*, pp. 726–731.

[152] Bedford, T. and Cooke, R., 2001, *Probabilistic Risk Analysis Foundations and Methods*, Cambridge University Press, Cambridge, UK.

[153] Xu, H. and Dugan, J., 2004, "Combining dynamic fault trees and event trees for probabilistic risk assessment," *Reliability and Maintainability, 2004 Annual Symposium-RAMS*, IEEE, pp. 214–219.

[154] Hosseini, S. and Takahashi, M., 2007, "Combining static/dynamic fault trees and event trees using bayesian networks," Computer Safety, Reliability, and Security, pp. 93–99.

[155] Umeda, Y., Tomiyama, T., and Yoshikawa, H., 1992, "A design methodology for a self-maintenance machine based on functional redundancy," *International Conference on Design Theory and Methodology*, Amer Society of Mechanical, p. 317.

[156] Derelöv, M., 2008, "Qualitative modelling of potential failures: on evaluation of conceptual design." Journal of Engineering Design, **19**(3), pp. 201 – 225.

[157] Hutcheson, R., McAdams, D., Stone, R., and Tumer, I. Y., 2006, "FACE A function-based methodology for analyzing critical events," *Proceedings of the ASME Design Engineering Technical Conferences*.

[158] Mehr, A. F. and Tumer, I. Y., 2006, "Risk based decision making for managing resources during the design of complex aerospace systems," Journal of Mechanical Design, **128**(4), pp. 1014–1022.

[159] Hoyle, C., Tumer, I. Y., and A. F. Mehr, a. W. C., 2009, "Health Management Allocation During Conceptual System Design," Journal of Computing and Information Science in Engineering, **9**(2).

[160] Barbacci, M., Clements, P., Lattanze, A., Northrop, L., and Wood, W., 2003, "Using the architecture tradeoff analysis method (atam) to evaluate the software architecture for a product line of avionics systems: A case study," .

[161] Kurtoglu, T., Johnson, S., Barszcz, E., Johnson, J., and Robinson, P., 2008, "Integrating system health management into early design of aerospace systems using functional fault analysis," *Proc. of the International Conference on Prognostics and Heath Management, PHM08.*

[162] Kurtoglu, T., Campbell, M., Bryant, C., Stone, R., and McAdams, D., 2005, "Deriving a component basis for computational functional synthesis," *International Conference on Engineering Design, ICED*, vol. 5.

[163] Kurtoglu, T., Campbell, M., Gonzales, J., Bryant, C., Stone, R., and McAdams, D., 2005, "Capturing empirically derived design knowledge for creating conceptual design configurations," *Proceedings of the ASME Design Engineering Technical Conferences And Computers In Engineering Conference. DETC2005-84405, in review, Long Beach, CA.*

[164] Poll, S., 2007, "Advanced diagnostics and prognostics testbed," *18th International Workshop on Principles of Diagnosis.*

[165] Hutcheson, R. and Tumer, I. Y., 2005, "Function-based design of a spacecraft power subsystem diagnostics testbed," *Proceedings of the ASME International Mechanical Engineering Congress and Exposition.*

[166] Jensen, D., Tumer, I. Y., and Kurtoglu, T., 2012, "Using fault mode dependencies to identify potential fault propagation paths," Journal of Engineering Design, **(In Review)**.

[167] Suh, N., 1990, *The principles of design*, vol. 226, Oxford University Press New York.

[168] Hasenkamp, T., Arvidsson, M., and Gremyr, I., 2009, "A review of practices for robust design methodology." Journal of Engineering Design, **20**(6), pp. 645 – 657.

[169] Padhke, M., 1989, *Quality engineering using robust design*, Prentice Hall, Englewood Cliffs, NJ.

[170] Clausing, D., 1994, *Quality function deployment*, MIT Press, Cambridge, MA.

[171] Tumer, I. Y. and Stone, R. B., 2003, "Mapping Function to Failure during High-Risk Component Development," Research in Engineering Design, **14**, pp. 25–33.

[172] Navlakha, J., 1987, "A survey of system complexity metrics," The Computer Journal, **30**(3), p. 233.

[173] Greer, J., Jensen, D., and Wood, K., 2004, "Effort flow analysis: a methodology for directed product evolution," Design Studies, **25**(2), pp. 193–214.

[174] Otto, K. N. and Wood, K. L., 1999, "Product evolution: A reverse engineering and redesign methodology," Research in Engineering Design, **10**(4), pp. 226–243.

[175] Matthiassen, B., 1997, *Design for Robustness and Reliability: Improving the Quality Consciousness in Engineering Design*, Department of Control and Engineering Design, DTU.

[176] Xing, L., Shrestha, A., Meshkat, L., and Wang, W., 2009, "Incorporating common-cause failures into the modular hierarchical systems analysis," Reliability, IEEE Transactions on, **58**(1), pp. 10–19.

[177] Malone, B. and Papay, M., 1999, "Modelcenter: An integration environment for simulation based design," *Simulation Interoperability Workshop*.

[178] OHalloran, B., Stone, R., and Tumer, I., 2011, "Link between function-flow failure rates and failure modes for early design stage reliability analysis," *International Mechanical Engineering Conference & Exposition*.

[179] Jensen, D., Hoyle, C., and Tumer, I. Y., 2012, "Clustering function-based failure analysis results to evaluate and reduce system-level risks," *ASME IDETC/CIE, Computers and Information in Engineering Conference*.

[180] Papakonstantinou, N., Jensen, D., Sierla, S., and Tumer, I. Y., 2011, "Capturing interactions and emergent failure behavior in complex engineered systems and multiple scales," *Proceedings of the ASME Design Engineering Technical Conferences; Computers in Engineering Conference*.

[181] Sierla, S., Tumer, I., Papakonstantinou, N., Koskinen, K., and Jensen, D., 2012, "Early integration of safety to the mechatronic system design process by the functional failure identication and propagation framework," Mechatronics, p. doi:10.1016/j.mechatronics.2012.01.003.

[182] Tumer, I. and Smidts, C., 2010, "Integrated design and analysis of software-driven hardware systems," IEEE Transactions on Computers, **60**, pp. 1072–1084.

[183] Coatanéa, E., Nonsiri, S., Ritola, T., Tumer, I., and Jensen, D., 2011, "A framework for building dimensionless behavioral models to aid in function-based failure propagation analysis," Journal of Mechanical Design, **133**, p. 121001.

[184] Lazarsfeld, P. F., 1959, *"Latent Structure Analysis" in Psychology: A Study of a Science*, vol. 3, New York: McGraw-Hill.

[185] Pearl, J., 2000, *Causality: models, reasoning and inference*, Cambridge Univ Press.

[186] Vermunt, J. and Magidson, J., 2004, *"Latent Class Analysis" in The Sage encyclopedia of social science research methods*, vol. 1, Sage Publications, Inc.

[187] Linzer, D. A. and Lewis, J., eds., 2011, *poLCA: Polytomous Variable Latent Class Analysis*, R package version 1.3.1. http://userwww.service.emory.edu/ dlinzer/poLCA.

[188] Linzer, D. A. and Lewis, J., 2011, "polca: an r package for polytomous variable latent class analysis," Journal of Statistical Software, **42**(10), pp. 1–29.

[189] R Development Core Team, 2011, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, URL `http://www.R-project.org/`, ISBN 3-900051-07-0.

[190] Magidson, J. and Vermunt, J., 2003, "Comparing latent class factor analysis with the traditional approach in datamining," forthcoming in Statistical Applications of Datamining.

[191] Denson, W., 1992, "Reliability assessment of critical electronic components," Tech. rep., DTIC Document.

[192] Thramboulidis, K., 2005, "Model-integrated mechatronics-toward a new paradigm in the development of manufacturing systems," Industrial Informatics, IEEE Transactions on, **1**(1), pp. 54–61.

[193] Van Amerongen, J., 2003, "Mechatronic design," Mechatronics, **13**(10), pp. 1045–1066.

[194] Price, C. and Taylor, N., 1998, "FMEA for multiple failures," *Reliability and Maintainability Symposium, 1998. Proceedings., Annual*, IEEE, pp. 43–47.

[195] Hu, T., Yu, J., and Wang, S., 2009, "Research on complex system fmea method based on functional modeling," *Reliability, Maintainability and Safety, 2009. ICRMS 2009. 8th International Conference on*, IEEE, pp. 63–66.

[196] Mauri, G., McDermid, J., and Papadopoulos, Y., 1998, "Extension of hazard and safety analysis techniques to address problems of hierarchical scale," *Systems Engineering of Aerospace Projects (Digest No. 1998/249), IEE Colloquium on*, IET, pp. 4–1.

[197] Papadopoulos, Y., Parker, D., and Grante, C., 2004, "Automating the failure modes and effects analysis of safety critical systems," *High Assurance Systems Engineering, 2004. Proceedings. Eighth IEEE International Symposium on*, Ieee, pp. 310–311.

[198] Pasquale, T., Rosaria, E., Pietro, M., Antonio, O., and Segnalamento Ferroviario, A., 2003, "Hazard analysis of complex distributed railway systems," *Reliable Distributed Systems, 2003. Proceedings. 22nd International Symposium on*, IEEE, pp. 283–292.

[199] Schreiber, S., Schmidberger, T., Fay, A., May, J., Drewes, J., and Schnieder, E., 2007, "Uml-based safety analysis of distributed automation systems," *Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on*, IEEE, pp. 1069–1075.

[200] Perera, J. and Holsomback, J., 2004, "Use of probabilistic risk assessments for the space station program," *Aerospace Conference, 2004. Proceedings. 2004 IEEE*, vol. 1, IEEE.

[201] Sturges, R., O'Shaughnessy, K., and Kilani, M., 1996, "Computational model for conceptual design based on extended function logic," Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM, **10**(4), pp. 255–274.

[202] Szykman, S., Sriram, R., Bochenek, C., and Racz, J., 1998, "The nist design repository project," Advances in Soft Computing-Engineering Design and Manufacturing, pp. 5–19.

[203] Terpenny, J. and Mathew, D., 2004, "Modeling environment for function-based conceptual design," *ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*.

[204] Huang, E., Ramamurthy, R., and McGinnis, L., 2007, "System and simulation modeling using sysml," *Proceedings of the 39th conference on Winter simulation: 40 years! The best is yet to come*, IEEE Press, pp. 796–803.

[205] Simpson, T., Poplinski, J., Koch, P., and Allen, J., 2001, "Metamodels for computer-based engineering design: survey and recommendations," Engineering with computers, **17**(2), pp. 129–150.

[206] Box, G. and Wilson, K., 1951, "On the experimental attainment of optimum conditions," Journal of the Royal Statistical Society. Series B (Methodological), **13**(1), pp. 1–45.

[207] Guo, J. and Du, X., 2010, "Reliability analysis for multidisciplinary systems with random and interval variables," AIAA journal, **48**(1), pp. 82–91.

[208] Zang, C., Friswell, M., and Mottershead, J., 2005, "A review of robust optimal design and its application in dynamics," Computers & structures, **83**(4), pp. 315–326.

[209] de Kleer, J., Kuhn, L., Liu, J., Price, B., Do, M., and Zhou, R., 2009, "Continuously estimating persistent and intermittent failure probabilities," Proc. SAFE Process.

[210] Abagyan, L., Golubev, V., Golyaev, N., Zvonarev, A., Koleganov, Y., Nikolaev, M., and Orlov, M., 1968, "Propagation of neutrons in uranium dioxide ii. doppler effect in u 238," Atomic Energy, **25**(4), pp. 1090–1094.

[211] Davis, M., Sigal, R., and Weyuker, E., 1994, *Computability, complexity, and languages: fundamentals of theoretical computer science*, Morgan Kaufmann.

[212] Gaffney, J. and Davis, C., 1988, "An approach to estimating software errors and availability," *Proceedings of the 11th Minnowbrook Workshop on Software Reliability July*.

[213] Conklin, J., 2006, *Wicked problems & social complexity*, CogNexus Institute.

APPENDICES

Appendix A: Subsystem and Component Models

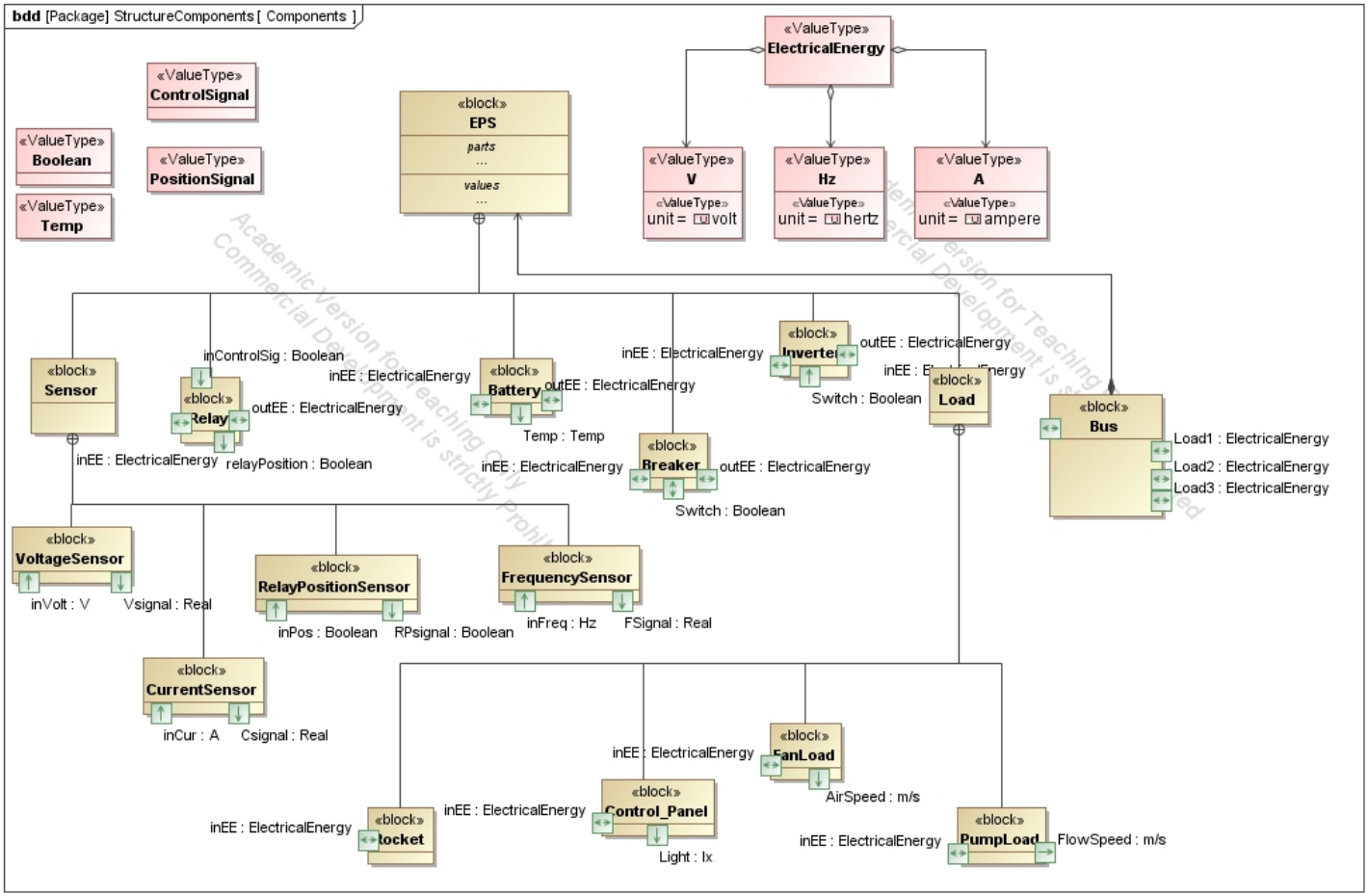


Figure A.1: A SysML block definition diagram illustrating the decomposition of the electrical power subsystem.

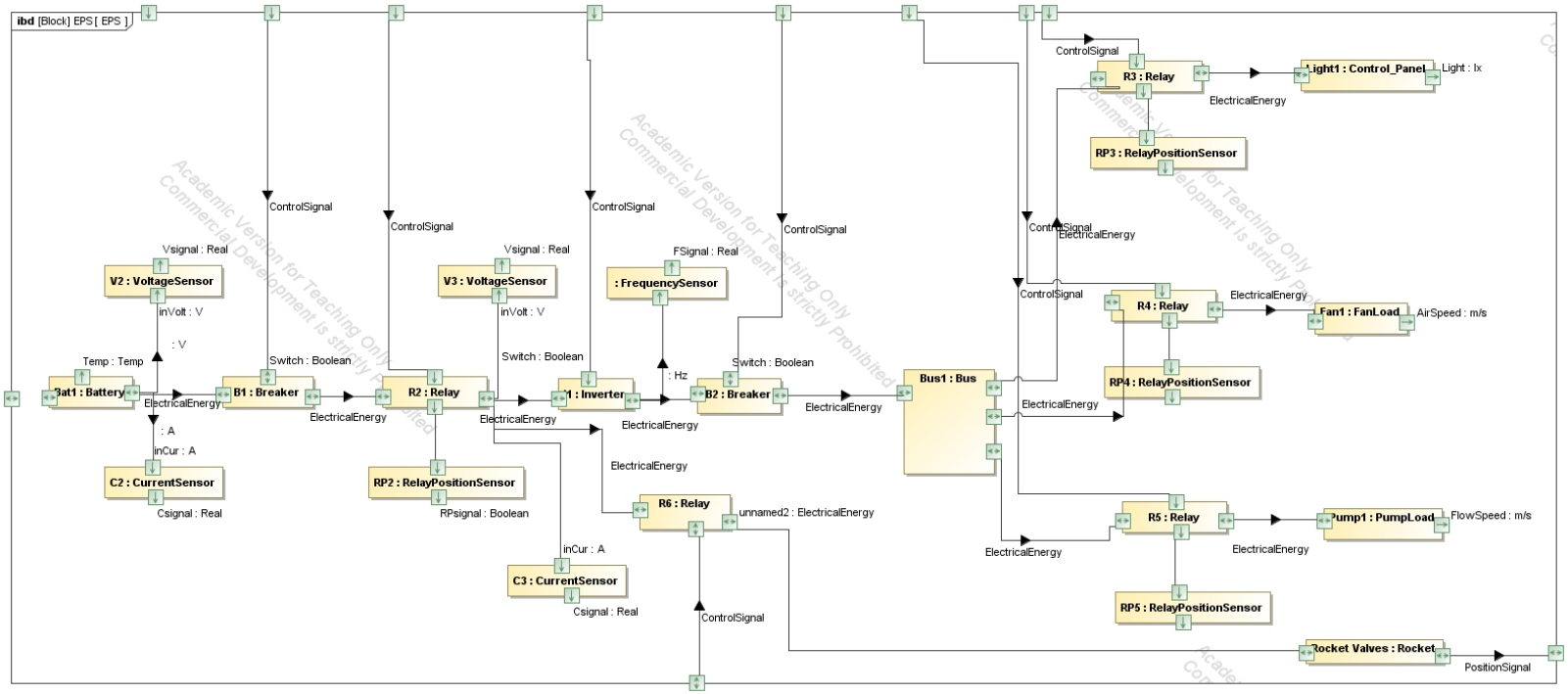


Figure A.2: A SysML internal block diagram illustrating the connections (structure) of the electrical power subsystem.

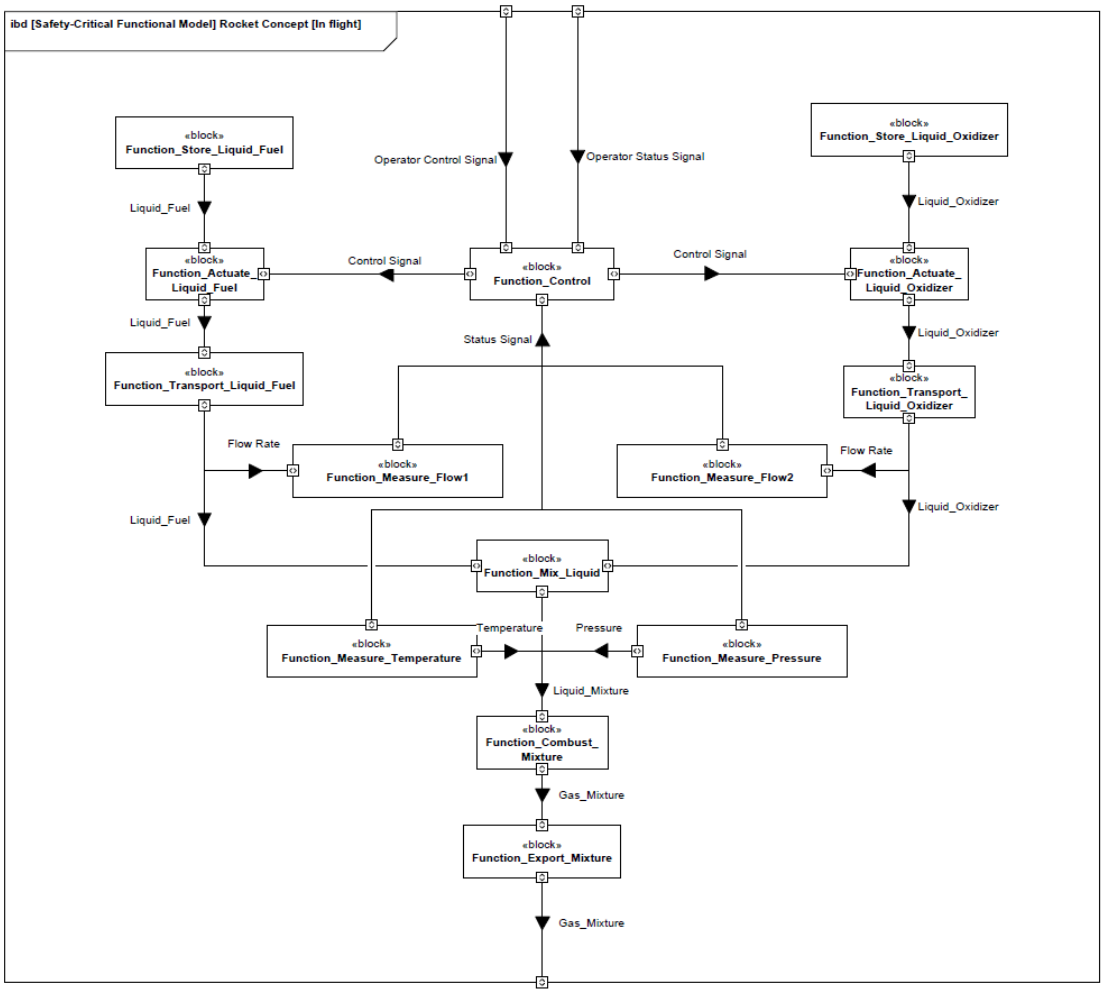


Figure A.3: A SysML internal block diagram illustrating the functional structure of the rocket subsystem.

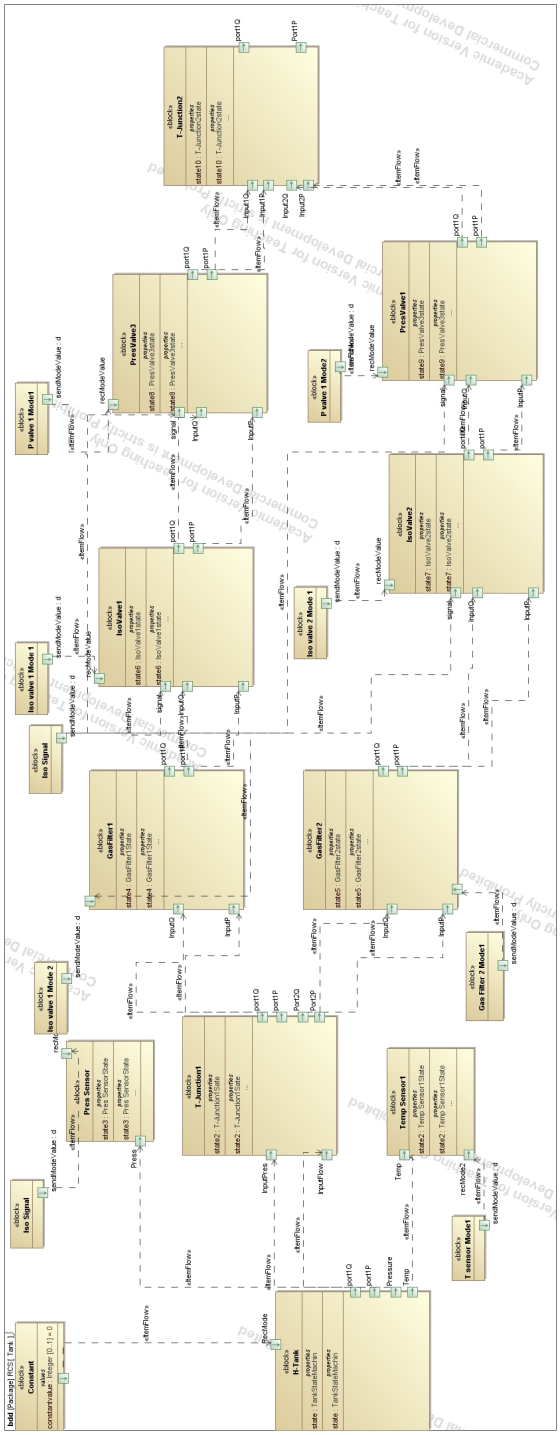


Figure A.4: A SysML internal block diagram illustrating the connections of components in the design of the rocket subsystem.

Appendix B: EPS Design Alternatives

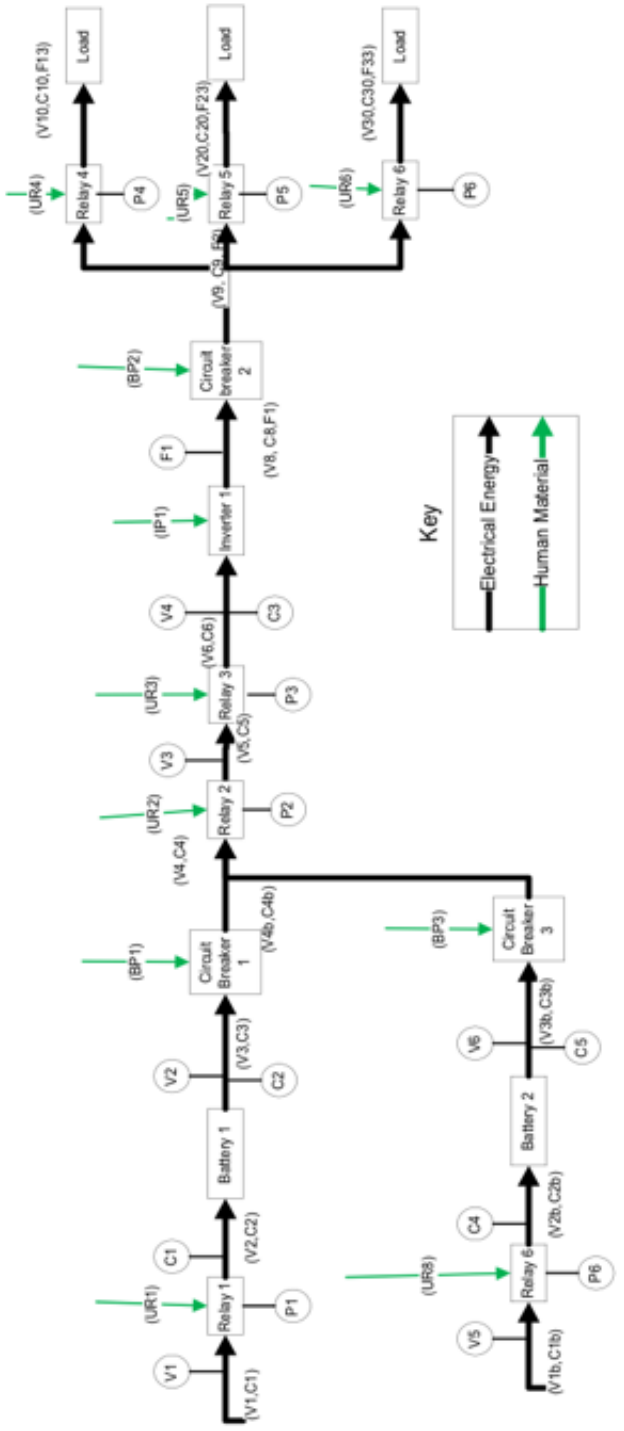


Figure B.1: Configuration Flow Graph for Electrical Powers System Alternative Design #1.

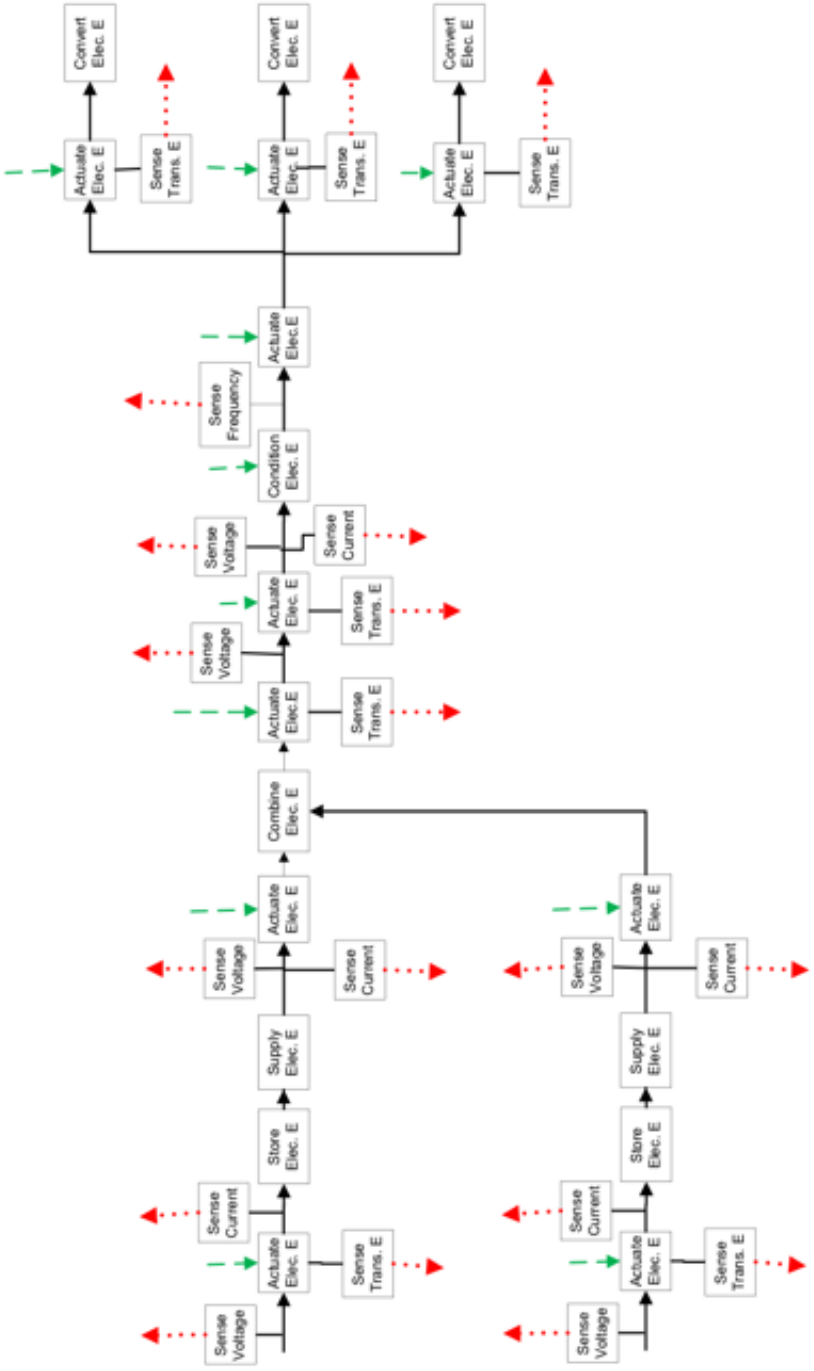


Figure B.2: Low-level Functional Model for Electrical Powers System Alternative Design #1.

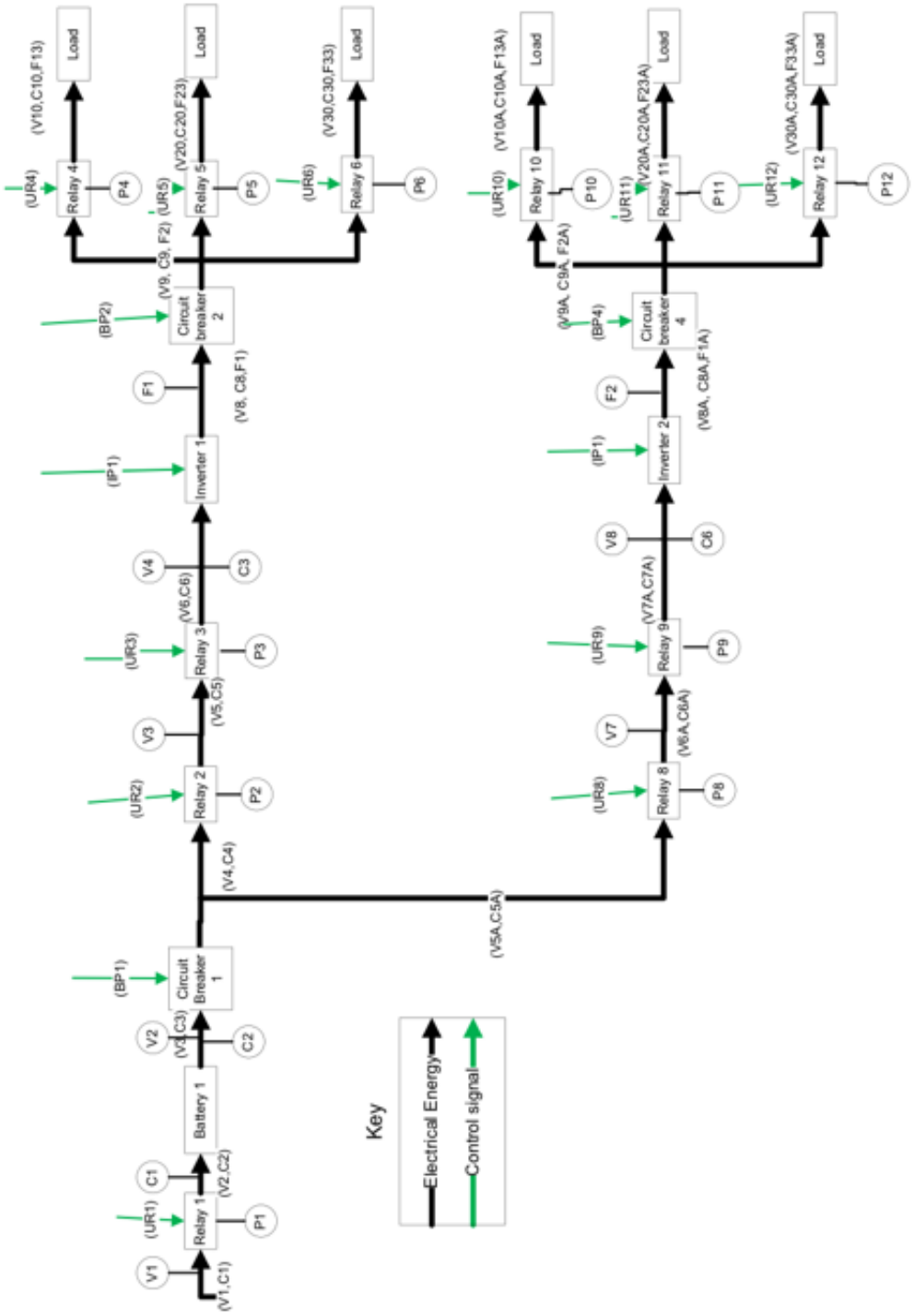


Figure B.3: Configuration Flow Graph for Electrical Powers System Alternative Design #2.

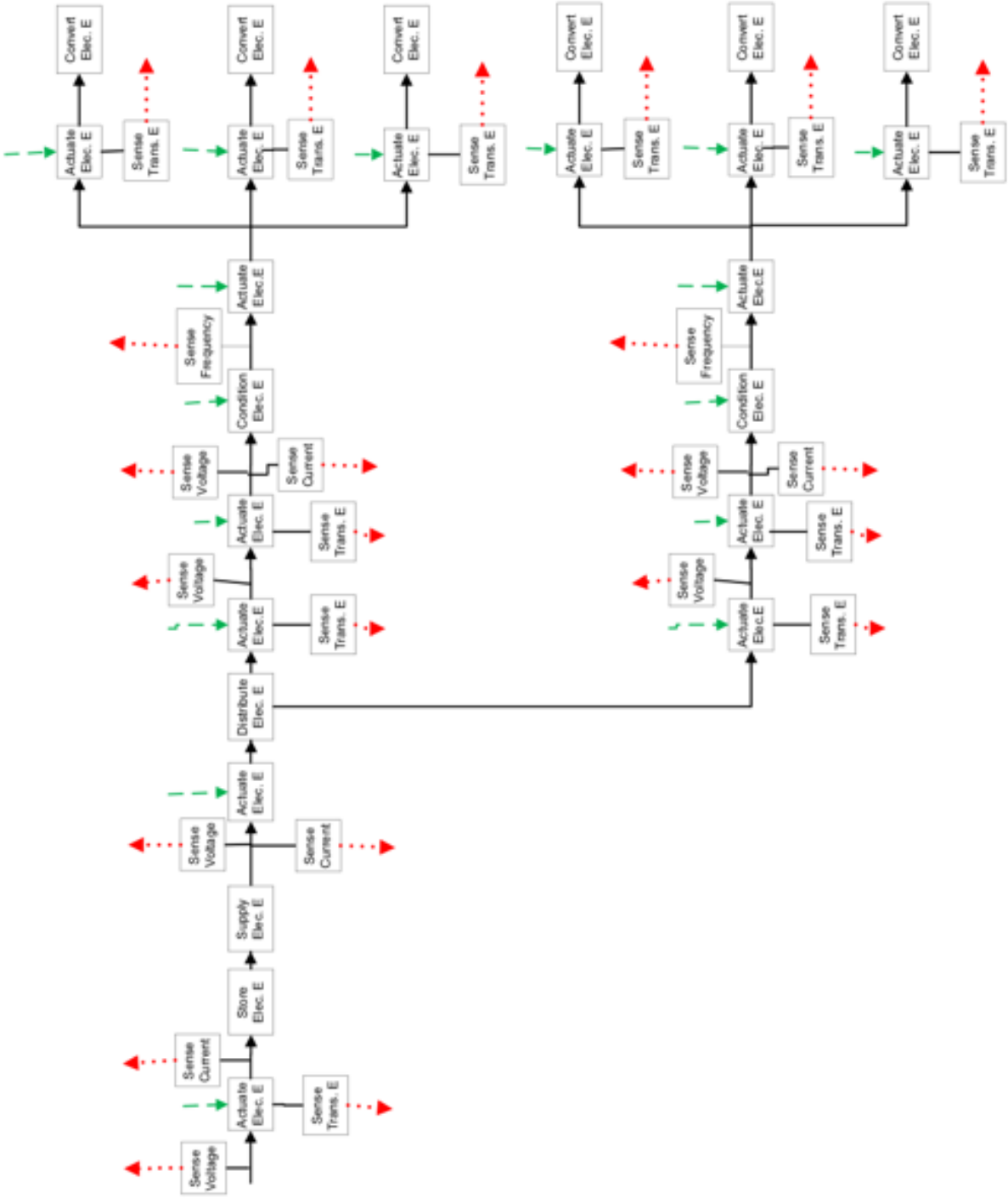


Figure B.4: Low-level Functional Model for Electrical Powers System Alternative Design #2.

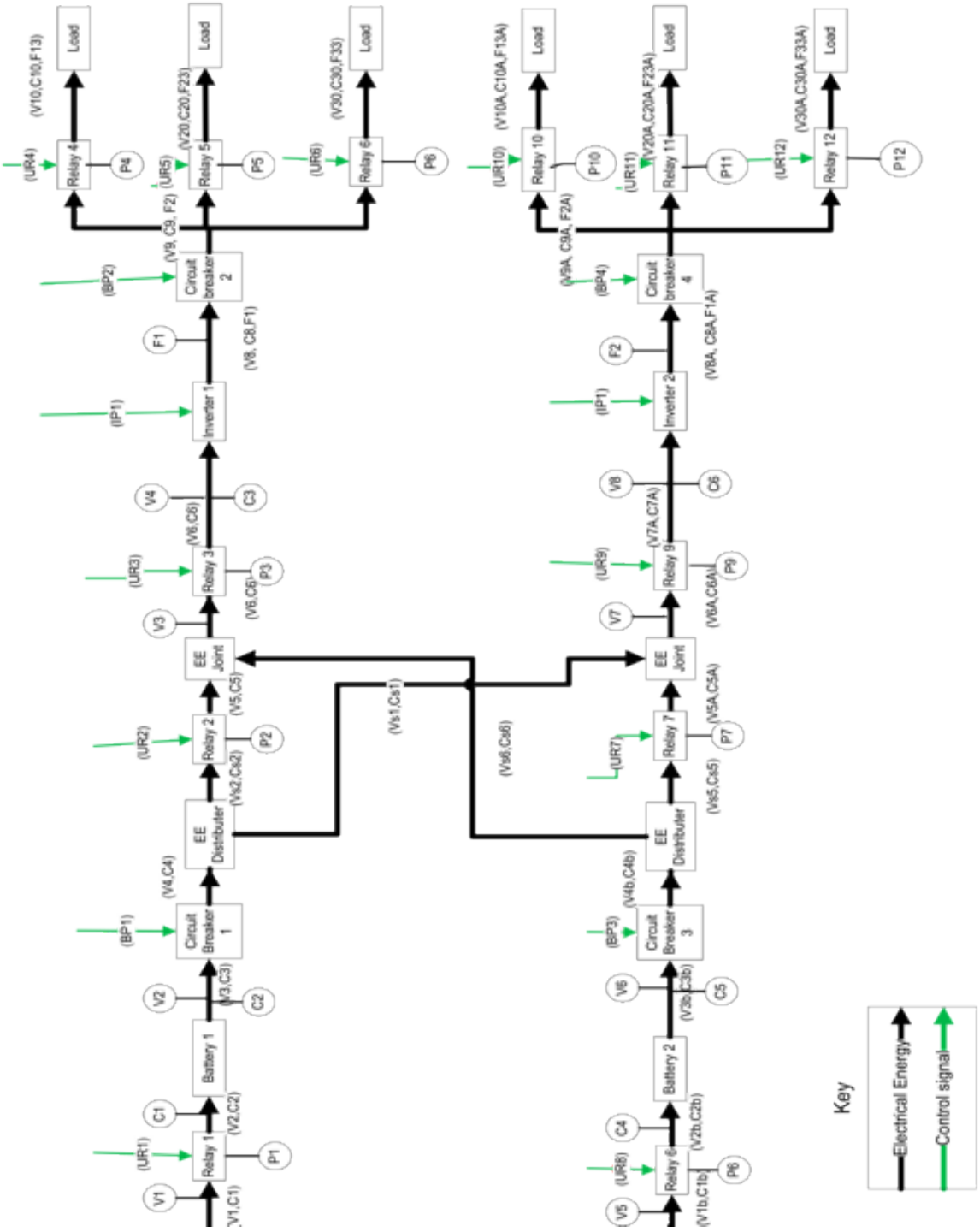


Figure B.5: Configuration Flow Graph for Electrical Powers System Alternative Design #3.

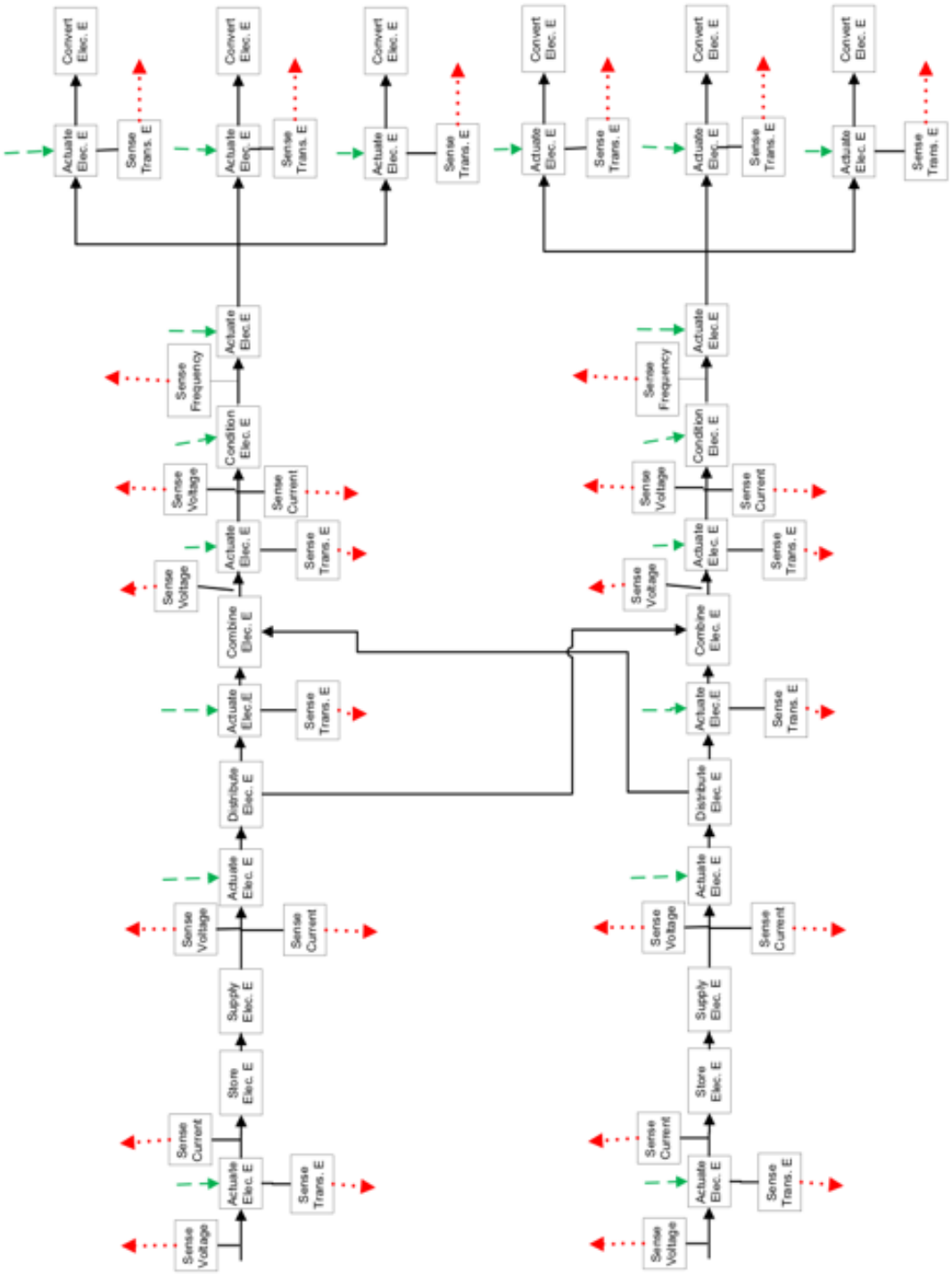


Figure B.6: Low-level Functional Model for Electrical Powers System Alternative Design #3.