Developing Methods of Obtaining Quality Failure Information from Complex Systems

Oladapo Olalekan Bello
University of Arkansas, Fayetteville

Developing Methods of Obtaining Quality Failure Information from Complex Systems

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Engineering

by

Oladapo Olalekan Bello
Obafemi Awolowo University
Bachelor of Science in Agricultural Engineering, 2008
University of Manchester
Master of Science in Mechanical Engineering Design, 2010

August 2017
University of Arkansas

This dissertation is approved for recommendation to the Graduate Council.

ABSTRACT

The complexity in most engineering systems is constantly growing due to ever-increasing technological advancements. This result in a corresponding need for methods that adequately account for the reliability of such systems based on failure information from components that make up these systems.

This dissertation presents an approach to validating qualitative function failure results from model abstraction details. The impact of the level of detail available to a system designer during conceptual stages of design is considered for failure space exploration in a complex system. Specifically, the study develops an efficient approach towards detailed function and behavior modeling required for complex system analyses. In addition, a comprehensive research and documentation of existing function failure analysis methodologies is also synthesized into identified structural groupings.

Using simulations, known governing equations are evaluated for components and system models to study responses to faults by accounting for detailed failure scenarios, component behaviors, fault propagation paths, and overall system performance. The components were simulated at nominal states and varying degrees of fault representing actual modes of operation. Information on product design and provisions on expected working conditions of components were used in the simulations to address normally overlooked areas during installation. The results of system model simulations were investigated using clustering analysis to develop an efficient grouping method and measure of confidence for the obtained results.

The intellectual merit of this work is the use of a simulation based approach in studying how generated failure scenarios reveal component fault interactions leading to a better understanding of fault propagation within design models. The information from using varying fidelity models for system analysis help in identifying models that are sufficient enough at the

conceptual design stages to highlight potential faults. This will reduce resources such as cost, manpower and time spent during system design. A broader impact of the project is to help design engineers identifying critical components, quantifying risks associated with using particular components in their prototypes early in the design process and help improving fault tolerant system designs. This research looks to eventually establishing a baseline for validating and comparing theories of complex systems analysis.

ACKNOWLEDGEMENTS

I would like to appreciate the guidance and support of my advisor, Dr. David C. Jensen who made my research work and journey over the recent years a success. I also have to mention my appreciation to the members of my dissertation committee, Dr. Darin Nutter, Dr. Wenchao Zhou, Dr. Harry Pierson and Dr. Uchechukwu Wejinya. Thank you for your invaluable assistance in making this research possible. Special thanks to Dr. Irem Tumer and Dr. Christopher Hoyle. I also acknowledge my lab mates in the Complex Engineered Adaptive Systems Research Laboratory especially Charlie for the ideas.

Of great importance is my sincere appreciation to my parents, Benjamin and Roseline Bello, my siblings, Tope, Tolu and Gbolahan, thank you for encouraging me and truly being a source of assistance from the beginning. I would also like to specially appreciate a big brother, Yele for his immense contribution towards me achieving this level of success.

I am especially very grateful for the support and sacrifice of my family. I really want to thank my wife, Seun and my son, Dara for being good to me. I am grateful for their love, prayers and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF PUBLISHED PAPERS

Chapter 3	Bello, O., and Jensen, D. (2017). FFIP methodology applications in conceptual design research. Journal of Mechanical Design (in press)
Chapter 4	Bello, O., Jensen, D., Hunter, S., Tumer, I. Y., Hoyle, C. 2016, The Impact of model detail and abstraction on system modeling. Journal of Mechanical Design (in press)
Chapter 5	Some contents of Chapter 5 were extracted and modified from the co-authored research paper detailed below. An additional content using the described methodology was applied to the results of the system model from Chapter 4. Jensen, D. C., Bello, O., Hoyle, C., and Tumer, I. Y. (2014). Reasoning about system-level failure behavior from large sets of function-based simulations. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 28(04), 385-398

CHAPTER 1

INTRODUCTION

This work discusses the successful advancements made in early design stage failure analysis using the Function Failure Identification and Propagation (FFIP) framework. The goal of this work is to explore ways by which the model representation can be used to successfully characterize behavioral and functional abstractions and their effect on design-stage predictions of functional analysis methods. Specifically, the approach in this work identifies critical components in system designs using different levels of model detail and fidelity. A clustering analysis research on FFIP results is also presented.

Engineering systems such as vehicles, airplanes, drones, and most industrial systems are defined as collections of physical entities that comprise a component, device or subsystem [1]. Most of these systems generally consist of interacting mechanical and electrical components with nontrivial dynamic behavior and complicated interaction topology [2].These systems generally fail at some point in their lifecycle. This makes studies into failure analysis of complex system an important area of research to design engineers.

Predicting and mitigating faults in engineering systems continues to be more complex due to the system complexities from increase in demand for more system operation functionalities. Different innovations are constantly being added to already complex systems to make them more robust in their applications. This however, increases the difficulty in studying the relationship between the components and subsystems of these systems. Many systems rarely have components operating independently such that failures in any part will affect neighboring parts. These systems have connecting links in function or behavior among neighboring parts causing a significant change in performance due to failure in one or more constituting components transferring its effect to the next connecting part. Hence, as the level of interactions between components continue to increase, studies on fault identification and propagation, based on failure scenario data within complex systems becomes very important for the purpose of building safer and more reliable designs [3].

Depending on the amount and type of connections to any component, failures in such components are usually exhibited in diverse forms, making it difficult to predict their performance. Some components usually have typical ways of failing throughout their span of operation. For example, typical valve failure modes are often failure to open, failure to close, or leakage. The results of years of operation of engineering systems have provided evidence data on components and systems failing with different characteristic behaviors. Implementing this historical failure information to develop improved methods for failure analysis is important in identifying and eliminating possible failure modes in order to design highly reliable products [4].

Despite the ever-increasing complexities within systems, conventional failure analysis tests such as failure mode, effects analysis (FMECA) [5], fault tree analysis FTA[6] event tree analysis [7] and root cause analysis [8] are still being used in industry. The methods adopt a quantitative approach for analyzing system faults. Most of these comprehensive quantitative methods are mostly applied at later phases of the design process where information such as failure probabilities and detailed failure modes can be established [9, 10]. They are usually not cost or time effective approaches particularly with the possibility of required redesigns when faults are detected. The usefulness of these techniques have also come with some significant limitations such as the need for expert user knowledge, their application early in design stages, their difficulty in identifying potential hazards when unexpected activities linked to unexpected faults occur and the deficiency of capturing multiple interacting faults [11-14].

Furthermore, due to the increase in the scrutiny that large-scale projects experience and the societal and program risks associated with catastrophic failures, risk and safety have become increasingly critical performance parameters for many systems. As a result, there have been a few model-based investigative failure analysis techniques implementing these parameters while addressing the limitations of the conventional tests at early conceptual stages of the design process of complex systems. Some of these methods adopt either a quantitative or a qualitative analysis with very few implementing both.

Qualitative analysis methods are practiced based on function-failure methods. These methods concentrate on the function-based performance which a component has on an entire system. This performance is based on the nominal, degraded or other faulty states that can be characterized as representing the components physical state while in operation. The advantage of utilizing these methods is its application at the conceptual stage of the design process where little information is known about the system [10]. They help by decreasing redesign, time, cost while improving quality and safety of systems [15].

The improvements in modeling and simulation tool packages have boosted the use of qualitative means in carrying out failure analysis. It is crucial for design engineers to be able to reason at the functional level. This aids in identifying system functions that are likely to fail and what the overall effect of the loss of these functions will be on system behavior and performance. Qualitative methods often apply function modeling knowledge in order to utilize function failure methods [5]. These methods focus on whether an individual components function-based performance is having the desired effect on the overall system, or if it is causing a degraded, or even totally defective, system performance state. The key effectiveness of these methods is that they can be implemented at concept level or later in design stages [10].

Using model-based systems design to perform a combination of quantitative and qualitative analysis offers a means of rapid evaluation and redesign with the overall goal of reducing risk of failures, design time, cost and effort. For example, in the early stages of a system design, using modeling tools, emphasis can be made on a system's functional requirements to identify what functions are likely to fail, degrade or impact the overall system behavior and performance before the physical prototype of the system is built. Inserting such analyses into the early design process allows systems engineers to make informed, robust design decisions prior to the allocation of resources. The Function Failure Identification and Propagation (FFIP) framework tool has been developed to support this type of work [16, 17].

Based on available information at early design stages, a sufficient model that adequately represents the various stages of operation of a system with or without faults can be created to explore its failure space. Increase in the scrutiny that large-scale projects experience and the societal and program risks associated with catastrophic failures, risk and safety have become increasingly critical performance parameters for many systems.

This research work uses an approach that implements a method to automatically input failure information obtained from conventional failure analysis tool database into a behavior-based system model. Using the model-based approach, a function and component based structure of the system was built following a real-life system design process. The technique assesses the system model and introduces potential component faults from the list in the failure information database, which is simulated to generate a list of possible failure scenarios. This work combines the effective strengths of conventional and qualitative failure analysis tools to identify what failure scenarios can or should be simulated for any type of engineering system at early design phases.

1.2 Terminology

Due to the need to use terms that are found and defined differently in multiple disciplines the following definitions are intended for this research work.

- Component: Any physical, software, or human element in a system that has nominal and failure behavior.
- Fault/Failure Mode: A discrete behavior of a component different from the nominal behaviors.
- Fault Scenario: The set of nominal and faulty component modes provided to a system simulation.
- Flow: The energy, material and signal that connects functions of a system.
- Function: The action a designer intends in a system that affects the flow of material, energy, or signal.
- Function Health State: The evaluation of the relationship between a component behavior and it’s intend function. With the following categories:
  - Healthy/Nominal: Function acts on flow as intended.
  - Degraded: Function acts on flow but not as intended.
  - Lost: Function does not act on flow.
  - No Flow: There is no ow on which the function could act. (A type of Lost)

    - Overacting: There is too much output of flow on which a function acts.

- System State: The set of health states for all functions resulting from the simulation of function health states.

1.3 Dissertation Outline

The outline of this research is highlighted as follows:

Chapter one gives an introduction of what the research is all about, the goals, the justification or broader impact of the research;

Chapter two gives a general background on what complex systems are, the challenges faced in complex system analyses, failure analysis tools (conventional /traditional) and the model based simulation approaches, as well as the type of result analysis being carried out;

Chapter Three gives a review of existing FFIP–related researches and its state of the art. The main contribution of this chapter is a novel grouping of the different function-based failure analysis methodologies. This chapter starts with a discussion on what functional modeling and what FFIP is all about;

Chapter Four presents the impact model details and fidelity abstractions have on system analyses particularly at the conceptual stages of such systems. Here, a detailed guide into how we propose failure analyses of a selected sample system (the Electric vehicle) was performed and the different results obtained are presented;

Chapter Five is a presentation of a co-authored journal article, “Reasoning about System-Level Failure Behavior from Large Sets of Function-Based Simulations”. This chapter focuses on the approach adopted in analyzing how function failure results are clustered before decisions can be made;

Chapter Six is the concluding chapter which details the logical findings obtained from the results of work carried out.

References

[1] Gao, J., Li, G., and Gao, Z., 2008, "Fault propagation analysis for complex system based on small-world network model," Reliability and Maintainability Symposium, 2008. RAMS 2008. Annual, IEEE, pp. 359-364.

[2] Newman, D. E., Nkei, B., Carreras, B. A., 2005, "Risk assessment in complex interacting infrastructure systems," System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on, IEEE, pp. 63c-63c.

[3] Kmenta, S., and Ishii, K., 2000, "Scenario-based FMEA: a life cycle cost perspective," Proc. ASME Design Engineering Technical Conf. Baltimore, MD, .

[4] Vesely, W.E., Goldberg, F.F., Roberts, N.H., 1981, "The Fault Tree Handbook," US Nuclear Regulatory Commission, NUREG0492, Washington, D.C.

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

[6] Vesely, W., Goldberg, F., Roberts, N., 1981, "Fualt Tree Handbook," Nuclear Regulatory Commission,

[7] Papazoglou, I. A., 1998, "Mathematical Foundations of Event Trees," Reliability Engineering & System Safety, 61(3) pp. 169-183.

[8] Mobley, R.K., 1999, "Root cause failure analysis," Butterworth-Heinemann.

[9] N., T., 2004, "Failure Mode Effects Analysis (FMEA)," ASQ Quality Press, pp. 236-240.

[10] DeStefano, C., and Jensen, D., 2014, "A Qualitative Failure Analysis Using Function-Based Performance State-Machines for Fault Identifification and Propagation During Early Design Phases," Proceedings of the ASME International Design Engineering Technical Conferences; IDETC/CIE}, .

[11] Hawkins, P. G., and Woollons, D. J., 1998, "Failure Modes and Effects Analysis of Complex Engineering Systems using Functional Models," Artificial Intelligence in Engineering, 12(4) pp. 375-397.

[12] Tumer, I. Y., and Stone, R. B., 2003, "Mapping Function to Failure Mode during Component Development," Research in Engineering Design, 14(1) pp. 25-33.

[13] RIPLOVÁ, K., 2007, "Tool of Risk Management: Failure Mode and Effects Analysis and Failure Modes, Effects and Criticality Analysis".

[14] Haider, A. A., and Nadeem, A., 2013, "A Survey of Safety Analysis Techniques for Safety Critical Systems," International Journal of Future Computer and Communication, 2(2) pp. 134.

[15] 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}, .

[16] Kurtoglu, T., and Tumer, I. Y., 2008, "A Graph-Based Fault Identification and Propagation Framework for Functional Design of Complex Systems," 130(5).

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

CHAPTER 2

RESEARCH BACKGROUND

This chapter covers related research for this dissertation. First, general background knowledge is presented. Additional sections are then used to cover specific research as they relate to the following chapters. Finally, other research topics that are covered provide good knowledge for reliability engineers to consider in early design.

2.1 Complex Systems

Complex systems are generally not defined by their physical size but are systems that consist of complicated or multiple interacting mechanical and electrical components with nontrivial dynamic behavior [1]. As all systems fail at some point regardless of their level of complexity, study how to mitigate failure in complex systems is a key focus of this research. Failure analysis is important in identifying and eliminating possible failure modes in order to design highly reliable products [2].

Conventional data analysis tests such as failure mode, effects analysis (FMECA) [3-8], fault tree analysis FTA [9-11] event tree analysis [12-15], reliability block diagram [8, 16-18], probabilistic risk assessment [19-25] and root cause analysis [26-28] are commonly used in industries such as the automobile, aviation and aerospace. These tools are useful in troubleshooting and carrying out maintenance works in many modern day systems and operations. However, the usage of these techniques have resulted in certain limitations such as the need for experts, limitation in conceptual design, failure to predict hazards due to emergent behaviors and the inability to capture multiple interacting faults [7, 29-31]. New and improved ways that adequately predict failure scenarios in complex systems from early stage of the design process are currently been encouraged.

Complexity in a system arises from the level of interconnectivity among the components within that system. Most components within a system usually have typical ways of failing individually through the span of their operation. For example, typical valve failure modes are often failure to open, failure to close, or leakage. The application in complex systems may vary but each component within them exhibits similar patterns when they fail. These systems generally have their components sub-divided into certain networks [2, 32, 33], which are in interaction with one another through hardware linkages (power cables, screws, bolts or even pipes). These interactions do not only imply functional partitions but also the potential failure functions.

Systems rarely have components operating independently such that failures in any part will affect neighboring parts. Instead, most systems have connecting links in function or behavior among neighboring parts and would have a significant change in performance due to failure in one or more of its constituting components transferring its effect to the next connecting part. Hence, as the level of interactions between components continues to increase, the study of fault propagation paths and component behavioral states, based on compiling data on failure scenarios within complex systems, then becomes very important for the purpose of building safer and more reliable designs [34, 35].

2.2 Concept of Modeling Faults

Some component failures are often initiated from the effect of faults in other components. Commonly used conventional failure tests usually obtain their information on performance from data sources on the life-cycles of components in order to study a component's failure modes. These tools then require the combination of engineering knowledge, experience and troubleshooting skills to detect component-level and system-level faults. Some recent approaches developed and adapted the use of system models to carry out similar tests through simulations that capture a close to exact operation of the system being observed [17].

The conventional practices, still in use by NASA, DARPA, Boeing, Airbus, and Toyota [4, 36], are more generic in their method of application, while model-based failure tests are intended to be specific to particular systems under consideration. Through the advancement of design methodologies and simulation techniques, reliability is now being early in the design stages, even as early as in the conceptual design stage [37-39]. Much of these new model based tests have revolutionized design analysis especially in terms of the way product performance and reliability tests are now being carried out [40]. There has been extensive work in the modeling of different systems. However, due to the intrinsic complexities involved, modeling of the interaction between these systems has been limited [41, 42].

Today, the use of modeling techniques in design engineering has become an efficient and cost-effective way of representing real-world systems together with or without their working environment. Models generally represent vital aspects of a system, including underlying requirements, the components, existing sub-systems, and how those components and sub-systems communicate with one another [43-46]. These models can then be simulated to enable designers to test designs before system hardware becomes available, or to test conditions that are either difficult or too expensive to replicate in the real world. During the design process, iterating between modeling and simulation will improve the quality of most systems early enough to significantly reduce the number of errors that will be discovered later [47, 48].

Simulation tests can be difficult and time-consuming, and when different tools are used for individual domains, obtaining a system-level view of the design can be challenging. Hence, defects that could have been revealed during the modeling and simulation phase may often be seen during the implementation phase, when defects are more expensive to fix. Many leading modeling and simulation platforms have however been designed to address this issues; examples are the MATLAB-Simulink [48], Wolfram SystemModeler [49], and Phoenix Integration [50].

Simulink, for example, supports not only multi-domain modeling but also simulation, in addition to its own set of ordinary differential equation (ODE) solvers. A vital advantage of using this platform is its ability to represent different domains, including control systems, state machines, and environmental models, in a single model, and then run its simulations to verify that the model is built correctly. The simulation analysis capabilities available here include data displays, state animation, and conditional breakpoints. The completed simulation results of logged data can then be analyzed using MATLAB scripts and visualization tools [51].

2.3 Related Research

The current approach in diagnostics and fault management has been on diagnostic reasoning to mitigate faults when they happen, based on matching data to models during operations [52]. Improved researches on functional design are currently being done into uncovering many hidden fault scenarios which have previously not been easily captured at conceptual design phases. Design engineers now work on providing solutions with high confidence levels in their design analysis (during early stages) to predict failure scenarios resulting from faulty components and how these propagate throughout the system.

Many types of risk analyses are used to study the lifecycle of complex systems [29, 53-57] which includes quantitative and probabilistic methods [58], reliability analysis techniques applied to design [3, 9], or knowledge-based approaches such as lessons learned databases and hazard analyses [59]. However, a critical flaw of these methods is the difficulty in applying them at early design stages where the models are vague, the knowledge, decisions and probabilities about the system are difficult to capture, and hard to designate [60]. Also studies and design reviews have shown that early design stages are one of the best times to catch potential failures and anomalies [61]. This stage is crucial as many design decisions and tasks are still open such as sensor and measurement point selection, safeguards, redundancies, diagnosis, signature and data fusion schemes. These decisions are made to effectively reduce the cost of risk mitigation efforts and increase the safety of designed systems [21, 62, 63].

There is a lack of formal representations and methods for enabling risk analysis at early design stages. Commonly used risk analysis techniques (FMEA, FMECA, FTA) require very detailed, high fidelity models of system components in order to study faulty system behavior and its consequences. For example, Failure Modes and Effects Analysis (FMEA) [64] is a method that systematically examines individual system components and their failure mode characteristics to assess risk and reliability. However, the analysis requires a detailed level of system design, and thus is not optimal to be used during conceptual design [65].

Some researchers have modified existing conventional techniques into improved ones to capture failure scenarios in systems in order to address the limitation of multiple faults capture [66]. Pickard et al [67], proposed the mFMEA (multiple Failure Mode and Effects Analysis), which is an integration of the FMEA method (Failure Mode and Effects Analysis) and the FTA method (Fault Tree Analysis) that provides an inclusive reliability analysis of complex, mechatronic systems. This approach was done by using risk analysis, risk assessment and measure controlling, paralleling systematically to the product design cycle (applicable for single failures). Applying the information from the FMEA, allows the expansion of the FTA through a failure analysis with the help of its combination option to network failures according to Boolean logic. This enables an approach that allows for the consideration of multiple failures while retaining all the characteristics of FMEA and also uses failure networks quantitative information in deriving system's availability along with the results of the mFMEA.

Other researchers perform risk analysis on the risk priority numbers (RPN) obtained from from FMEAs to identify and prioritize failures in systems. The risk priority number (RPN) can be evaluated using the following: failure occurrence (O), effect severity (S), and detection difficulty (D) and are evaluated using a 10-point scale.

$$\mathrm{RPN} = \mathrm{O} \times \mathrm{S} \times \mathrm{D} \tag{1}$$

Eq. (1) shows the RPN. Higher RPN values imply greater risks of failure modes.

However, it has been argued that the RPN may not be a good measure of risk [68, 69]. This has led to other modifications of FMEA in order to deal with the difficulties of assigning risk factors. Wang et al. [70] proposed fuzzy risk priority numbers (FRPNs). Using the centroid defuzzification method, FRPNs are defuzzified to distinguish between failure modes. Chin et al.

[71] used the data envelopment analysis (DEA) to identify risk priorities of failure modes measured by overall risks. They also used interval DEA to solve the incomplete and imprecise assessment of FMEA. Also, in order to address the issue of integrating different types of information into the traditional RPN and fuzzy logic methods, Chin et al. [72] developed a new FMEA methodology for multiple attribute decision analysis using the group-based evidential reasoning (ER) approach. Most other fuzzy methods [73, 74] allow the addition of flexibility to FMEA, but still possess limitations from the use of subjective factors.

Another important conventional tool for system reliability is the fault tree analysis (FTA). It is suitable for application in existing systems and new systems simple or complex engineering systems [75, 76]. Shalev and Tiran [75] proposed modifications to these tools and then developed a practical operative tool called condition-based fault tree analysis (CBFTA) to improve system reliability. Dynamic FTA (DFTA) [77] provides another extension of the FTA. This method defined additional gates called the dynamic gates to model complex interactions. Some researchers have also recently used the fuzzy set theory and evidence theory in FTA analysis [78] to reduce the error from the inaccuracy of primary event data.

Inductive methodology tools such as reliability block diagrams (RBD) also help in performing system reliability analysis though the use of graphical representations [79]. The system structure is usually arranged in series or parallel or their combination. Extensions of the RBD method include the RBD method for repairable multi-state systems [80] and the RBD with general gates [81].

2.4 Conceptual Stage Failure Analysis Methodologies

In conceptual design stages, very limited reliability information is usually available. Traditional statistical approaches restrict the information to what is obtained from current relevant data [82, 83]. Addressing the limitations previously discussed on the conventional failure analysis techniques, many Bayesian approaches perform better. All the information available with Bayesian approaches can be used, whether old or new, objective or subjective, or points or interval values.

The Bayes' Theorem is expressed by

$$\pi(\theta/y) = \frac{f(y/\theta)\pi(\theta)}{\int f(y/\theta)\pi(\theta)d\theta} \qquad (2)$$

Where θ is a parameter vector, y is a data vector, $\pi(\theta)$ is a prior probability density function, and $f(y/\theta)$ is the probability density function of the data, referred to as the likelihood when viewed as a function of the parameter vector given the data. The result of integrating the data with prior information in Eq. (2) is the joint posterior distribution. Eq. (2) provides significant flexibility for various types of input information mentioned above [84, 85].

Bayesian methods are also able to integrate lifetime data collected at component, subsystem, and system levels with prior information at any level. A typical Bayesian model for assessing the reliability of such multicomponent systems is discussed in [83]. The model allows sourcing for information from similar components and expert opinions. Several sources of information relevant to estimating system reliability are assumed available such as lifetime data. The relationships between the state of the system and those of components is established and modeled as a series, parallel, or the combination system. Under the assumption that all the component lifetimes are independent, the distribution of the system lifetime is analytically available given the distributions of component lifetimes.

The Bayesian reliability methods have been further expanded by using the Bayesian Network (BN). BN is a probabilistic graphical model, which represents a set of random variables and their conditional dependencies through a directed acyclic graph (DAG). The BN methodology has become a popular approach applied to assess system reliability of nuclear power system, military vehicles, and sensors [86, 87]. Martz et al. [32, 88] used static Bayesian procedure to estimate the reliability of a complex system. Weber and Jouffe [89, 90] developed dynamic Bayesian networks (DBN) to dynamically model and control the complex manufacturing processes. Hamada et al. [91, 92] developed a fully Bayesian approach which automatically propagates the highest-level data to lower levels in the fault tree and developed YADAS software to assess system reliability.

The introduction of simulation packages for use in system modeling has been successful in performing failure interaction studies to improving failure analyses of complex systems. Wang and Li [93] studied the redundancy allocation problems for multistate systems (containing a main subsystem and an auxiliary subsystem, and their possible backups) with failure interactions. They observed there were failure interactions from the auxiliary subsystem to the main subsystem; that is, when the auxiliary subsystem failed, the failure process of the main subsystem increased. They used semi-Markov process models in their system model with two cases; one where all auxiliary subsystems work sequentially and another with all auxiliary subsystems working in parallel and they also allowed their main subsystems work sequentially for both cases. They were able to use an enumeration method to solve the redundancy allocation problem. Through their case study, they were able to show that for the specific applications of multistate systems with failure interactions, the optimal redundancy allocation schemes could be obtained considering different effects of adding redundancy under different failure rates and the repair actions of components that interact with each other.

Innovative approaches to improving how faults are captured in complex system design have considered including fault propagation studies in their model analysis [1, 94]. These studies recommend including all system component in their model for analysis. Fault propagation considers nodes having physical or logical coupling, when any node fails, then adjacent nodes will also appear to have faults. In the conventional process of fault diagnosis, most considerations on fault propagation are done from the perspective of probability analysis. However, from a practical standpoint, the structure of complex system strongly affects the fault propagation of the entire system. Usually, if a node is connected to many nodes, failure of this node (even for nodes with low failure rates) will cause other number of nodes to become faulty. Also a good fault propagation path study will require adequate understanding of some uncertainties that can develop in the system being considered during the course of its operation [1, 94].

Other methods attempt to reconcile various approaches of performing sensitivity analysis, which another method used in conceptual design. Hutcheson and McAdams [95] presented a local sensitivity analysis used for screening a large number of concepts during conceptual design and a global sensitivity analysis performed during the later stages of design. Also, the concept of the multi-stage uncertainty quantification method [96], which was originally developed for model validation, could be modified for uncertainty quantification in conceptual design. Currently, the commonly used approach in industry for quantitative risk analysis is the Probabilistic Risk Analysis (PRA) [19].

2.5 Function-Based System Design

Functional modeling is an important component in concept generation during the design process. It has been used extensively to aid engineers in the generation of system and product requirements, and in system-architectural decision making. Functional modeling is a technique that is used to represent the functionality of a system and it does not depend on the form of the system [97-100]. Functional models typically consist of functions and flows represented as verbs and nouns (such as transmit current, close valve, stir fluid, store data) [101]. Several efforts have been made to formalize the language and syntax used in functional modeling to enhance the usefulness and efficiency of such methods. These efforts are centralized around the idea of defining distinct levels of detail, or abstraction, that provide a contextual lens through which to observe and improve the design process [100, 102, 103]. One such approach is the Functional Basis framework, introduced by Wood et. al.[97]. As decisions are made during the design process, components and subsystems are generated and refined to accomplish the functions required by the product. These components may reveal additional functions that need to be completed.

The Function Failure Design Method (FFDM) methodology was developed from utilizing the function modeling approach. It generates relationships between functional losses and system failure states. The FFDM approach allows for the identification of potential failures prior to commitment of resources to a particular physical design configuration by using historical data of component failures [104]. The Risk in Early Design method augments FFDM by the inclusion of consequence and likelihood values, allowing the designer to understand the results of potential failures [105-107]. Change prediction method applies failure analyses to the Design Structure Matrix (DSM) to evaluate the propagation path of failures [108]. By applying the DSM to the evaluation process, the designer is able to connect changes in failure performance to system architectures.

An extended version of the FFDM is the Functional Failure Rate Design Method (FFRDM) [109, 110]. It utilizes a robust knowledge base and repository data to effectively provide recommendations that mitigate failure modes having high likelihood of occurrence. The component's function-flow failure rates and knowledge of the failure modes provided quantitative reliability results which assists the decision making process in early design phases. A combination of all these previous methodologies became the bedrock for the development of more robust frameworks that would be utilized for early fault assessment such as the Function Failure Identification and Propagation (FFIP).

2.6 Function Failure Identification and Propagation (FFIP)

The Function Failure Identification and Propagation (FFIP) framework [60, 65, 101, 111-113] is an early design stage predictive method which effectively evaluates undesired behavior of complex systems [114, 115]. The functional, behavioral, and component architectural information of a system are generated in order to model and simulate discrete fault scenarios. The system behavior simulation is based on abstract, state-based descriptions of each component behavior. Functional reasoning obtained through the Function Failure Logic (FFL), is then applied to flows of energy, material, and signals (EMS) to confirm if specific component-level functions are influenced by deviations in the behavior of individual components within the system. FFL as a reasoning tool does not rely on form and architecture, which makes it ideal for early design stage decision making. The output of this analysis produces a set of the qualitative health states of each function in the system model. This technique has been effectively applied to assess the fault tolerance, from the functional perspective, for various complex systems, such as aviation electronics [114, 116] and internal combustion engines [117].

As described in Figure 1, the inputs of an FFIP analysis are: critical scenarios, functional and behavioral representations, and mapping logic between the behavior and intended function. The outputs that a designer uses are the system's functional response to the scenario and the system's behavioral response to the scenario.

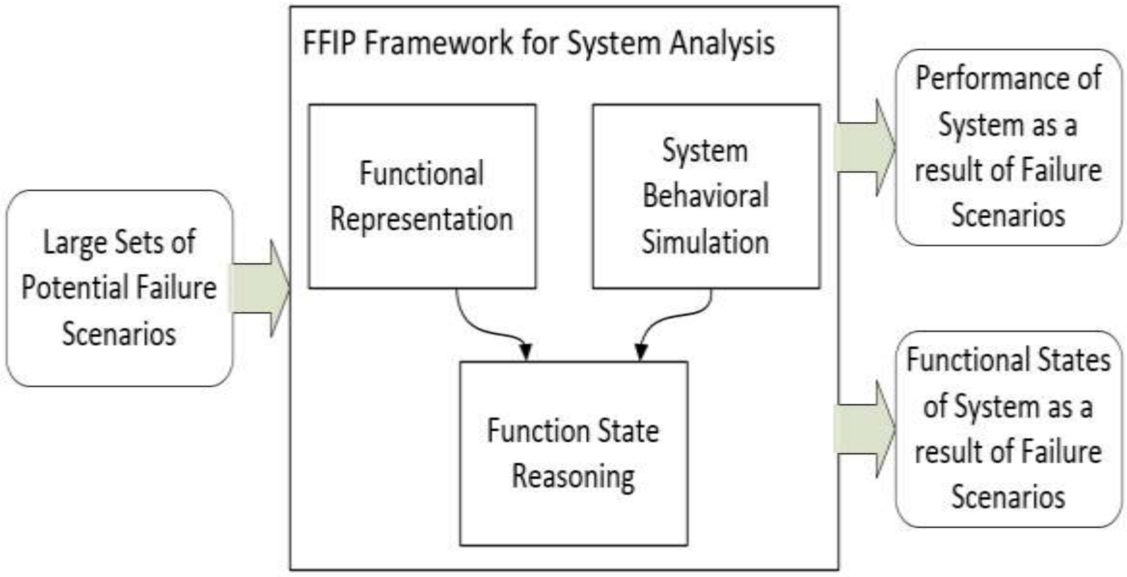

FFIP Framework for System Analysis
Large Sets of Potential Failure Scenarios
Functional Representation
System Behavioral Simulation
Function State Reasoning
Performance of System as a result of Failure Scenarios
Functional States of System as a result of Failure Scenarios

FIGURE 1: FUNCTION FAILURE IDENTIFICATION AND PROPAGATION FRAMEWORK

For complex systems, FFIP framework allows the system modeler to cluster models by high-level functions, adding detail and fidelity as design decisions and parameters are further identified [116]. Systems that share identical or similar components can reuse subsystems of the models, improving the efficiency with which analysis can occur. The overall goal of the FFIP analysis approach has always been to demonstrate the possibility of identifying faults and failure propagation paths by mapping component fault states to function 'health,' described by the qualitative states ‘Healthy,’ ‘Degraded,’ ‘Lost,’ and ‘No Flow,’. These states are further described in the following list:

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 present for the function to affect.

The function failure reasoning introduced by the FFIP methodology is used as the primary logic within the simulation execution. The function failure logic (FFL) reasoning module and a flow state logic (FSL) provide the logic rule that determines the function states of components based on system levels and types. The health states listed above are used to describe the health of the system at any given point during the simulations. Evaluating failure scenarios as they are implemented in models of a system with different fidelity levels provides a different challenge, due to the highly specific demands of defining such scenarios.

While the FFIP method has been shown to successfully reveal fault propagation paths in various systems, its validity is currently being evaluated next to physical platforms to evaluate the usefulness and applicability of a functional representation of system modeling in making system design decisions. However, in this work, previous researches carried out relating to the use of the FFIP methodology on systems will be reviewed. This will serve as primary guidance and validation to the robustness of utilizing the framework in our approach presented amongst other works in this dissertation.

References

[1] Gao, J., Li, G., and Gao, Z., 2008, "Fault propagation analysis for complex system based on small-world network model," Reliability and Maintainability Symposium, 2008. RAMS 2008. Annual, IEEE, pp. 359-364.

[2] Augustine, M., Yadav, O. P., Jain, R., 2012, "Cognitive Map-Based System Modeling for Identifying Interaction Failure Modes," Research in Engineering Design, 23(2) pp. 105-124.

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

[4] Stamatis, D.H., 2003, "Failure mode and effect analysis: FMEA from theory to execution," ASQ Quality Press,

[5] Blischke, W.R., and Murthy, D.N.P., 2000, "Reliability: Modeling, Prediction, and Optimization," John Wiley and Sons,

[6] Teng, X., and Pham, H., 2006, "Reliability Modeling of Hardware and Software Interactions, and its Applications," 55:4pp. 571-577.

[7] RIPLOVÁ, K., 2007, "Tool of Risk Management: Failure Mode and Effects Analysis and Failure Modes, Effects and Criticality Analysis".

[8] Modarres, M., Kaminskiy, M.P., and Krivtsov, V., 2009, "Reliability engineering and risk analysis: a practical guide," CRC press.

[9] Vesely, W.E., Goldberg, F.F., Roberts, N.H., 1981, "The Fault Tree Handbook," US Nuclear Regulatory Commission, NUREG0492, Washington, D.C..

[10] Dhillon, B.S., and Singh, C., 1981, "Engineering reliability: new techniques and applications," Wiley New York.

[11] Ericson, C., 1999, "Fault Tree Analysis–A History from the Proceeding of the 17th International System Safety Conference," .

[12] Wang, J.X., and Roush, M.L., 2000, "What every engineer should know about risk engineering and management," CRC Press.

[13] Huang, D., Chen, T., and Wang, M. J., 2001, "A Fuzzy Set Approach for Event Tree Analysis," Fuzzy Sets and Systems, 118(1) pp. 153-165.

[14] Ferdous, R., Khan, F., Sadiq, R., 2009, "Handling Data Uncertainties in Event Tree Analysis," Process Safety and Environmental Protection, 87(5) pp. 283-292.

[15] Kenarangui, R., 1991, "Event-Tree Analysis by Fuzzy Probability," IEEE Transactions on Reliability, 40(1) pp. 120-124.

[16] Wang, W., Loman, J. M., Arno, R. G., 2004, "Reliability Block Diagram Simulation Techniques Applied to the IEEE Std. 493 Standard Network," IEEE Transactions on Industry Applications, 40(3) pp. 887-895.

[17] Xu, H., Xing, L., and Robidoux, R., 2009, "Drbd: Dynamic Reliability Block Diagrams for System Reliability Modelling," International Journal of Computers and Applications, 31(2) pp. 132-141.

[18] Rausand, M., and Arnljot, H., 2004, "System reliability theory: models, statistical methods, and applications," John Wiley & Sons.

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

[20] Fullwood, R., 1999, "Probabilistic safety assessment in the chemical and nuclear industries," Butterworth-Heinemann.

[21] Kumamoto, H., and Henley, E., "1996, Probabilistic Risk Assessment and Management for Engineers and Scientists, IEEE Press, New York".

[22] 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..

[23] Stamatelatos, M., Dezfuli, H., Apostolakis, G., 2011, "Probabilistic Risk Assessment Procedures Guide for NASA Managers and Practitioners".

[24] Stamatelatos, M., 2000, "Probabilistic Risk Assessment: What is it and Why is it Worth Performing it?" NASA Office of Safety and Mission Assurance, 4(05) pp. 00.

[25] Stewart, M., and Melchers, R.E., 1997, "Probabilistic risk assessment of engineering systems," Springer.

[26] Mobley, R.K., 1999, "Root cause failure analysis," Butterworth-Heinemann.

[27] Wilson, P.F., 1993, "Root cause analysis: A tool for total quality management," ASQ Quality Press.

[28] Andersen, B., and Fagerhaug, T., 2006, "Root cause analysis: simplified tools and techniques," ASQ Quality Press.

[29] Hawkins, P. G., and Woollons, D. J., 1998, "Failure Modes and Effects Analysis of Complex Engineering Systems using Functional Models," Artificial Intelligence in Engineering, 12(4) pp. 375-397.

[30] Tumer, I. Y., and Stone, R. B., 2003, "Mapping Function to Failure Mode during Component Development," Research in Engineering Design, 14(1) pp. 25-33.

[31] Haider, A. A., and Nadeem, A., 2013, "A Survey of Safety Analysis Techniques for Safety Critical Systems," International Journal of Future Computer and Communication, 2(2) pp. 134.

[32] Martz, H., and Wailer, R., 1990, "Bayesian Reliability Analysis of Complex Series/Parallel Systems of Binomial Subsystems and Components," Technometrics, 32(4) pp. 407-416.

[33] Martz, H., Wailer, R., and Fickas, E., 1988, "Bayesian Reliability Analysis of Series Systems of Binomial Subsystems and Components," Technometrics, 30(2) pp. 143-154.

[34] Kmenta, S., and Ishii, K., 2000, "Scenario-based FMEA: a life cycle cost perspective," Proc. ASME Design Engineering Technical Conf. Baltimore, MD,

[35] Kmenta, S., and Ishii, K., 2000, "Scenario-based FMEA: a life cycle cost perspective," Proc. ASME Design Engineering Technical Conf. Baltimore, MD,

[36] Abdelgawad, M., and Fayek, A. R., 2010, "Risk Management in the Construction Industry using Combined Fuzzy FMEA and Fuzzy AHP," Journal of Construction Engineering and Management, 136(9) pp. 1028-1036.

[37] Nachtmann, H., and Chimka, J., 2003, "Fuzzy reliability in conceptual design," Reliability and Maintainability Symposium, 2003. Annual, IEEE, pp. 360-364.

[38] Ormon, S. W., Cassady, C. R., and Greenwood, A. G., 2002, "Reliability Prediction Models to Support Conceptual Design," IEEE Transactions on Reliability, 51(2) pp. 151-157.

[39] 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},

[40] Marini, V. K., Ahmed-Kristensen, S., Kozine, I., 2013, "Information about Robustness, Reliability and Safety in Early Design Phases," .

[41] Little, R. G., 2003, "Toward more robust infrastructure: observations on improving the resilience and reliability of critical systems," System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on, IEEE, pp. 9 pp.

[42] Rinaldi, S. M., 2004, "Modeling and simulating critical infrastructures and their interdependencies," System sciences, 2004. Proceedings of the 37th annual Hawaii international conference on, IEEE, pp. 8 pp.

[43] Estefan, J. A., 2007, "Survey of Model-Based Systems Engineering (MBSE) Methodologies," Incose MBSE Focus Group, 25(8) .

[44] Friedenthal, S., Griego, R., and Sampson, M., 2007, "INCOSE model based systems engineering (MBSE) initiative," INCOSE 2007 Symposium,

[45] Lavi, J. Z., and Kudish, J., 2005, "Systems Modeling & Requirements Specification using ECSAM: An Analysis Method for Embedded & Computer-Based Systems," Innovations in Systems and Software Engineering, 1(2) pp. 100-115.

[46] Military, U., 1992, Reliability Prediction of Electronic Equipment.

[47] Wood, W. H., and Agogino, A. M., 2005, "Decision Based Conceptual Design: Modeling and Navigating Heterogeneous Design Spaces," 127(1) pp. 2-11.

[48] Wood, G. D., and Kennedy, D. C., 2003, "Simulating Mechanical Systems in Simulink with SimMechanics," .

[49] Mossberg, A. I.,O., 2014, "Modeling Aircraft Flap System Failure Scenarios with SystemModeler@ONLINE," .

[50] Malone, B., and Papay, M., 1999, "ModelCenter: An Integration Environment for Simulation Based Design," Simulation Interoperability Workshop,

[51] Mahapatra, S., Egel, T., Hassan, R., 2008, Model-Based Design for Hybrid Electric Vehicle Systems.

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

[53] Zang, T., Hemsch, M. J., Hilburger, M. W., 2002, "Needs and Opportunities for Risk-Based Multidisciplinary Design Technologies for Vehicles," NASA TM, July.

[54] Backman, B., 2000, "Design Innovation and Risk Management: A Structural Designer's Voyage into Uncertainty," ICASE Series on Risk-Based Design.

[55] Choi, K., 2001, "Advances in Reliability-Based Design Optimization and Probability Analysis-PART II," ICASE Series on Risk-Based Design.

[56] Smith, N., and Mahadevan, S., 2003, "Probabilistic Methods for Aerospace System Conceptual Design," Journal of Spacecraft and Rockets, 40(3) pp. 411-418.

[57] Venkatasubramanian, V., Zhao, J., and Viswanathan, S., 2000, "Intelligent Systems for HAZOP Analysis of Complex Process Plants," Computers & Chemical Engineering, 24(9) pp. 2291-2302.

[58] Greenfield, M. A., 2000, "NASA's Use of Quantitative Risk Assessment for Safety Upgrades," IAAA Symposium,

[59] Hong, Y., Adler, R., and Huffman, G., 2006, "Evaluation of the Potential of NASA Multi-satellite Precipitation Analysis in Global Landslide Hazard Assessment," Geophysical Research Letters, 33(22) .

[60] Kurtoglu, T., and Tumer, I. Y., 2007, "Ffip: A Framework for Early Assessment of Functional Failures in Complex Systems," ICED, Cite Des Sciences Et De L'industrie, Paris, France.

[61] Wertz, J.R., and Larson, W.J., 1999, "Space Mission Analysis and Design, 3rd Edition," Space Technology Library, Microcosm, Kluwer Academic, Dordrecht.

[62] Kong, J. S., and Frangopol, D. M., 2003, "Life-Cycle Reliability-Based Maintenance Cost Optimization of Deteriorating Structures with Emphasis on Bridges," Journal of Structural Engineering, 129(6) pp. 818-828.

[63] Henley, E.J., and Kumamoto, H., 1981, "Reliability engineering and risk assessment," Prentice-Hall Englewood Cliffs (NJ).

[64] Carmignani, G., 2009, "An Integrated Structural Framework to Cost-Based FMECA: The Priority-Cost FMECA," Reliability Engineering & System Safety, 94(4) pp. 861-871.

[65] Kurtoglu, T., and Tumer, I. Y., 2008, "A Graph-Based Fault Identification and Propagation Framework for Functional Design of Complex Systems," 130(5) .

[66] Kmenta, S., and Ishii, K., 2000, "Scenario-based FMEA: a life cycle cost perspective," Proc. ASME Design Engineering Technical Conf. Baltimore, MD,

[67] Pickard, K., Muller, P., and Bertsche, B., 2005, "Multiple failure mode and effects analysis-an approach to risk assessment of multiple failures with FMEA," Reliability and Maintainability Symposium, 2005. Proceedings. Annual,   IEEE, pp. 457-462.

[68] Gilchrist, W., 1993, "Modelling Failure Modes and Effects Analysis," International Journal of Quality & Reliability Management, 10(5) .

[69] Harpster, R., 1999, "How to Get More Out of Your FMEAs," Quality Digest, 19pp. 40-42.

[70] Wang, Y., Chin, K., Poon, G. K. K., 2009, "Risk Evaluation in Failure Mode and Effects Analysis using Fuzzy Weighted Geometric Mean," Expert Systems with Applications, 36(2) pp. 1195-1207.

[71] Chin, K., Wang, Y., Poon, G. K. K., 2009, "Failure Mode and Effects Analysis by Data Envelopment Analysis," Decision Support Systems, 48(1) pp. 246-256.

[72] Chin, K., Wang, Y., Poon, G. K. K., 2009, "Failure Mode and Effects Analysis using a Group-Based Evidential Reasoning Approach," Computers & Operations Research, 36(6) pp. 1768-1779.

[73] Liu, H., Liu, L., Bian, Q., 2011, "Failure Mode and Effects Analysis using Fuzzy Evidential Reasoning Approach and Grey Theory," Expert Systems with Applications, 38(4) pp. 4403-4415.

[74] Liu, H., Liu, L., Liu, N., 2012, "Risk Evaluation in Failure Mode and Effects Analysis with Extended VIKOR Method Under Fuzzy Environment," Expert Systems with Applications, 39(17) pp. 12926-12934.

[75] Shalev, D. M., and Tiran, J., 2007, "Condition-Based Fault Tree Analysis (CBFTA): A New Method for Improved Fault Tree Analysis (FTA), Reliability and Safety Calculations," Reliability Engineering & System Safety, 92(9) pp. 1231-1241.

[76] Volkanovski, A., Čepin, M., and Mavko, B., 2009, "Application of the Fault Tree Analysis for Assessment of Power System Reliability," Reliability Engineering & System Safety, 94(6) pp. 1116-1127.

[77] Čepin, M., and Mavko, B., 2002, "A Dynamic Fault Tree," Reliability Engineering & System Safety, 75(1) pp. 83-91.

[78] Ferdous, R., Khan, F., Veitch, B., 2009, "Methodology for Computer Aided Fuzzy Fault Tree Analysis," Process Safety and Environmental Protection, 87(4) pp. 217-226.

[79] Čepin, M., 2011, "Assessment of power system reliability: methods and applications," Springer Science & Business Media.

[80] Lisnianski, A., 2007, "Extended Block Diagram Method for a Multi-State System Reliability Assessment," Reliability Engineering & System Safety, 92(12) pp. 1601-1607.

[81] Kim, M. C., 2011, "Reliability Block Diagram with General Gates and its Application to System Reliability Analysis," Annals of Nuclear Energy, 38(11) pp. 2456-2461.

[82] Johnson, V. E., Moosman, A., and Cotter, P., 2005, "A Hierarchical Model for Estimating the Early Reliability of Complex Systems," IEEE Transactions on Reliability, 54(2) pp. 224-231.

[83] Reese, C. S., Wilson, A. G., Guo, J., 2011, "A Bayesian Model for Integrating Multiple Sources of Lifetime Information in System-Reliability Assessments," Journal of Quality Technology, 43(2) pp. 127.

[84] Weber, P., Medina-Oliva, G., Simon, C., 2012, "Overview on Bayesian Networks Applications for Dependability, Risk Analysis and Maintenance Areas," Engineering Applications of Artificial Intelligence, 25(4) pp. 671-682.

[85] Oliva, G. M., Weber, P., Simon, C., 2009, "Bayesian Networks Applications on Dependability, Risk Analysis and Maintenance," IFAC Proceedings Volumes, 42(5) pp. 215-220.

[86] Langseth, H., and Portinale, L., 2007, "Bayesian Networks in Reliability," Reliability Engineering & System Safety, 92(1) pp. 92-108.

[87] Pourret, O., Naïm, P., and Marcot, B., 2008, "Bayesian networks: a practical guide to applications," John Wiley & Sons.

[88] Martz, H., Wailer, R., and Fickas, E., 1988, "Bayesian Reliability Analysis of Series Systems of Binomial Subsystems and Components," Technometrics, 30(2) pp. 143-154.

[89] Weber, P., and Jouffe, L., 2003, "Reliability modelling with dynamic bayesian networks," In 5th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes (SAFEPROCESS'03), Washington, DC, USA, IFAC, pp. 57-62.

[90] Weber, P., and Jouffe, L., 2006, "Complex System Reliability Modelling with Dynamic Object Oriented Bayesian Networks (DOOBN)," Reliability Engineering & System Safety, 91(2) pp. 149-162.

[91] Hamada, M., Martz, H. F., Reese, C. S., 2004, "A Fully Bayesian Approach for Combining Multilevel Failure Information in Fault Tree Quantification and Optimal Follow-on Resource Allocation," Reliability Engineering & System Safety, 86(3) pp. 297-305.

[92] Graves, T. L., and Hamada, M., 2004, "Combining Multi-Level Data to Assess System Reliability and Allocate Resources Optimally: Bayesian Methods and Computation," Los Alamos National Laboratory Technical Report, LA-UR-04-7157.

[93] Wang, J., and Li, M., 2015, "Redundancy Allocation Optimization for Multistate Systems with Failure Interactions using Semi-Markov Process," Journal of Mechanical Design, 137(10) pp. 101403.

[94] WANG, Y., SHI, H., and LIN, S., "Fault Propagation of System Network Based on State Transition Equation Modelled".

[95] Hutcheson, R. S., and McAdams, D. A., 2010, "A Hybrid Sensitivity Analysis for use in Early Design," Journal of Mechanical Design, 132(11) pp. 111007.

[96] Hoyle, C., Tumer, I. Y., Kurtoglu, T., 2011, "Multi-stage uncertainty quantification for verifying the correctness of complex system designs," ASME 2011 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. 1169-1178.

[97] Wood, K. L., 2000, "Development of a Functional Basis for Design," Journal of Mechanical Design, 122pp. 359-370.

[98] Stone, R., Wood, K., and Crawford, R., 2000, "Using Quantitative Functional Models to Develop Product Architectures," 21pp. 239-260.

[99] Stone, R. B., Tumer, I. Y., and VanWie, M., 2005, "The Function Failure Design Method," 14pp. 25-33.

[100] Hirtz, J., Stone, R., McAdams, D., 2002, "A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts," 13pp. 65-82.

[101] Tumer, I., and Smidts, C., 2011, "Integrated Design-Stage Failure Analysis of Software-Driven Hardware Systems," IEEE Transactions on Computers, 60(8) pp. 1072-1084.

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

[103] Umeda, Y., Ishii, M., Yoshioka, M., 1996, "Supporting Conceptual Design Based on the Function-Behavior-State Modeler," Artificial Intelligence for Engineering Design, 10(4) pp. 275-288.

[104] Stone, R. B., Tumer, I. Y., and Stock, M. E., 2006, "Linking Product Functionality to Historical Failures to Improve Failure Analysis in Design," 16(2) pp. 96-108.

[105] Grantham-Lough, K., Stone, R. B., and Tumer, I. Y., 2009, "The Risk in Early Design Method," 20(2) pp. 144-173.

[106] Grantham-Lough, K., Stone, R. B., and Tumer, I. Y., 2008, "Implementation Procedures for the Risk in Early Design (RED) Method," 2(2) pp. 126-143.

[107] Grantham-Lough, K., Wie, M. V., Barrientos, F., 2009, "Promoting Risk Communication in Early Design through Linguistic Analyses and Tools," 20(1) pp. 29.

[108] Clarkson, P., Simons, C., and Eckert, C., 2004, "Predicting Change Propagation in Complex Design," 126pp. 788.

[109] O'Halloran, B. M., Stone,R.B., and Tumer, I. Y., 2011, "Link between function-flow failure rates and failure modes for early design stage reliability analysis," Proc. ASME 2011 International Mechanical Engineering Congress and Exposition; American Society of Mechanical Engineers}, pp. 457-467.

[110] O'Halloran, B. M., Stone,R.B., and Tumer, I. Y., 2011, "Early design stage reliability analysis using function-flow failure rates," Proc. ASME 2011 International Design Engineering Technical Conferences; IDETC/CIE}, pp. 455-464.

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

[112] 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," San Diego, CA.

[113] 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},

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

[115] Kurtoglu, T., Johnson, S., Barszcz, E., 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, PHM'08,

[116] Jensen, D. C., Bello, O., Hoyle, C., 2014, "Reasoning about System-Level Failure Behavior from Large Sets of Function-Based Simulations," Artificial Intelligence for Engineering Design, 28(04) pp. 385-398.

[117] O'Halloran, B. M., Haley, B., Jensen, D. C., 2014, "The Early Implementation of Failure Modes into Existing Component Model Libraries," Research in Engineering Design, 25(3) pp. 203-221.

CHAPTER 3

FFIP METHODOLOGY APPLICATIONS IN CONCEPTUAL DESIGN RESEARCH

This section is focused on reviewing various analyses relating to the application of the Functional Failure Identification and Propagation (FFIP) framework to complex systems over the past decade. The history of FFIP and its importance in the domain of failure analysis in different complex systems are illustrated. The contributions made by the FFIP methodology and the current research applications evolving from it for safety in complex engineering systems are discussed. Other works related works adapting the FFIP methodologies and its hybrids are also discussed.

The key questions on function-based failure analysis research that are addressed in this chapter are:

- What types of systems are evaluated?
- What types of modeling are used?
- What types of analysis are used in carrying out the researches?

This will provide guidance to researchers and system designers towards aiding the decision-making process by understanding the importance of applying this method, suitable at the conceptual stage of designs. The researches relating to FFIP have all been grouped into the categories shown in Figure 2.1 below.

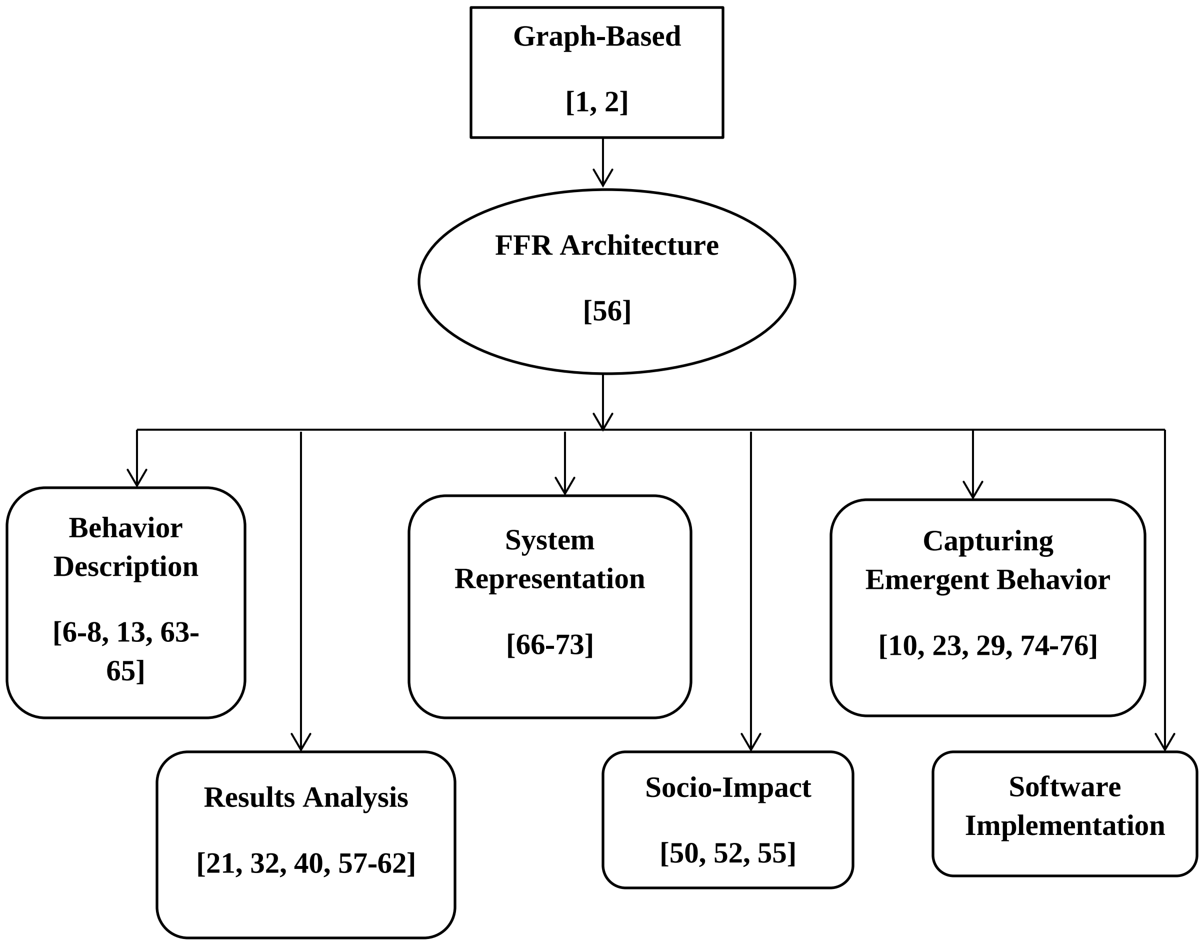


FIGURE 3.1: FFIP RESEARCH TREE

3.1 A Graph-Based Representation of Complex System Models (FFIP) [1, 2]

The FFIP framework was introduced by Kurtoglu and Tumer [1, 2] as an approach that assesses the functional-failure risks of physical systems during the conceptual stage of the design process. In their paper, they described creating system models using graphical representations where FFIP system functions were represented as using function structures. A combination of hierarchical system models with behavioral simulation and qualitative reasoning, were used to develop their framework which highlights faults and their propagation paths when failure scenarios are triggered.

Graphical models, behavioral simulation and FFL reasoners were illustrated as three major components vital to the FFIP framework. Using a Hold-Up tank system model as case study, a functional model and configuration flow graph (CFG) [3] capturing a direct map between the functional and the structural architecture of the system [4] were built. The CFG is a specific implementation of the topology or the configuration layout of systems. It follows the general functional topology of a system and maps the desired functionality into the component configuration domain.

The behavioral simulation was developed using high-level, qualitative models of system components at various discrete nominal and faulty modes. Transitions between these discrete modes were defined using mode transition diagrams. Component behaviors in each mode are the input-output relations and underlying first principles governing the components operation. All the models followed the form of the CFG hence, the state variables critical to the system behavior were incorporated into the representations by associating them with their respective CFG flows.

The overall assessment of potential functional failures and fault propagation paths were made through a reasoner that translates the input and output variable state changes in the system configuration graph into functional failures. The function failure logic (FFL) reasoner defines a set of form-independent system function models that describe conditions under which functions deviate from their intended operation. The results of the FFL are used to classify each system function as operating, degraded, or lost.

The sets of novel discoveries that were made using the FFIP approach to test faulty conditions on the Hold-Up tank are as follows:

(1) FFL and FFIP conveniently reasons at the function level to assess the impact of failures on system performance;

(2) FFIP does not require designers to make prior assumptions to speculate fault propagation paths of causal relationships;

(3) FFIP has the ability to capture various non-trivial, non-linear fault propagation paths by adequately considering the maps between the function, the physical structure, and the behavior of a system; and

(4) The extensive ability of FFIP to identify functional failures arising from global component interactions instead of direct component links.

The authors concluded that by using a model paradigm capable of representing desired components functionality, structure and interactions, the FFIP method effectively integrates fault prevention and management. This is done by systematically exploring risks and vulnerabilities without committing to design decisions at the conceptual stages of system designs.

3.2 Functional Failure Reasoning Methodology [56]

Following the introduction of the FFIP framework, an effective reasoning methodology for comparing the input-output relationship of component models was evaluated. Kurtoglu et al [5], proposed a function failure reasoning (FFR) approach suitable for failure analysis at early design phases of complex systems. The method was based on two assumptions: that failure occurs when functional elements in a system deviate from performing intended tasks; and risk is dependent on the role of functionality in accomplishing designed tasks.

An aerospace vehicle's electric power system (EPS) that supplies power to the vehicle's subsystems was used as the case study for the FFIP analysis. A physical software-hardware testbed provided by the Advanced Diagnostic and Prognostic Testbed (ADAPT) at NASA Ames Research Center where automated fault diagnosis were recorded was utilized. Function criticality ratings (FCR) were used to identify system critical elements incorporated in EPS model needed for its effective operation. The FCR for each system sub-function were determined by comparing the criticality of individual system function and converting the ratings to a normalized coefficient based on the combined criticality of all system functions. The different components needed for redundancy and reconfiguration capability by the EPS to perform its functions of power storage, power distribution and load operation were identified.

The functional model (for component function) and CFG (for component behavior) capturing the direct mapping between the functional and structural architecture of the system were created. The function-failure logic (FFL) module of the FFIP framework used its reasoner to determine the condition state of each system function (i.e. if operational, degraded, or lost). The FFL reasoner received information on the state of the system at the end of each time steps and the state of each system function gets evaluated at the discrete points. Thus, the FFL reasoner translates the dynamics within the system into function failure identifiers and facilitates the assessment of potential function failure and resulting fault propagation paths. By comparing the values of the input and output states of the CFG of a particular component, the FFL allows the assessment of the operability of its designated function.

Upon running the FFIP analysis on the system, the functional failure impact (FFI) of selected scenarios were calculated by summing all the FCR for all elemental sub-functions classified as deviants from the systems' nominal operating state and multiplying it by a consequential cost factor. A Reduction in risk (RIR) value was also calculated to quantify the amount in risk reduction based on a specific architectural change. The implementation of the RIR is based on the assumption that the severity of consequence of failure can be reduced by making architectural changes to reduce the risks associated with certain functional elements in a system. The value helped in deciding what most efficiently mitigates risks associated with functional elements in a design.

Thirty scenario cases for the EPS design were used to show how the FFR methodology evaluates different conceptual system architectures based on functional failure impact. This framework built to support tackling multiple failures, provided an analytical approach to quantify individual risk of basic functional elements in the system together with the combine risk emanating from the functional failures propagation.

3.3 FFIP Related Behavior Description-based Research [6-8, 13, 63-65]

In this section, different works based on component behavioral model exploration in relation to their inherent systems are studied. A description of different methods employed in building behavioral models and the impact of such information has on the system analysis are highlighted. The following researches present works that have been done on behaviour model representations in relation to using the FFIP framework.

3.3.1 Fault-based Behavior Modeling

O'Halloran et al [6] worked on increasing design verification and validation during complex systems design as an alternative method to capture faulty behavior by developing a framework to create component behavior models. The behavior model adopted was built based on failure mechanisms using a gear as case study for system analysis. The performance of the system was chosen for measurement rather than its functional health. This was done by defining an ideal output for a given input and assuming that a fault mechanism led to failure in performance of the system. They identified key elements of generating fault mechanism models. These elements involve generating faulty behavior variables, defining nominal behavior for a component using faulty behavior variables, linking component fault modes to fault interactions defining how to construct the fault mechanism model and how the fault model affects performance.

The results of their analysis showed that taxonomy can be used to describe and catalogue fault events. Among the multiple steps that comprise the fault event description, hierarchical fault mechanism taxonomy was created to describe all potential fault mechanisms in primary, secondary and tertiary terms. These described fault mechanisms at different levels of abstraction to accommodate designers' needs during fault analysis. The results further demonstrated that the number of fault modes and mechanisms increased with the addition of more components to the design. This highlighted a new challenge of modeling behaviour of complex systems with a large number of components. The paper proposed using hierarchical fault mechanisms taxonomy through which the fault-based behavior models generated could be organized.

3.3.2 Building Dimensionless Behavioral Models

Coatanea et al [7, 8] modelled component behavior in the context of the FFIP method in order to complement qualitative reasoning using the dimensional analysis. Dimensional analysis utilizes the properties of physical quantities (mass, length, time etc.) to model physical phenomena through the knowledge of their working relationship [9]. On the other hand, behavior models describe qualitative behavior for component types based on mode transitions used in the function-based failure analysis [1, 10]. Also, behavior models provide a means to integrate with requirements engineering, functional models and configuration models.

Using a fluidic system as case study [11], the capabilities of evaluating and assessing failure risk of physical systems during early design was demonstrated. A more developed version of the product theorem, the Vashy-Buckingham theorem, was used to provide the potential to generate complex interaction models through the composition properties. The theorem demonstrated that the physical description of a phenomenon can easily be reduced into its minimum set of variables by combining the dimensions involved in order to obtain only dimensionless variables. A function and configuration model of the fluidic system was developed using this theorem.

Component behavior was modelled using dimensional analysis principles and causal ordering algorithm by integrating extracted requirement information to create behavioral component models driven by physical quantities and their compositions. The component behavioral models were then created based on key design variables and their physical quantities. The behavioral component models were presented in the form of an interaction graph between design variables associated with units and physical quantities. Performance and repeating variables were also used in modeling the interaction graphs. They were grouped as power variables (i.e., flow and effort), state variables (i.e., displacement and momentum) and connecting variables [12]. The power variable and state variable [11] sets were the two groups from which the performance variables were selected.

Upon running an FFIP analysis on the interaction model, failures and their propagations were identified. Amongst other result, their approach also helped in determining the variables of a design problem at early design stage using a representation of the configuration model.

3.3.3 Implementing Failure Modes Models Into Behavior Models

O'Halloran at al [13] assessed limitations of current libraries in modelling failures when accounting for the performance of a design in its intended operational environment. Transfer function with use case graphs and existing failure modes were utilized in building failure mode models to address these limitations. The approaches developed used the Modelica Standard Library (MSL) as the component library of nominal models while a basic vehicle powertrain model was used as case study.

Figure 2.2 below summarizes the approach that was developed. It begins with information on how behavior variables are affected by a modeled failure.

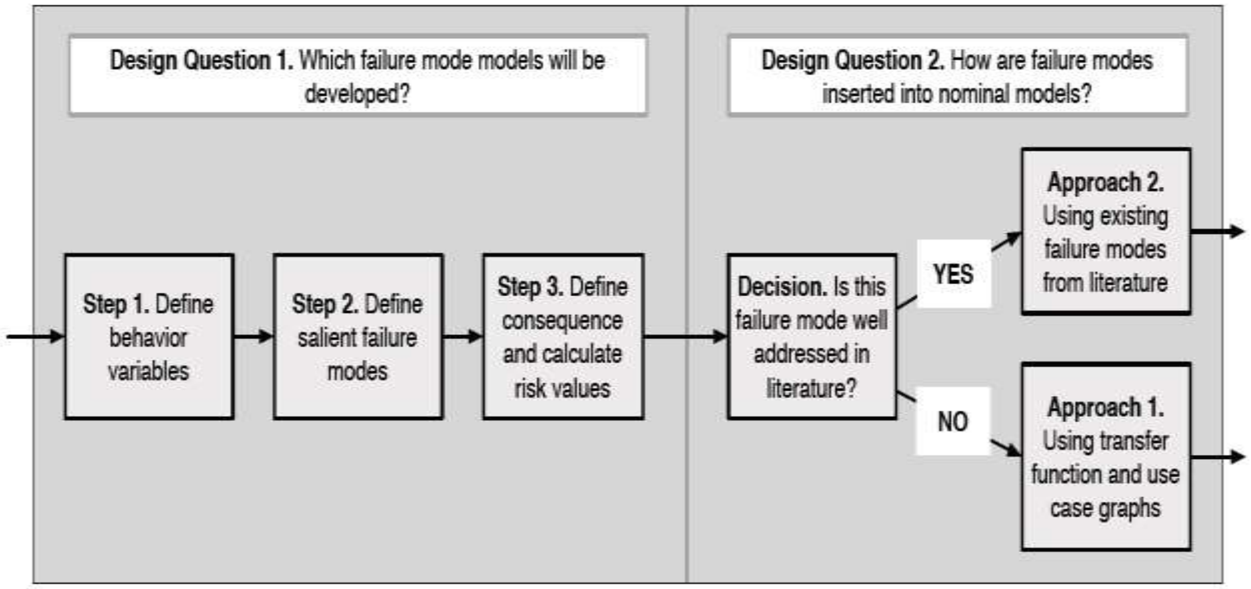

Design Question 1. Which failure mode models will be developed?
Design Question 2. How are failure modes inserted into nominal models?
Step 1. Define behavior variables
Step 2. Define salient failure modes
Step 3. Define consequence and calculate risk values
Decision. Is this failure mode well addressed in literature?
YES
NO
Approach 2. Using existing failure modes from literature
Approach 1. Using transfer function and use case graphs

FIGURE 3.2: FLOWCHART DESCRIBING THE SELECTION AND INSERTION OF FAILURE MODES INTO NOMINAL PHYSICS-BASED MODELS [13]

The behavior variables were derived based on functionality and they represent variables that affect behavior in the presence of a failure. The Failure Modes/Mechanism Distributions 1997 (FMD-97) report [46] was utilized in generating a list of significant failure modes together with the information on how they occur. This was useful in defining the salient failure modes and it also helped in determining the impacts associated with each component/subsystem failure mode while helping to compute their risk values.

Two approaches were used in implementing each failure mode into the corresponding component model. The first approach uses basic transfer function and use case graphs in understanding a component's nominal and failure output. The Second approach uses existing literature to provide information about the failure mode of components. Using the literature approach limited the application to high fidelity that behavior models libraries are unable to capture.

A validation of the built failure mode model was carried out to ascertain an accurate description of the failure space of the system. A simulation analysis from using the two approaches in the failure mode models provided significant results. A classification scheme was then done to identify the general characteristics of failure mode behavior when implemented in flow-based behavioral models. The classifications were derived from modeling different failure modes in multiple components using different domains. This procedure illustrated a broader application across components within model libraries. The classification results also included descriptions of failure mode behavior together with the variables affected by the failure mode for finding relationships between such parameters.

3.4 FFIP Related System Representation-based Research [66-73]

An overview of how systems are represented during modelling and simulation analysis in existing function-based works is discussed below.

3.4.1 Integrating Software and Hardware Systems for Failure Analysis

Tumer and Smidts [14] presented a means of evaluating how a combination of software-hardware system behaves and how failure propagation in them results in potential failures downstream, during the conceptual design stage. High-level system modeling and model-based reasoning approaches were used to model failure propagation in combined software-hardware systems, using Function-Failure Identification and Propagation (FFIP) analysis framework; this helped in formalizing the design of safety-critical systems.

The intent of the work was to bridge the gap between hardware and software designers due to difference in background, knowledge, methods, or language which have significant impact in building software/hardware failure interactions. The FFIP framework offered a unification of the languages and modeling concepts suitable for system analysis. The redundancy management system of the Reaction Control System (RCS) jet from the NASA space shuttle was selected as the case study. The function of RCS jets is to help space vehicles with maneuvering during missions through controlled combustions of fuel and an oxidizer.

The redundancy management software receives the signals from RCS of temperature, pressure and valve position for all jets and also receives a signal from their reaction jet drivers (RJDs) indicating the command sent to the fuel and oxidizer valves. During operation, leaks in either the fuel or the oxidizer lines are monitored with the use of temperature sensors on the injectors and in the jet exhaust. In the event of failure, the monitors will flag after three clock cycles. Based on this knowledge, the software and hardware component models of the RCS jet were built together using the FFIP procedure.

The identified components' function and configuration models were mapped appropriately for the system. The primary flow within the software components for this work is the data being transferred. The software component components were represented in a functional model, this only allowed the component behavior to be limited to ”functioning” and ”not functioning” states. It was also noted that for complex distributed software systems, modeling software systems prone to intermittent failures can add ”intermittent functioning” state to the behavior models.

In order for proper mapping to the software domain, the integration process involved the development of a software functional basis analogue to the FFIP hardware functional basis, a software component basis analogue to the FFIP hardware component basis and a matching hardware and software design representations. These were described as the main elements required to develop a truly integrated analogue representation that allow cross-connections between the software and hardware design spaces.

Imitating the hardware components functional basis, a vocabulary of software functions was developed through the analysis of the specification of the RCS. Three generic control functions were identified namely, “control”, “configure”, and “measure” while a more specific instance of these functions was termed “open”, “close” or “regulate”. The main advantage of the software functional basis is the ability to represent a software system using a restricted number of terms/abstractions and the increasing degree of specificity allows for repeatable and systematic progressive refinement of the design.

A software component ontology organized around flows and structured from specific to generic was developed for the set of application interest [15-17]. The components were the logical components of the software application while the ontology maps relationship between software components and software functions.

From the FFIP analysis performed using two failure scenarios, the dependency and inadequacy of two software monitors were found. The results of the analysis highlighted needs for making pertinent design changes. The advantages of integrating the FFIP underlying concepts and analysis mechanisms into software intensive systems using either functional development were discussed. It was concluded that with further research into incorporating software model into the FFIP framework, a useful software tool could be developed to assist design engineers in the analysis, evaluation, and comparison of complex systems in the conceptual design stage, where decisions are reversible and costs of changes are still minimal.

3.4.2 A Functional Modeling-Based Methodology

If the predictions of a fault detection system can be tested and confirmed or disproved, then their value is increased. This led to the development of an extension of the Hierarchical Functional Fault Detection and Identification (HFFDI) system by adding a test preparation and test-based verification phase. The method was introduced to generate tests that can confirm or disprove the existence of specific faults in a monitored process. Utilizing these tests will assist when testing predictions of the fault detection system.

Verification is a process that reassures the correct function of a system according to its specification [18]. In this research, the focus was on checking that the system keeps being error-free during operation. Formal verification techniques and model checking [19] are example strategies for verification, which can prove the correctness of a system. When formal methods are not applicable or practical (e.g. complex systems), then testing can be used for verification [20]. The basic concepts of verification were extended to the HFFDI system using a test generation and a testing phase to verify the fault detection results. Relatively minor automation faults were targeted in redundant systems as this system can complement the emergency operating procedures designed to handle more serious events. A generic Nuclear Power Plant was used as case study for this research. It was used for training and testing the HFFDI system [21] and for testing predictions.

The FFIP framework was utilized in determining the functional decomposition of the system. The framework was also used to develop the function-to-component and the function-to-process signal mappings that are used in defining the tests. The functional model of the system was mapped to components of a Configuration Flow Graph (CFG). These models contained the behavioral logic, which describes how the system degrades during a specific fault or initiating event. Each system function is linked to a specific Functional Failure Logic (FFL), which uses the simulation signals from the CFG and reasons about the functional health of the function.

A HFFDI system that combines a plant-wide FDI system and a set of function-specific FDI systems was used as the basis fault detection system due to its ability in overall system prediction and the possibility of testing function-specific predictions. The decision tree machine learning algorithm was used to generate tests for the predictions of a data-driven fault detection and identification system. The HFFDI development methodology required using functional decomposition and the function- to-component mappings of FFIP to generate function-specific training and testing data sets, which were used as source for training and testing a plant-wide FDI system and multiple function-specific FDI systems. The FDI systems were run in parallel, and a reasoner combined their results to provide the final HFFDI fault prediction. The logic of the reasoner looks for agreement between the function-specific FDIs, and then its prediction overrides the prediction of the plant-wide FDI. Otherwise the plant-wide FDI prediction is used as the HFFDI output.

The faults considered were pairings of a process component and a failure mode of that component. The knowledge of the information on the component name, the component type and the failure mode were used to force the component to the state of the failure and the transient response of the process was monitored. The monitored process signals were determined by the function the component was mapped to. The command driving components to failure states and the resulting transients were then saved to the library of tests. If, during the operation of the HFFDI, a fault was identified as a predicted fault, then the saved test was performed. If the results of the test match the transients stored in the library of tests, then it was concluded that the component was healthy. If the test results do not match the expected transients, then the component was identified as faulty.

For the case study, two predictions in single fault scenarios and one prediction in a two-fault scenario were tested. The test results gave the correct output for every predicted fault, the successful predictions are confirmed, and the incorrect prediction was disproved. The test generation was done manually and the tests of the HFFDI predictions were also judged manually and subjectively.

3.5 FFIP Related Emergence Capturing Research [10, 23, 29, 74-76]

Most times, complex systems exhibit behaviors from interactions within their subsystems that are not intended or planned by the original designer. These behaviors are called emergent behavior. These are often caused by poorly chosen design parameters in other subsystems. Examples of the use of function-based methodologies within the scope of studying emergent behavior are discussed below.

3.5.1 Simulating Interactions and Emergent Failure Behavior

Papkonstantinou et al [22, 23] addressed some identified technical challenges that arise during the design stages of large complex systems. The challenges identified for their resolution were codesign of the multiple domains of technology; determining emergent behavior effects; and determining risks across a system from fault propagation.

Codesign was used in this work to reference technologies that require the close integration of electrical hardware and software systems similar to those in mechatronic and consumer electronic systems [24]. Some challenges exist in representing the necessary system design information across technical domains at similar and relatable abstraction levels. The advantage of developing different subsystems concurrently during the design stage is as a result of their existence at various levels of design refinement. Hence, the utilization of a formal model representation language such as Systems Modeling Language (SysML) [25] would help capture design information across domains when subsystems are at different levels of design refinement.

Emergent behavior was defined as the degradation or loss of the functionality of a subsystem due to poorly chosen design parameters in other subsystems. Methods and tools for studying how changing several such parameters impact the occurrence of emergent behavior were provided. An extension of the FFIP framework and its supporting tools were used to address challenges involving emerging behavior. The paper explains that since high fidelity simulation at an early design phase was not possible then the results should not totally depend on specific model parameter values. Hence, the simulation carried out varied the values of key design parameters and the timing of critical events; the simulation results revealed the impact of the variations on the emergent behavior. The use of the extended form of FFIP framework was demonstrated on a boiling water reactor (BWR) model.

From the default FFIP approach, the component behavior models determine the output flows from the input flow values and the current state of the component. The previous works on the FFIP framework utilized discrete set of flow state values and a simple behavioral logic, which had the advantage of limiting the range of possible parameter values, but lacked the possibility of modeling continuous process dynamics. The previous approach did not sufficiently capture how several parameter changes influence each other to cause emergent behavior. This resulted in the modification of the framework 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.

A summary of the general operation of the BWR was done to fully understand the subsystem functions. The methodology used to analyze the BWR started by defining a number of values of interest for the parameters to be varied and then systematically perform FFIP simulation to identify the combinations of parameter values leading to degradation or loss of functions. While applying the FFIP framework, the CFG and functional models of the BWR were created to have the same flows between functions and components. This allows the function failure logic (FFL) to passively observe how abnormal flow levels propagate in the simulated CFG, and to use the information to determine if a function defined in the functional model becomes degraded or lost.

The relationship between input and output flows of a component in the CFG were defined in a component behavioral model. The behavioral models in this work were a system of first order linear difference equations relating input flow, output flow and components internal variables. The behaviors of electromechanical components were described using the first order linear approximations, which were implemented as Simulink blocks. This approach provided an avenue for more sophisticated modelling that would be appropriate for early concept design phase. State charts were used to create the behavioral models and a state was defined for each nominal and failed mode of the component. An example of a sample-failed mode is a leaking tank. Critical events were injected to the simulation at any time, and these cause mode changes (e.g., the leakFailure event triggers a transition to the TankLeaking state.)

The effects of utilizing different parameter values and different timing of critical events were investigated by running the FFIP simulation for each combination of parameter values. As more parameters were introduced into the scope of the study, the number of simulations runs grew exponentially. The identified parameters range from design parameters, timing of critical event scenarios or parameters of faults. A user-interface for specifying parameters and the variation ranges made the process feasible. A generic and scalable algorithm was used for automatically running complete sets of selected simulations. The algorithm and user-interface were implemented and interfaced through Matlab/Simulink based on the FFIP framework. The results were produced as Excel outputs for each run, that can be further filtered according to the health status of any specified function of interest. These tools were then used to identify relevant parameters for hazards and identifying interesting ranges suitable for subsequent series of automatic simulation runs.

The automated FFIP simulation solved many algorithmic and technical challenges related to the generation and simulation of various valid configurations. A sensitivity analysis to discover aspects of the design having the greatest impact on reliability was made available through this work. The extended FFIP framework results evaluated how the results of the FFIP analysis are impacted by changes in model parameters and the timing of critical events.

3.5.2 Using a Functional Failure and Flow State Logic Reasoning Methodology

The Flow State Logic (FSL) method as a means for reasoning on the state of EMS flows allows the assessment of failure propagation over potential flows that are not considered in a functional representation of a nominally functioning design [10, 26]. Their work asserts that when failures are modeled to propagate along energy, material, and signal (EMS) flows, a nominal-state functional model is insufficient for modeling all types of failures. To capture possible failure propagation paths, a function-based reliability method needs to consider all potential flows, and not be limited to the function structure of the nominal state.

Configuration changes and environmental factors were identified with having the ability to cause a functional model, as designed, to no longer represent a system in its failed state accurately. However, a technical challenge being faced by design engineers while performing failure tests include the difficulty of analyzing potential failures that propagate along unknown or unintended paths and the assessment of effects of failure propagation on other elements of a system. In order to address this challenge, modelling the interactions between design elements and the effects of failure propagation along all potential paths needs to be formally analyzed. A liquid fueled rocket engine system was chosen as case study for this work. The system was modeled using a failure identification and propagation analysis framework, and then the Flow State Logic (FSL) methodology was integrated to the FFIP.

During the FFIP framework analysis, the Function Failure Logic (FFL) reasoner was used to capture the health of functions embodied by components; however it does not capture the state of flows between components. The paper discusses the implementation of the FSL methodology. First, the state of the designed and potential EMS flows in a system can be classified using this methodology. Second, the ability to map the failure propagation along the non-nominal paths provides a way of analyzing failure scenarios that introduce new EMS flows to the system. When combined with other function-based failure propagation methods, FSL provides a complete representation of the system state. The basis for this method is that EMS flows exist both as designed and as possibilities.

It is necessary to distinguish between designed flows and non-designed or potential flows. Non-designed flows are the cause and/or effect of certain failure events. To capture the possibility of failure propagation of these potential flows, the Flow State Logic reasoner identifies the state of any flow in the system of interest for any given system state.

The logic of both FFL and FSL operates on the inputs and outputs of component behavioral models called ports. FFL and FSL read port values to determine function health and flows respectively. The FSL reasons on both the designed and potential flows in a system. To accommodate the addition of the FSL reasoner, the behavioral model of a system was made to incorporate two features. A model element that corresponds to the environment around the system is created as a block. This environment block is the source for new flows created during critical scenarios. The behavioral model is created by establishing a relationship between the component behavior mode and the propagation characteristics of a flow. The types of flows used in this method are the secondary level of flow as specified by the Functional Basis [27, 28].

For each component, the behavioral model is created by defining the relationship between designed input and output EMS flows based on component mode. Then for each type of potential flow that is considered, a designer specifies the critical level at which a component mode would change. If a critical level exists, then the component mode change is specified. This work concluded that a model of a system in a failed state may not exactly match the model of the system as it was designed to operate. If only the designed EMS flows in a system are considered as the paths for failure propagation, then failures propagating along new or different flow paths will not be captured.

3.5.3 Model-Based FFIP Using Hazards

The development of a model-based failure identification and propagation (MFIP) framework was done in order to identify early potential safety issues due to undesirable interactions between subsystems and components, and failures due to environmental factors within a complex avionic system design [29]. MFIP maps hazards and vulnerability modes to specific components in the system and analyzes failure propagation paths. This provides an automated means for system designers to detect multiple and cascading failures that are not limited to component interactions.

Hazards was defined as potential sources of energy, material, and signal that cause harm and constitute deviations from the intended design or function. These hazards are results of undesired interactions between components or environmental impact on the system. An example of unsuspected hazard comes from the sources and propagation paths of stored energy in electrical, chemical, or mechanical form. For any particular domain in a complex system, expert judgment is required to expand the types of hazards.

Using hierarchical hazard types for reference, the hazard ontology was created for the EPS design to identify failure scenarios. The ontology contained hazard properties defining the types of hazards a component transmits, the types of hazards generated by a component, and component vulnerabilities to existing hazards. The fundamentals of hazard-vulnerability pairs and propagation path identification [30] and hazard ontology were used to expand design failure analysis. While using the System Modeling Language (SysML) and XMISearch tool for scripting hazardous scenarios, a satellite electrical power system (EPS) was used as case study.

System Modeling Language (SysML) [25, 31], a graphical modeling language for systems engineering applications, was used to specify, analyze, design and verify requirements, structure, and functional be havior of the system. SysML was created as an extension of the Unified Modeling Language (UML), improved for systems engineering. It provides system engineers with a standard taxonomy of diagrams in two main categories of requirement and structural diagrams. These diagrams provide ontologies and component connection models for identifying and investigating system functions, threats, and safeguards.

The EPS requirement diagram enables designers construct a system and model safety requirements from a text-based specification document in order to identify the relationship between constraints. The diagram also traces specifications to model elements, track model elements that satisfy a particular specification, and verify whether each model element fulfils requirement. The EPS block definition diagram (BDD) that describes the internal system structure, connects components and defines properties, operations, relationships, hazards, vulnerabilities, and transmitted entities.

The BDD is derived from the requirement diagram, which is also derived from the system specification document. The construction of the BDD diagram is based on each component, and decomposition established in the general failure analysis methods are naturally reused. In the BDD, the default hazard, vulnerability, and transmitted risks are associated with each component by the using the hazard ontology, which provides a structure for matching hazard and vulnerability types with each component in the system. The BDD highlights the fault propagation between components by describing the flow ports and the state of the flow.

In the block definition diagram, all components and connections were associated with the hazard carrier type. Using a path analyzer, XML Metadata Interchange (XMI) file, the hazard types were compared with specifications of each component. When components could not mitigate the effect of failure, it gets propagated to the next component or connection otherwise the proposed path analyzer deems the specific hazard as resolved. The XML Metadata Interchange (XMI) file enabled quick and easy hazard path analysis through a java-based application called XMISearch.

An evaluation of the design architecture and identification of potential hazards in assisting system designers to modify designs to mitigate identified safety issues was carried out. The process involves an iterative approach, where each cycle is repeated until no hazard is detected by the algorithm. In addition, the framework uses the function failure logic (FFL) similar to the one in use by the FFIP framework. This captures the impact of faults, based on the identification of hazard propagation paths and provides a logical assessment of the impact of component level failures at the functional level. The FFL was utilized to create a relationship between the identified failures and the health of functional elements to provide additional failure information. In order to integrate the FFL reasoner into MFIP framework, each input and output flow ports in the BDD diagram was evaluated for identified vulnerable components.

EMS flows need to be analyzed to investigate failures and impacts on system design. For a large number of failures such as explosions, leaks and operating environment, the functional model representations of the system being considered do not include the EMS flows that occur during system failure. This work transformed requirement and hazard information in enabling the investigation of system interactions and identification of hazard scenarios.

3.6 FFIP Results Analysis Research [21, 32, 40, 57-62]

This section explores different researches relating to how FFIP results are obtained and how they are being applied.

3.6.1 Applying Fault Propagation Analysis on Cyber-Physical Systems

Papkonstantinou et al [32] addressed the limitation of FFIP simulation results as only being specifically applied to a particular component model without the exploration of the impact of alternative modeling choices on such results. The limitations of utilizing the FFIP methodology during design to evaluate reliability rather than discovering more robust design alternatives were also considered. It was recommended for the FFIP component model to incorporate the capabilities of describing variation in design and the analysis of specific variants.

Solutions to the identified limitations required formal semantics and syntax for describing design alternatives while supporting their automatic configuration and analysis through software tools. The application of software configuration technology into the FFIP simulation model were considered through approaches such as incremental software development [33], software product lines [34], stepwise refinement [35], feature oriented programming and aspect oriented programming [36]. The purpose of selecting these methods ranged from improving the maintainability and scalability of codes and supporting the coordination of several software developers to mass customization of software products for different clients [32].

Feature modeling which underlies feature-oriented programming was used to describe possible options available to customers. Feature modelling is a generic technique for describing variability that can be equally used to describe design alternatives that are subjected to reliability analysis [37, 38]. The choice of feature modeling was done as it is scalable to complex applications and it supports further work for achieving full automatic reliability analysis of design configurations. A formal logic syntax help express unreliable configurations and a restricted feature model describing the reliable set of design alternatives can automatically be reverse engineered [39].

The results of the FFIP framework applied to the boiling water nuclear reactor (BWR) [23] with the added extension of a set of cyber-physical design alternatives was used as case study. A description of the system functions was given and the failures in the steam outlets of the reactor were selected for investigation. Under the automatic control system, there are two subsystems that are mandatory: coolant pump control and turbine protection. The coolant pump control has a mandatory feature: an algorithm for dropping the pumps' rotations per minute (RPM) to a minimal level. For implementing this feature there are three alternatives (step, decay and ramp); the semantics of alternative features are that exactly one of them must be chosen. Each of these alternatives represented a software feature. The choice of software control algorithm for meeting conflicting requirements relating to the nominal operation of BWR was the first set of design alternatives subjected to the FFIP analysis.

The initial step taken in the FFIP-based design created a single functional model specifying the desired functionality in an implementation independent way. The proposed methodology for identifying reliable design alternatives for the functionality was described using a flowchart. The design alternatives in the feature model were then implemented as behavioral simulation alternatives in the Simulink environment. The FFIP simulation with the extended capability to incorporate a description of design alternatives and main feedback loops of the process was developed from first principle.

The iterations were chosen by following the processes described in the flowchart for each valid configuration of the feature model. Combinations of design alternatives are then specified in order to subject a configuration to safety analysis. FeatureIDE which is open source was used with an extension to export Matlab model configuration script that automatically creates the FFIP simulation corresponding to the choice of features. A Function Failure Logic (FFL) within the FFIP simulation identified degradation and loss of function health in the functional model as simulations were run by monitoring the input and output flows. As some failures do not compromise the overall reliability of a system, using this approach, the functionality and not just the individual component failures is used to assess designs. The inputs for analysis were a set of design alternatives with the results expressed as a restricted set, from which unreliable alternatives had been removed [32].

The results of the simulations for the entire feature model were obtained automatically. The results showed that the ramp algorithm had the most reliability which conforms to the linear ramp being broadly used in boiling water reactors worldwide [32]. Hence, this showed that complex system risk analysis could be obtained from the conceptual level detail of the FFIP simulation model. These results can be passed on as information sources to the detailed design phase. This helps designers using FFIP to specify system structure and behavior at an arbitrary level of detail in order to achieve the best balance between early risk analysis and meaningfully detailed simulations.

3.6.2 Simulation Based Machine Learning for Detecting Faults

Papkonstantinou et al [40] continued efforts in applying the FFIP framework into improving fault detection in complex systems. A simulation based framework was utilized to identify a large number of faults, when there are no adequate historic data for training. An extension of the FFIP was used to generate training and testing data sets for developing fault detection systems based on data driven machine learning methods. This was done to support a simulation based framework for training and testing alternative machine learning based methods for fault detection. The case study used in this research was a generic nuclear power plant.

Machine learning studies algorithms that help computers to learn from data [41]. Significant advancements have been made in machine learning research since the development of the first artificial neural networks [42]. This is due to increases in the availability of computing power and the development of new methods [43] with engineering applications. In this work, particular emphasis was made on data-driven quantitative machine learning methods for fault detection, such as artificial neural networks and decision trees [44]. Without utilizing the knowledge of structure or logic of a system, these methods are “trained” to detect faults using data sets with system variables that describe system behavior. The sources of the data sets are either historical process data from real-life scenarios or data generated from simulated models. When any of the machine learning methods is trained to give good results, they are then evaluated with a “test” data set.

The data-driven quantitative fault detection techniques used were the artificial neural networks (ANNs) and decision trees. For accurate fault identification, these techniques require being trained and tested with process data. The data used for training and testing were generated by process monitoring of key variables such as pressure and temperature in the presence of faults. For example, in the case of single faults, successful training and testing of more than one entry in the data set were. This information was obtained due to multiple occurrences of the fault, or by performing multiple simulations per fault using different simulation parameters.

The FFIP functional failure results were used to generate training and testing input data for process history based quantitative fault detection methods. The data sets on the component faults to be detected by the fault detection system were prepared using information from the FFIP models. To increase the size of the data set, faults were simulated multiple times, using different process parameter. The functional health results for all the simulations were then used to compile the training and testing data sets. The Configuration Flow Graph from the FFIP was simulated for every pair of fault and process parameters. The simulation provided a time series of the monitored process variables (e.g. temperatures, pressures, flows). The simulation results were used by the FFL to generate the functional health results.

The functional health results gave three statistical values per monitored signal connected to the FFL. The Functional Failure Logic (FFL) for every function of the process used one or more signals from the simulation results, as well as steady state reference values for these signals, to calculate the functional health result. These values were the maximum positive deviation from Steady State (SS) average divided by the SS average, the maximum negative deviation from the SS average divided by the SS average and the maximum deviation of average signal value from the SS average divided by the SS average. All the functional health result values are expressed as percentages with the he higher percentages indicating high impact to the function in the simulation scenario. The functional results for every function of the functional model were serialized and classification attributes were added.

From the nuclear power plant model used as case study, a set of 116 automation components (primarily valve and pump actuator controllers) were selected to obtain potential faults. Three failure modes were chosen for each automation component type (e.g. a pump actuator controller can be triggered to the “failed stop”, “failed start” or “no electric supply” failure modes which results in stopping, starting or stop controlling the pump). The combinations of the 92 detectable faults and the 11 power levels (a total of 1012 simulation scenarios) were used to create the data sets for developing the fault detection systems. The simulations were performed using the Simulation Server component of Apros 6, developed by VTT [44].

A software tool was developed to parse all the simulation result files (a total of 1012 files) generated by the simulation server. The result from the 1012 entries gave 111 functional health results generated by the FFL (related to 37 monitored signals) and the classification attribute. This data set was split to create the training and testing data sets. Six power plant FFL outputs were used to build the training data set while five power plant FFL outputs were used for the testing data set. The WEKA tool’s multi-layer perceptron ANN [45] and a decision tree were used to train and test the fault detection systems. WEKA is a tool developed by the Machine Learning Group at the University of Waikato which contains a set of machine learning algorithms for data mining applications. The feedforward ANN used its input and output layers to train data using back propagation with momentum [45]. The decision tree was based on the J48 algorithm, an open source implementation of the C4.5 algorithm [46]. The training was for identifying 92 possible faults and locating 9 possible locations within the system.

The results showed 64% accuracy in fault detection using ANN and 82% accuracy while using decision tree for machine learning. For detecting fault locations, the artificial neural network had a 93% success rate while the decision tree had 97% success rate. However, ANN took longer to train than the decision tree, while the speed for testing was fast for both. It was also concluded that the decision tree provided a readability advantage.

3.6.3 Hierarchical Functional Fault Detection and Identification

The Hierarchical Functional Fault Detection and Identification (HFFDI) system for fault identification in multiple fault scenarios was developed for complex mechatronic systems [21]. HFFDI is based on machine learning techniques, commonly used as a basis for Fault Detection and Identification (FDI) systems, and the functional system decomposition of the FFIP framework. The HFFDI was designed to identify multiple faults in multiple fault scenarios using only single fault data sets for training and testing. It combines a plant-wide FDI system and a set of function-specific FDI systems.

Machine learning algorithms application is very important in the classification needed for the development of Fault Detection and Identification (FDI) systems. These support systems monitor the status of a system and try to determine the presence of faults based on past examples. "Qualitative models and search strategies" includes methodologies which use non-quantitative system models, such as fault trees and topographic templates of expert knowledge [44].

"Quantitative model-based" methodologies utilize system models to analytically detect abnormal behavior and then decision rules help in fault identification [47]. The third category, "Process history based methods" are developed using data sets with examples of process signals in fault situations. These FDI systems can be qualitative, such as expert systems, or quantitative, such as Artificial Neural Networks [44].

The HFFDI system was developed and tested using a generic Nuclear Power Plant (NPP) model as case study. The NPP model was built with the Apros 6 first principles dynamic process simulator [48], developed by Fortum and VTT Technical Research Centre of Finland. A modular method was used in this work as the FDI systems run in parallel and independently. This allows flexibility in using different machine learning algorithms for different functions (ANN, decision trees etc.).

The method in this research allowed the use of use any data-driven quantitative fault detection technique, such as artificial neural networks (ANNs) or decision trees. These techniques were trained and tested using previous fault data sets gathered through simulation or historical sources to identify faults. The training and testing data set contained entries of simulation values for the plant signals and the “class” representing the failure mode of the simulation scenario (if no failure mode was present, then class is the “No fault”).

FFIP was utilized to determine the functional decomposition of the system. The use of function-to-component mappings of the FFIP framework to generate function-specific training and testing data sets was a new research component that was introduced. The function-specific data set contained component faults relevant to a function, while other types of faults were replaced with “fault in other function” class. The data sets were then used to train the function-specific FDI systems. Each function-specific FDI system was trained using a custom version of plant-wide training data set. A written software utility was used by the function-to-component mapping of the FFIP algorithm and the plant-wide training and testing data sets to automatically generate all the function-specific data sets. The results of these FDI systems were used in addition to the plant-wide FDI results to provide the final HFFDI fault prediction result. The functional health results were used to identify faults and track failure propagation.

A high level functional decomposition of the generic NPP model was carried out to obtain 17 functions. A set of 116 automation components, mainly valve and pump actuator controllers, were used to manifest the list of faults. Two failure modes were selected per component type (e.g. “failed open” and “failed closed” failure modes for the valve actuator, which result in fully opening or fully closing the valve). Faults were identified by component name – failure mode name pairs (e.g., “Valve A” – “Failed Open”). A fault list of 232 total failure modes (116 components x 2 failure modes per components) were developed but only 92 failure modes had an effect on the NPP model when running at steady state. In all, only 84 faults were selected for detection by an FDI system for this research. A Software code was also written to produce the training and testing data sets, following the methodology presented in [40].

The fault detection results of the HFFDI system in single and multiple fault scenarios were compared to a plant-wide only decision tree based FDI system, similar to what was developed in previous research [40]. The fault detection accuracy result of the HFFDI system gave better outputs than the simple plant-wide FDI system when 510 entries training data set and 425 entries testing data set was used. Since the difference between the two systems was small, a more extensive comparison was needed to determine whether there is a significant advantage for the HFFDI system.

The data set for 11 plant power levels was used for performing an 11 fold cross-validation [49] of the plant-wide FDI system alone and of the HFFDI system (a combination of the plant-wide and the function-specific FDIs). The results from this validation showed that the HFFDI system had a minor accuracy gain, 1.4% on average, over the plant-wide only FDI system. However, the standard error based on the results of the 11 fold validation did not allow this gain to be conclusive.

For fault detection in multiple fault scenarios, similar the training data sets with the single fault scenario case (i.e. the FDI systems are not trained to identify the combination of faults, but are trained to identify single faults) was used. These testing data sets were built by selecting a fault per function (for the 17 system functions of the case study) and the resulting set of faults was used to create combinations of two and three faults. These fault scenarios were used to test the multiple fault identification capability of the HFFDI system and compare to a plant-wide only FDI system for two fault scenarios (a total of 136 scenarios) and for three fault scenarios (680 scenarios). The results showed that in two fault scenarios the HFFDI was able to identify one of the faults with 79% accuracy and both faults with 13% accuracy. In three fault scenarios, the HFFDI was able to identify one of the faults with 69% accuracy, two faults with 22% accuracy and all three faults with 1% accuracy.

3.7 Human Applications of FFIP (Socio-Impact) [50, 52-55]

This section is focused on researches involving the human interface with FFIP application

3.7.1 A Feasibility Study of Humans Computing Failure Scenarios

Arlitt et al [50] worked on a social component of engineering design in addressing how a distributed group of non-expert humans can outperform a brute force algorithm handling a failure scenario prediction task in tools such as FFIP. Human computation is a problem-solving paradigm that works well for problems that are computationally impossible [51].

Potential component failures in systems are commonly identified through expert opinion. However, many experts miss a wide range of unprecedented failures. Computational approaches on the other hand are often limited by the availability of historical data and the static encodings of expert knowledge. In general, expert analysis provides quality information at the expense of speed and breadth, while computation offers speed at the expense of quality. As result, a need for failure analysis techniques that improves either speed or quality without harming the other is needed for reliable system designs.

Arlitt et al [50] explored the possibility of applying human computation to failure analysis problems by examining non-expert reasoning about an abstracted complex system. A human computation approach to failure analysis was carried out using the reasoning abilities of non-experts in identifying failures based on abstracted system information. This is contrary to the popular approach of encoding knowledge as a set of heuristics into a failure analysis algorithm. Using this approach, a computer would perform an analysis task, and the human would interpret, synthesize, and iterate on the results.

The possibility of using human computation to augment experts' identification of new failure mode classes using existing simulation tools were investigated. For scenarios involving multiple simultaneous component failures, subsets of individual failures differentiated the failure scenarios. Hence distinct failure scenario classes emerged based on those subsets. The intuitive problem solving abilities of non-experts, for the purpose of identifying a variety of critical failure scenarios were explored.

Combinations of failures were identified in two ways; one by non-expert human subjects, and one at random. The results on the advantages and disadvantages of utilizing both were then compared. The FFIP framework, a function-based fault propagation framework used to quantify a system's functional health was utilized by the subjects as the failure analysis tool.

The software framework used consisted of a random critical event scenario generator, a simulation server, and a simulator that runs the process and automation model of the system under test. The simulation server acts as a proxy that can accept multiple requests for simulations, set up the critical event scenarios, run the simulations, and return the simulation results. The simulation request originated from human users and the automated scenario generators.

A group of 14 mechanical engineering graduate students at Oregon State University were provided with a simplified diagrammatic representation of a nuclear power plant. A brief introductory lecture on failure, FMEA, and FFIP, was provided as the only source of formal failure analysis training to the students. The students were instructed to individually generate potential failure scenarios containing exactly six simultaneous component failures. Scores were assigned to each student upon submitting a scenario for simulation indicating the level of damage severity with higher scores corresponding to more serious failure scenarios. Based on these scores, students were able to judge the relative utility of different failure scenarios, and iterate upon their failure mode scenarios. The scores by students were then compared to those of a Monte Carlo algorithm's scenarios.

It was discovered that the Monte Carlo algorithm did not outperform on the level of humans in identifying failure scenarios, suggesting a baseline of technical feasibility. Also the solution space of potential failure scenarios was sufficiently large, and the complex system simulation was sufficiently expensive. This highlighted the potential for beneficial synergy between failure mode analysis and common sense reasoning skills typically leveraged in human computation. The results obtained indicated that while non-expert reasoning may not be directly applicable to effective exploration of many possible failure modes, human computation has the potential to augment or compete with stochastic algorithms in a complex systems failure analysis context.

3.7.2 Using Simulated Failure Models for Risk Assessment

Nikula et al [52], acknowledged the lack of system representation supporting the study of hazards due to interactions between systems and their environment. Most previous works conclude that a complete system analysis can only be as detailed and informative as the simulation models. Three identified weaknesses of the FFIP method were tackled to address this issue.

As most components or subsystems exhibit broad ranges of deviations from design intent, it was identified that applying FFIP without accounting for this weakens the effectiveness of the tool. Qualitative flow state values in FFIP, such as no flow, low flow, or high flow are currently being used to model these deviations [53]. However, there had been no discussion on the set of qualitative values necessary to capture every deviation from the design intent.

HAZOP was integrated into the process of building the FFIP models in this work. HAZOP which already is an important risk analysis method in many industries and processes can efficiently analyse the hardware and software components of systems [54]. The HAZOP analysis is usually carried out by matching a set of guidewords to attributes of the design representation and interpreting those combinations as hazards. The analysis was applied to the flows in the FFIP behavioral simulation, based on the functional basis taxonomy. The guidewords were applied as the possible flow state values. Using HAZOP, the results were introduced into the FFIP simulation model as the possible deviations from design intent.

For the purpose of discovering new hazards, a designer can only build the models on limited foreseeable hazards. The authors identified a second weakness of FFIP as the inability of behavioral simulations in handling component failures due to abnormal process conditions. This was addressed by adding a capability of automatically transitioning components to failure modes while responding to abnormalities in system operations in the FFIP framework. This was done by integrating the FFIP simulation model building process with HAZOP.

The third weakness Identified was the need for the person performing FFIP analysis to be an expert on system domain, operating environment of the system, risk analysis, simulation and modeling techniques, and algorithms. The authors addressed this limitation by helping to define an information system that incorporates workflows with data models to help individuals from various backgrounds can work using the concepts from their fields of expertise.

The case study used in this research was a simplified boiling water nuclear reactor including a feed water line and emergency water line. The FFIP framework was used to model interactions between the system and its environment using an environmental flow graph (EFG) and the configuration flow graph (CFG). Failures were identified using combination of the HAZOP guide words and functional flows. The usefulness of HAZOP was manifested in capturing deviations that can later be incorporated into the simulation models.

Since both the FFIP and HAZOP methodology are normally applied at stages where high-fidelity models of the system are not readily available, the flow values were discretized into a set of qualitative values. The selection of the FFIP framework and HAZOP were done as HAZOP has proven to be effective in identifying deviations from design intent in the design representation of FFIP. Using both the FFIP and HAZOP permits interfacing between tasks for different experts: the domain safety expert, information modeling expert, and the simulation expert.

3.8 Software Implementations in FFIP

From the research discussed above together with on-going works, a number of software tools have been used to successfully deploy the FFIP framework. Examples of the research tools include Python, MADE, Matlab, Simulink State machines, LabVIEW, Modelica, SysML and so on. This tools help in describing the function, structure and behaviour of inherent components within the complex systems to be analyzed. The results of the FFIP analysis carried out have also been evaluated using most of the tools listed above.

3.9 Conclusion

This chapter contributes a novel grouping for function-based failure analysis research methodologies that are available for the research community. The different existing applications of the FFIP framework and history have been reviewed. The variation in its applications stems from the implementation of the framework to systems with different levels of complexities as well as the component types (software or hardware) in systems.

The ability to conduct either quantitative or qualitative analysis using FFIP depends on the designer's ability to detail the representation of physical components through simulation models (i.e. the Functional model, CFG, Behavior model) and the construction of an appropriate FFL. Nonetheless, the ability to identify all potential failure modes is needed to generate adequate simulation results Future works are still being pursued on FFIP applications. The impact of fidelity in the abstraction of model representation used for analysis has been investigated and presented in the subsequent chapter. The effective measure of confidence on samples from FFIP analysis is also among some of the researches currently being carried out. Early in the design stages where there is limited knowledge, FFIP is definitely a suitable failure analysis tool for prognostics and health management of systems.

<u>References</u>

[1] 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) pp. 051401.

[2] Kurtoglu, T., and Tumer, I. Y., 2007, "Ffip: A Framework for Early Assessment of Functional Failures in Complex Systems," ICED, Cite Des Sciences Et De L'industrie, Paris, France.

[3] Kurtoglu, T., Campbell, M., Gonzalez, J., 2005, "Capturing Empirically Derived Design Knowledge for Creating Conceptual Design Configurations," Proceedings of the ASME Design Engineering Technical Conferences and Computers in Engineering Conference},

[4] Wertz, J.R., and Larson, W.J., 1999, "Space Mission Analysis and Design, 3rd Edition," Space Technology Library, Microcosm, Kluwer Academic, Dordrecht.

[5] Kurtoglu, T., Tumer, I. Y., and Jensen, D., 2010, "A Functional Failure Reasoning Methodology for Evaluation of Conceptual System Architectures," 21(4) pp. 209.

[6] O'Halloran, B. M., Jensen, D. C., Tumer, I. Y., 2013, "A framework to generate fault-based behavior models for complex systems design," Reliability and Maintainability Symposium (RAMS), 2013 Proceedings-Annual, IEEE, pp. 1-6.

[7] Coatan\'ea, E., Nonsiri, S., Ritola, T., 2011, "A Framework for Building Dimensionless Behavioral Models to Aid in Function-Based Failure Propagation Analysis," 133pp. 121001.

[8] Coatanéa, E., Ritola, T., Tumer, I. Y., 2010, "A Framework for Building Behavioral Models for Design-Stage Failure Identification Using Dimensional Analysis," ASME 2010 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. 591-601.

[9] Langhaar, H.L., 1951, "Dimensional analysis and theory of models," Wiley New York.

[10] 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},

[11] Bhaskar, R., and Nigam, A., 1990, "Qualitative Physics using Dimensional Analysis," Artificial Intelligence, 45(1) pp. 73-111.

[12] Shim, T., 2002, "Introduction to Physical System Modelling using Bond Graphs," .

[13] O'Halloran, B. M., Haley, B., Jensen, D. C., 2014, "The Early Implementation of Failure Modes into Existing Component Model Libraries," Research in Engineering Design, 25(3) pp. 203-221.

[14] Tumer, I., and Smidts, C., 2011, "Integrated Design-Stage Failure Analysis of Software-Driven Hardware Systems," IEEE Transactions on Computers, 60(8) pp. 1072-1084.

[15] Valente, A., Russ, T., MacGregor, R., 1999, "Building and (Re)using an Ontology of Air Campaign Planning," pp. 27-36.

[16] Sjachyn, M., and Beus-Dukic, L., 2006, "Semantic Component Selection," Fifth International Conference on Commercial-of-the-Shelf (COTS)-Based Software Systems},

[17] Frakes, W. B., and Kang, K., 2005, "Software Reuse Research: Status and Future," 31pp. 529-536.

[18] Preece, A., 1998, "Building the Right System Right Evaluating V&V Methods in Knowledge Engineering," pp. 38-45.

[19] Pike, L., 2011, "Pervasive formal verification in control system design," Formal Methods in Computer-Aided Design (FMCAD), 2011, IEEE, pp. 206-206.

[20] Moy, Y., Ledinot, E., Delseny, H., 2013, "Testing Or Formal Verification: Do-178c Alternatives and Industrial Experience," IEEE Software, 30(3) pp. 50-57.

[21] Papakonstantinou, N., Proper, S., O'Halloran, B., 2015, "A Plant-Wide and Function-Specific Hierarchical Functional Fault Detection and Identification (HFFDI) System for Multiple Fault Scenarios on Complex Systems," ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. V01BT02A039-V01BT02A039.

[22] Papakonstantinou, N., Sierla, S., Jensen, D. C., 2012, "Simulation of Interactions and Emergent Failure Behavior during Complex System Design," Journal of Computing and Information Science in Engineering, 12(3) pp. 031007.

[23] 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},

[24] Thramboulidis, K., 2005, "Model-Integrated Mechatronics-Toward a New Paradigm in the Development of Manufacturing Systems," IEEE Transactions on Industrial Informatics, 1(1) pp. 54-61.

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

[26] 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," San Diego, CA.

[27] Wood, K. L., 2000, "Development of a Functional Basis for Design," Journal of Mechanical Design, 122pp. 359-370.

[28] Hirtz, J., Stone, R., McAdams, D., 2002, "A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts," 13pp. 65-82.

[29] Mehrpouyan, H., Jensen, D. C., Hoyle, C., 2012, "A model-based failure identification and propagation framework for conceptual design of complex systems," ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. 1087-1096.

[30] Malin, J. T., and Fleming, L., 2006, "Vulnerabilities, influences and interaction paths: failure data for integrated system risk analysis," 2006 IEEE Aerospace Conference, pp. 12.

[31] Ahmad, S. Z., 2007, "Analyzing Suitability of SysML for System Engineering Applications," .

[32] Papakonstantinou, N., Sierla, S., Tumer, I. Y., 2012, "Using fault propagation analyses for early elimination of unreliable design alternatives of complex cyber-physical systems," ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. 1183-1191.

[33] Rajlich, V., 2006, "Changing the Paradigm of Software Engineering," Communications of the ACM, 49(8) pp. 67-70.

[34] Clements, P., and Northrop, L., 2002, "Software product lines," Addison-Wesley.

[35] Batory, D., Sarvela, J. N., and Rauschmayer, A., 2004, "Scaling Step-Wise Refinement," IEEE Transactions on Software Engineering, 30(6) pp. 355-371.

[36] Apel, S., Leich, T., and Saake, G., 2008, "Aspectual Feature Modules," IEEE Transactions on Software Engineering, 34(2) pp. 162-180.

[37] Sun, J., Zhang, H., Fang, Y., 2005, "Formal semantics and verification for feature modeling," Engineering of Complex Computer Systems, 2005. ICECCS 2005. Proceedings. 10th IEEE International Conference on, IEEE, pp. 303-312.

[38] Helming, J., Koegel, M., Schneider, F., 2010, "Towards a unified requirements modeling language," Requirements Engineering Visualization (REV), 2010 Fifth International Workshop on, IEEE, pp. 53-57.

[39] Czarnecki, K., and Wasowski, A., 2007, "Feature diagrams and logics: There and back again," Software Product Line Conference, 2007. SPLC 2007. 11th International, IEEE, pp. 23-34.

[40] Papakonstantinou, N., Proper, S., O'Halloran, B., 2014, "Simulation Based Machine Learning For Fault Detection In Complex Systems Using The Functional Failure Identification And Propagation Framework," ASME 2014 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. V01BT02A022-V01BT02A022.

[41] Simon, P., 2013, "Too big to ignore: the business case for big data," John Wiley & Sons.

[42] Haykin, S.S., Haykin, S.S., Haykin, S.S., 2009, "Neural networks and learning machines," Pearson Upper Saddle River, NJ, USA:.

[43] Alexander, F. J., 2013, "Machine Learning," Computing in Science & Engineering, 15(5) pp. 9-11.

[44] Venkatasubramanian, V., Rengaswamy, R., Kavuri, S. N., 2003, "A Review of Process Fault Detection and Diagnosis: Part III: Process History Based Methods," Computers & Chemical Engineering, 27(3) pp. 327-346.

[45] Jain, A. K., Mao, J., and Mohiuddin, K. M., 1996, "Artificial Neural Networks: A Tutorial," Computer, 29(3) pp. 31-44.

[46] Quinlan, J.R., 2014, "C4. 5: programs for machine learning," Elsevier.

[47] Venkatasubramanian, V., Rengaswamy, R., Yin, K., 2003, "A Review of Process Fault Detection and Diagnosis: Part I: Quantitative Model-Based Methods," Computers & Chemical Engineering, 27(3) pp. 293-311.

[48] Juslin, K., 2005, "A companion model approach to modelling and simulation of industrial processes," VTT Technical Research Centre of Finland.

[49] Kohavi, R., 1995, "A study of cross-validation and bootstrap for accuracy estimation and model selection," Ijcai, Stanford, CA, 14, pp. 1137-1145.

[50] Arlitt, R., Papakonstantinou, N., O'Halloran, B., 2014, "Using a Feasibility Study of Human Computation for Failure Scenario Identification," ASME 2014 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. V01BT02A004-V01BT02A004.

[51] Von Ahn, L., 2008, "Human computation," Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on, IEEE, pp. 1-2.

[52] Nikula, H., Sierla, S., O'Halloran, B., 2015, "Capturing Deviations from Design Intent in Building Simulation Models for Risk Assessment," Journal of Computing and Information Science in Engineering, 15(4) pp. 041011.

[53] Kurtoglu, T., and Tumer, I. Y., 2008, "A Graph-Based Fault Identification and Propagation Framework for Functional Design of Complex Systems," 130(5) .

[54] Redmill, F., Chudleigh, M., and Catmur, J., 1999, "System safety: HAZOP and Software HAZOP," Wiley.

[55] Sierla, S., O'Halloran, B. M., Karhela, T., 2013, "Common Cause Failure Analysis of Cyber–physical Systems Situated in Constructed Environments," Research in Engineering Design, 24(4) pp. 375-394.

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

[57] Jensen, D. C., Bello, O., Hoyle, C., 2014, "Reasoning about System-Level Failure Behavior from Large Sets of Function-Based Simulations," Artificial Intelligence for Engineering Design, 28(04) pp. 385-398.

[58] DeStefano, C., and Jensen, D., 2015, "UTILIZING FAILURE INFORMATION FOR MISSION ANALYSIS FOR COMPLEX SYSTEMS," DS 80-3 Proceedings of the 20th International Conference on Engineering Design (ICED 15) Vol 3: Organisation and Management, Milan, Italy, 27-30.07. 15,

[59] DeStefano, C., and Jensen, D., 2016, "Adaptive Mission Planning and Analysis for Complex Systems," Journal of Computing and Information Science in Engineering.

[60] McIntire, M. G., Hoyle, C., Tumer, I. Y., 2016, "Safety-Informed Design: Using Subgraph Analysis to Elicit Hazardous Emergent Failure Behavior in Complex Systems," Ai Edam, 30(4) pp. 466-473.

[61] McIntire, M. G., Hoyle, C., Tumer, I. Y., 2015, "Safety-Informed Design: Using Cluster Analysis to Elicit Hazardous Emergent Failure Behavior in Complex Systems," ASME 2015 International Mechanical Engineering Congress and Exposition, American Society of Mechanical Engineers, pp. V011T14A043-V011T14A043.

[62] Jensen, D., Hoyle, C., and Tumer, I. Y., 2012, "Clustering Function-Based Failure Analysis Results to Evaluate And Reduce System-Level Risks," ASME 2012 International Design Engineering Technical Conference and Computers and Information in Engineering Conference.

[63] Uckun, S., 2011, "Meta Ii: Formal Co-Verification of Correctness of Large-Scale Cyber-Physical Systems during Design," Palo Alto Research Center, Technical Report.

[64] Tumer, I. Y., Hoyle, C., Jensen, D. C., 2015, "Validating model-based design simulation: The impact of abstraction and fidelity levels," Complex Systems Engineering (ICCSE), 2015 International Conference on, IEEE, pp. 1-6.

[65] Hunter, S. C., Jensen, D. C., Tumer, I. Y., 2016, "The Impact of Abstraction and Fidelity Levels on the Usefulness of Early System Functional Models," ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. V01BT02A018-V01BT02A018.

[66] Papakonstantinou, N., Proper, S., Van Bossuyt, D. L., 2016, "A Functional Modelling Based Methodology for Testing the Predictions of Fault Detection and Identification Systems," ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. V01BT02A015-V01BT02A015.

[67] 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},

[68] Metha, C., Jensen, D. C., Tumer, I. Y., 2013, "An Integrated Multi-Domain Functional Failure and Propagation Analysis Approach for Safe System Design," In Print.

[69] Stack, C., and Van Bossuyt, D. L., 2015, "Toward a Functional Failure Modeling Method of Representing Prognostic Systems During the Early Phases of Design," ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. V02AT03A051-V02AT03A051.

[70] O'Halloran, B. M., Papakonstantinou, N., and Van Bossuyt, D. L., 2015, "Modeling of function failure propagation across uncoupled systems," Reliability and Maintainability Symposium (RAMS), 2015 Annual, IEEE, pp. 1-6.

[71] Sierla, S., Tumer, I. Y., Papakonstantinou, N., 2012, "Early Integration of Safety to the Mechatronic System Design Process by the Functional Failure Identification and Propagation Framework," pp. do:10.1016/j.mehatrons.2012.01.003.

[72] O'Halloran, B. M., Papakonstantinou, N., and Van Bossuyt, D. L., 2016, "Cable routing modeling in early system design to prevent cable failure propagation events," Reliability and Maintainability Symposium (RAMS), 2016 Annual, IEEE, pp. 1-6.

[73] Jensen, D. C., and Tumer, I. Y., 2013, "Modeling and Analysis of Safety in Early Design," Procedia Computer Science, 16pp. 824-833.

[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] Papakonstantinou, N., Sierla, S., Tumer, I., 2012, "Multi-Scale Simulation on Interactions and Emergent Failure Behavior during Complex System Design," ASME Journal of Computing & Information Sciences in Engineering, 12(3) pp. 10001.

[76] Ramp, I. J., and Van Bossuyt, D. L., 2014, "Toward an automated model-based geometric method of representing function failure propagation across uncoupled systems," ASME 2014 International Mechanical Engineering Congress and Exposition, American Society of Mechanical Engineers, pp. V011T14A007-V011T14A007.

CHAPTER 4

THE IMPACT OF MODEL DETAIL AND ABSTRACTION ON SYSTEM MODELING

4.1 Introduction

When performing conceptual stage failure analysis of complex systems, two key questions are needed. What level of model detail is needed to make risk and safety decisions at early design stages of complex systems and how can the validity of the analysis tools enabling decision-making be characterized? This section addresses some important findings in attempting to answer these two design questions in the context of function-based analysis of complex engineered systems. The required level of detail needed by models and the abstraction level are explored in order to understand how they affect the validity of a model-based failure analysis method. The approach also supports how these systems can be designed to avoid failure.

4.2 Background

The many achievements of model-based system design include a means of providing faster system evaluation and redesign while reducing design cost, failure risks, design time and manpower. A lot of model-based analysis tools have been developed support works with this similar goal [1]. Now, using model-based systems, reasoning tools can be developed to evaluate parameters such as component performance, system performance or functional robustness in relation to the existence of different faults. Utilizing this type of analysis at the early design stage helps designers to make informed decisions before the allocation of design resources.

The accuracy to which behavior can be modelled from actual objects through abstraction and the fidelity with which the functional analysis of a system model is represented for different analyses impacts the measure of usefulness of a system model [2]. The concept of model fidelity has been established as an uncertainty source, yet it has not seen the level of extensive research as other aspects of model building and simulation [3]. Fidelity is defined as the degree of exactness of a model or simulation representation in comparison to the real world model [3]. Therefore, this research identifies and studies the various qualities and characteristics of a model which defines its abstraction level and determines how those characteristics affect usefulness of analysis using those models.

By exploring the way that functional detail and behavioral detail affect the analysis results from using the Function Failure Identification and Propagation (FFIP) framework [4-7], we form insights into what level of modelling is needed for making particular design decisions. With emphasis on system architecture, design refinement and evaluation during the conceptual design stage, this research aims to characterize behavioral and functional abstractions and their effect on the design-stage predictions of functional analysis methods. Scaling systems to include large numbers of component and subsystem interactions are some of the challenges being tackled at the early design stages of complex engineered systems [8, 9].

This work builds on the recognition of function as a means of generating system architecture and embodiment. Because function represents designer intent, functional representations are not entirely objective. Rather, there is a choice in the level of abstraction when defining a system’s functions. In this work, how the choice of abstraction affects the analysis using selected representations are investigated.

4.3 Detail and Fidelity in Functional Modeling

Functional decomposition of systems has helped explore the concept of fidelity resulting in the high-level function of the system comprising several levels of functions [10-12]. Designers need the knowledge of fidelity levels in order to come up with right models that will provide precise and accurate analysis results suitable for decision making.

Generating system architecture and product requirements has been boosted through the extensive use of function modeling. Current researches are being done to develop a formal language and syntax to improve functional modeling [2]. Most of the researches are based on defining distinct levels of detail, or abstraction, that accurately represents the intended physical system in an effort to improve the design process [10-12]. The Functional Basis framework [13] exists among these approaches and it has proven effective in the development of some failure analysis tools.

During the design process, informed decisions are constantly needed on the type of components and subsystems needed to accomplish functions required by the proposed product. The Functional Basis framework which is described by three distinct abstraction levels of functions and flows assists in making these decisions. Through this framework a component can be described using different forms of detail from simple to complex which are grouped as primary, secondary or tertiary levels.

Increase in the level of specificity achieved by using each level of the Functional Basis framework decreases the potential physical means by which that function is achieved. This results in refining the behavior of the system to a smaller and smaller set of components that can achieve the specific level of functional description. In this work, the function and flow terms in

Functional Basis will be used to define the three distinct levels of abstraction to be considered in the functional representation [13]. Function is not the only aspect of function-based system analysis that can be represented with various levels of abstraction. Behavioral representation can have an entirely different degree of abstraction.

4.4 Function-Based and Behavior-Based Failure Analysis

Over the past decade, different methods have been developed to describe and predict the undesirable performance of systems at various stages of the design process that utilize functional representations, behavioral representations or both. Most of these methods have been discussed in the previous chapters.

A key aspect of systems engineering design process is behavioral modeling. The process involves using quantitative models obtained from functional models to investigate the performance of a system relative to design requirements and specifications [14, 15]. Methods such as the function-based behavioural modeling (FBBM) permits internal iterations between the starting functional model and the end solution provided in the analysis [14, 15]. The application of behavioral models for model-based safety approaches have assisted in providing sufficient system details during failure analysis at early design stages. This approach usually requires language support for specifying fault modes and a method for introducing these modes into the nominally working system model [15].

The use of the FFIP tool has previously revealed fault propagation paths in various systems although its validity has yet to be evaluated on physical platform in assessing its applicability in functional representation of systems to making design decisions. However, in this research, simple state machines and system dynamic simulations will be used for crossevaluating the failure information obtained from behavior abstraction models and function-based models. The function failure reasoning logic will also be used as the primary logic within the FFIP simulation. The health states listed previously are used to represent the health of components of the system at given times during the simulations. This research uses the FFIP method in the evaluation of the functional health of the system. By exploring different behavioral and functional abstractions using this method, their impact on decision-making is illuminated.

4.5 Abstraction, Fidelity, and Resolution

In the scope of model generation and analysis abstraction, fidelity, and resolution often mean the same thing [2]. A model of any system is an abstraction or reduced reality of the actual system [16]. While abstraction can also be defined as the degree of separation with which a representation of a system deviates from the true system, fidelity measures the accuracy in the reproduction of a model, or a measure of the exactness of that same model [17]. Most design engineers often prefer high fidelity simulators and models for effective system analysis, leading to corresponding expensive model development. However, recent studies show that high fidelity simulators may not be as necessary in producing required results [18]. Also, high and low resolution models exist in the realm of fidelity description.

Assessing the impact of failure scenarios within models of a system at different fidelity levels is always challenging as it requires specific demands of defining such scenarios. For complex systems, it is usually a daunting task to observe and compare each component between models [18]. However, simulation results within two models of different fidelity levels can be used to show and compare on correlation plots the results of experimentation [18, 19].

4.6 A Study on Model Abstraction and Functional Analysis

To explore the relationship between modeling abstraction and its effects on functional analysis, simulation, and reasoning capabilities we focus on the design-stage failure tool, FFIP (Function Failure Identification and Propagation framework). As discussed above, the intent of using this type of tool is to evaluate the functional robustness of a system design in response to scenarios of interest to the designer. As described in Figure 1, the inputs of an FFIP analysis are: critical scenarios, functional and behavioral representations, and mapping logic between the behavior and intended function. The outputs that a designer uses are the system's functional response to the scenario and the system's behavioural response to the scenario. These results can be used in many ways to aid decision-making as described in previous work.

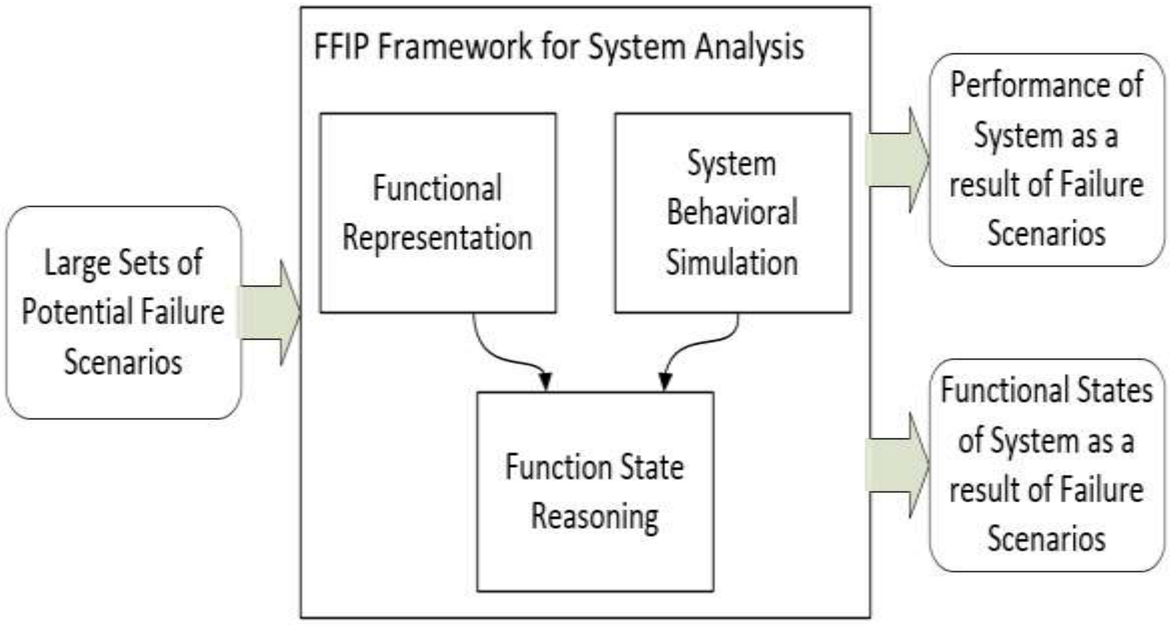

FFIP Framework for System Analysis
Functional Representation
System Behavioral Simulation
Function State Reasoning
Large Sets of Potential Failure Scenarios
Performance of System as a result of Failure Scenarios
Functional States of System as a result of Failure Scenarios

FIGURE 1: FUNCTION FAILURE IDENTIFICATION AND PROPAGATION FRAMEWORK

The interest here is to identify the role that modeling abstractions plays in affecting analysis results and, therefore, the decision-making capabilities of this tool. The reasons for selecting this tool for analysis are:

1. The current abstraction specification approach for FFIP and the set of tools related to it is ad-hoc.
2. This tool relies on abstraction in both the function-flow paradigm as well as the behavioral paradigm, allowing for the exploration of both.

In order to identify the impact of abstraction we need to establish a consistent terminology that will enable useful descriptions of the models. Table 1 summarizes the classifications chosen for this study. It should be noted that the consistent use of three levels of abstraction is arbitrary and is based on developing an experimental framework to explore the space of potential representations.

For functional representation and reasoning, we build upon the descriptions established in the Functional Basis [11]. In the Functional Basis, there are three levels identified for both functions and flows and their naming was selected to avoid implying significance of one level over another. From a design synthesis perspective, the use of functional representations is based on the refinement of the artefact at the time of modeling.

Table 4.1: DISTINCTIONS OF ABSTRACTION FOR FUNCTION, FLOWS AND BEHAVIOR

Functional Abstractions		
Primary	Secondary	Tertiary
General Principle	Specific Principle	Specific with Parameter
Example: Channel	Example: Guide	Example: Translate
Flow Abstractions		
Primary	Secondary	Tertiary
Domain	Type	Characteristic Property
Example: Energy	Example: Electrical	Example: Voltage
Behavior Abstractions		
Primary	Secondary	Tertiary
Qualitative	Discrete	Continuous
Example: Signs Inequalities, Orders of Magnitude	Example: Discrete State Machines	Example: Time-based Calculus

In order to validate a system chosen for conceptual design stage analysis, the selected abstraction level used for system modeling will have during implementation. Hence, this work utilizes a combination of computer modeling simulation packages and techniques to evaluate the impact that abstraction and model fidelity have on the validation of early-design stage failure analyses. This specifically requires conceptual creation of system simulation models at multiple abstraction levels, and conducting an FFIP analysis on these system models using Matlab-Simulink and WolframSystem Modeler.

The level of fidelity that each set of FFIP reasoning results can provide is evaluated against possible measurable output parameters from dynamic simulations which are much similar to the physical prototype tests carried out at later stages of the design process. In the scope of this work, it is assumed that the designers will follow a design process similar to that which was articulated in the previous chapter. The engineering problem will be broken down into functional and behavioral model analysis of the desired solution and eventually carried through to product realization.

The case study model for this research is an electric vehicle drivetrain. The electric vehicle drivetrain gives sufficient insights into the functional and behavioral effects of failures and their propagation. This information can later be incorporated into larger complex engineered systems which may include a team of similar vehicles operating in pursuit of mission completion (such as autonomous taxis) or include the manufacturing processes and operators that add complexity to the behavior of the system.

4.7 A Method to an Effective Failure Analysis

Once there is a decision to create a system either based on identified customer needs (for product improvement) or to meet organization's competitive goals (new product release), the identification of the proposed system's requirement comes next. It is at this stage that an effective failure analysis method, such as shown in this research, needs to be put in place to reduce resources and improve the safety operation of the system.

The following subsections will highlight the procedure on the approach to predicting faults in our case study, an electric vehicle (EV) powertrain which is a relatively new complex system within the automobile industry.

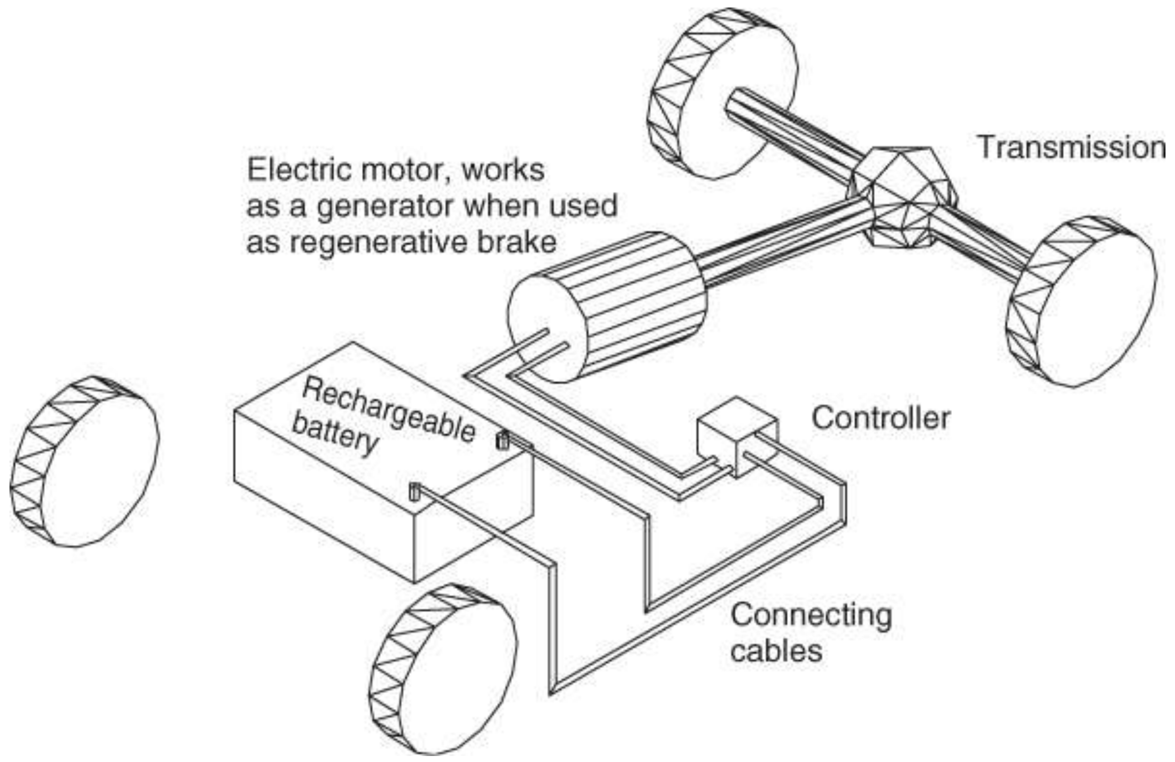


FIGURE 4.1: RECHARGEABLE BATTERY ELECTRIC VEHICLE [20]

The concept behind battery electric vehicles is simple, as shown in Figure 2 above. The vehicle usually consists of an electric battery for energy storage, an electric motor and a controller. The battery is generally recharged from an electricity supply through a plug and a mobile battery charging unit or at a charging point. The controller would be in charge of regulating the amount of power supplied to the motor, correspondingly affecting the vehicle speed [20].

4.7.1 Model: Functional

This model representation is usually done based on the information obtained from the proposed system’s components/subsystems requirement. In this work, an abstract functional model based on the general requirement for an electric vehicle’s powertrain (Table 2) was built. The electric vehicle is normally required to have the ability of traversing different road conditions, be equipped with an on-board power supply (with recharging capabilities) which can be regulated based on need, and the ability to respond to control from a driver. The EV may also be required to capture information about itself and its environment (for autonomous operations). The functions and flows for the model are generated from the primary levels of the Functional Basis [11].

In order for the EV to meet its requirements, the functional model was made to contain Provide Energy, Control Signal, Control Energy, Channel Solid Material, and Direct Signal, as shown in Figure 3. Provide Energy and Control Energy can be described in less abstract terms by moving to the secondary level of the Functional Basis. This is shown in Fig. 4 and includes functions of Supply, Transfer, and Regulate Electrical Energy. The channel material function is further defined to include Convert Electrical Energy to Mechanical Energy and Guide Solid

Material. Figure 4 also includes swim lanes to indicate the physical component types that can implement those functions.

The conversion of electrical energy to mechanical energy is required to interface between the electrical power source (the battery) and the wheels for motion. It should be noted that upon expansion of the functional model, the input and output flows to the environment, will usually remain the same. The implication is that, the energy flow going into ‘provide’ is similar to the input flow to ‘supply’ while the output flow of ‘provide’ is the output flow of ‘regulate’. This allows the sub-sections of the model to be assigned terms which are consistent with the secondary levels of the Functional Basis [11].

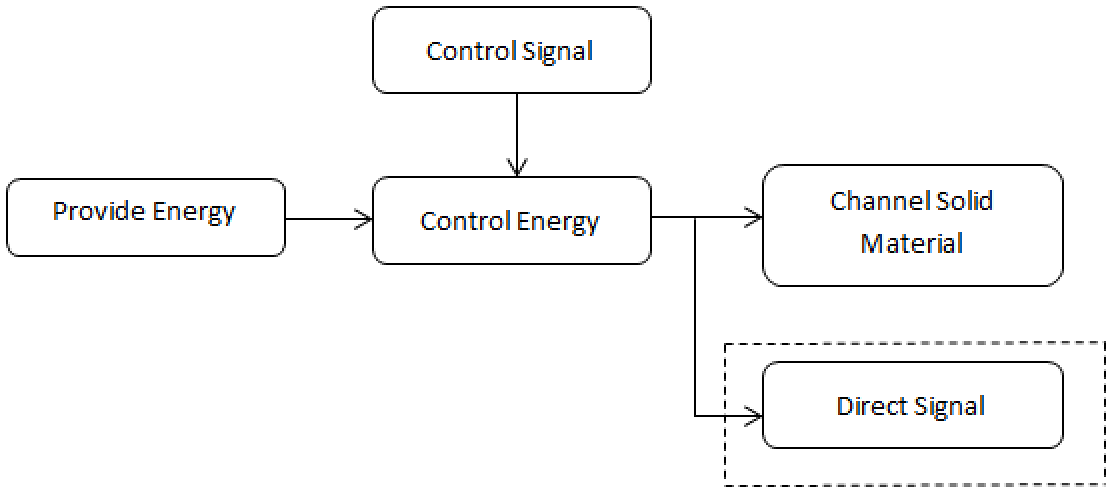


FIGURE 4.2: FUNCTIONAL MODEL OF AN ELECTRIC VEHICLE, DESCRIBED WITH THE PRIMARY LEVEL OF THE FUNCTIONAL BASIS

To understand the effect of modeling and fidelity analysis on the validation of the early failure prediction methods, an interconnected model of behaviors and functions is needed. A mapping between the functional and behavioral models should be explored to establish a consistent approach.

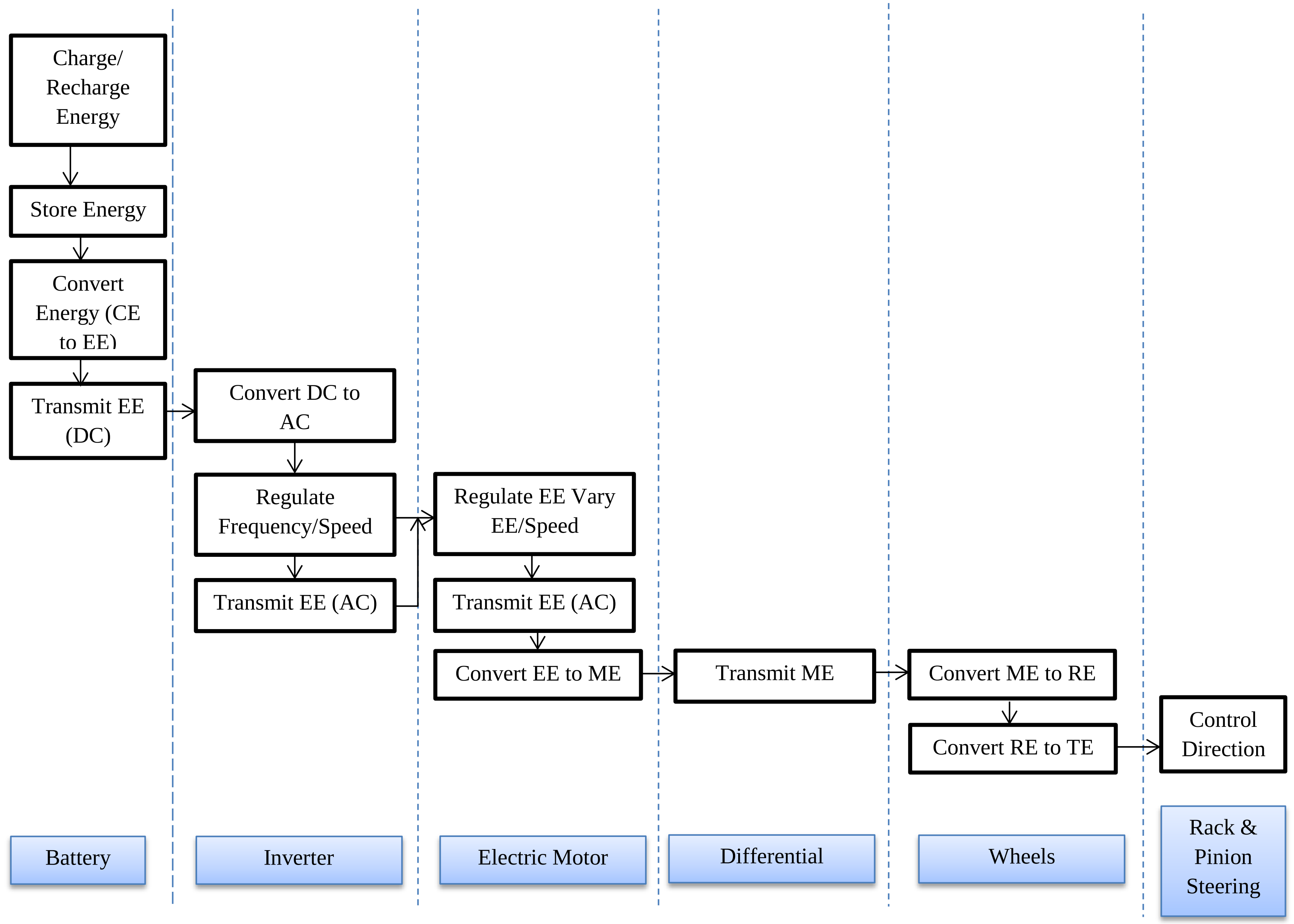


FIGURE 4.3: EXPANDED VIEW OF THE FUNCTIONS: PROVIDE ENERGY, CONTROL ENERGY, DIRECT SIGNAL AND CHANNEL SOLID MATERIAL FROM FIGURE 3

TABLE 4.2: GENERAL REQUIREMENTS FOR AN ELECTRIC VEHICLE POWERTRAIN

Battery	7100 cells of 3.7V and 3400mAh or 403V and 220Ah (after assembly)
Electric Motor	4-Pole, 3-phase Induction motor, 443lb-ft, 416Hp
Differential	Gear ratio: 4.27
Inverter	DC/AC, 50Hz ~ 60Hz
Wheels	0.1905m radius
Curb Weight	2000kg ~ 2200Kg

4.7.2 Model: Behavior

A behavioral model is a structure of connected components which shows the expansion within these components to include more fidelity as abstraction is removed from the model. It is a quantitative approach that uses physics, engineering knowledge and principles to describe the internal operations that make a component perform its task. Behavioral modeling is essentially a component-driven approach since immediately the intended functions of a system are identified, component solutions needed for the identified functionality are selected and then the behavioral models of the selected components are created [14].

When applying the mathematical equations to describe physical behavior, the description exists in certain states of abstraction from the true physical phenomena that is occurring. Describing a behavior in abstract terms may lead to over-simplifying or neglecting certain characters. In the early stages of the system design process, specific design parameters are often unknown, leading to uncertainties in the model representation which depend on the functional model development and knowledge of system parameters [19].

In this research, the behavior model of the electric vehicle was created from the functional model information. The components needed to complete a vehicle's powertrain were identified, sets of general requirements and capacities for each component were selected and physics based models on how the chosen components would behave while executing each of its functions were created. The electric vehicle components specification used for this work is given in the table 2. Using online and textbook resources, the values in the specification table are assumed only as a guide to the designer to ensure that outputs obtained from model simulations can be validated.

Due to its quantitative nature, behavior modeling is not hierarchical where certain sets of component behavior can be ranked. It is usually up to the designer to pay significant importance to the intended component behaviors desired during modeling. For example, an electric vehicle's battery will behave in different ways while performing its function of storage, transmitting, converting and recharging. All these functions would require different behavior models to represent them within the battery. However, this research will explore different levels of detail in describing a behavioral model that adequately describes complex systems.

This work recognizes that different fidelity levels can be used to model component behaviour depending on level of expertise and knowledge base available to the designer. As such we show the various types of behaviour models that can be created for an EV powertrain at the early stage of design as shown in Figure 5. Using "Function 1", "Function 2", and "Function 3" level of detail is increased in describing component's function behaviors. We will run various analyses on these models to determine what level of modelling should be considered sufficient to predict faults at this stage.

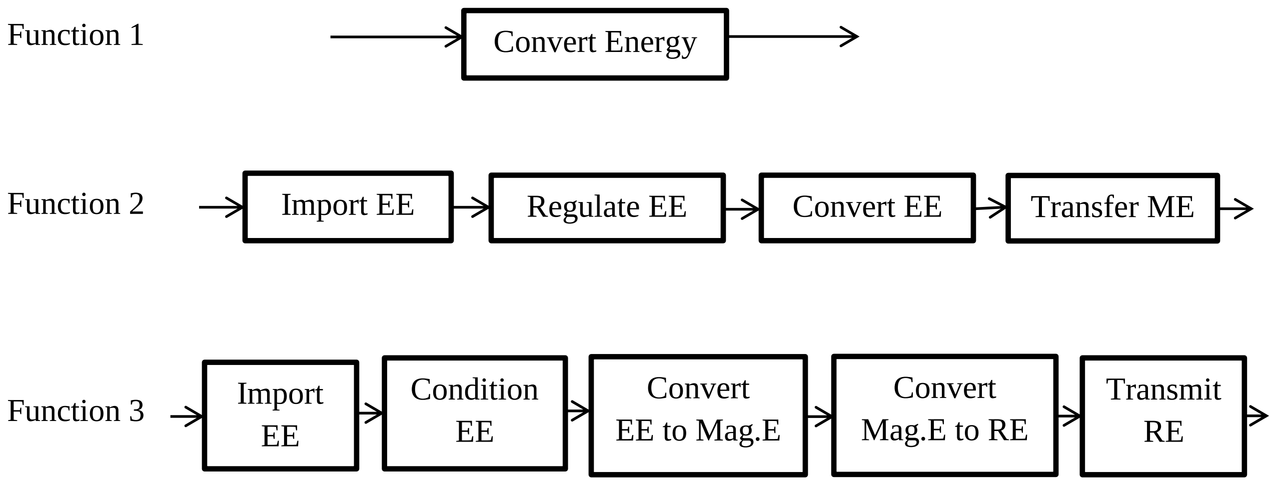


FIGURE 4.4: FIDELITY IN BEHAVIOR DESCRIPTIONS

For each behavioral module, qualitative and quantitative physics can be used in describing the behavior of the component at different modes of operation [6]. The transitions between each state refer to the function health states. The health states are logical statements that evaluate the relationship between the input and output flow of the behavioral model. The functional health is calculated using a Function Failure Logic (FFL) at each time-step of the simulation, as the flows are adjusted by component behavior descriptions. Figure 6 illustrates how information from the behavior model is passed through a reasoning model to obtain function health state of the component.

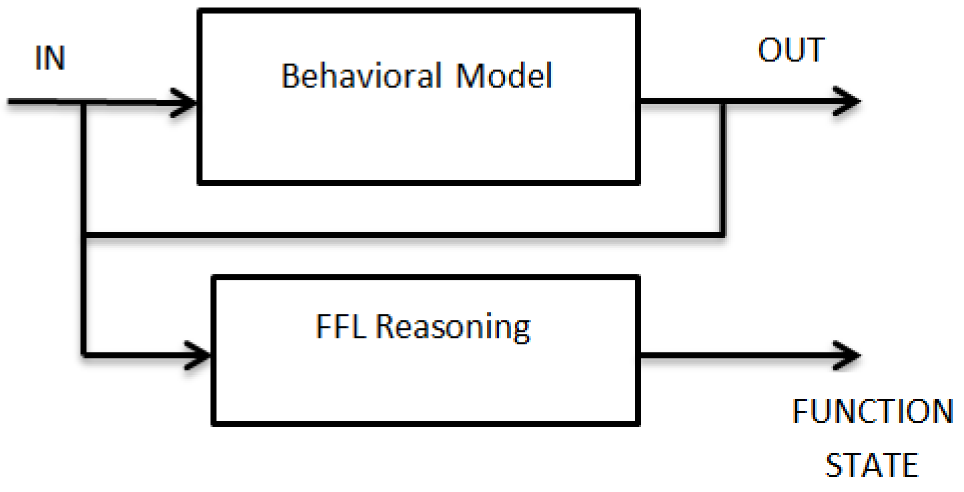


FIGURE 4.5: BEHAVIORAL REASONING AT THE MOST ABSTRACT LEVEL

4.7.3 Historical Failure Database: FMECA+ Mode Number

An important section of this research developed a method to explain the possible component faults introduced into the simulation environment. To address this, an historical failure database based on the potential performance of the individual components in the powertrain of an electric vehicle was generated. This was carried out in order to derive a classification of fault modes. The modes describe the behavior of the component in their nominal states while also adequately capturing the deviations from desired states.

Using engineering and practical knowledge, an FMECA of the electric vehicle powertrain model was constructed. This was done to include an additional column called “mode number” The mode number primarily highlights the magnitude of the effect of failure modes of the component on a number scale, where higher values indicate fault severity. This approach reflects a real world task of utilizing historic failure data as a starting point in failure analysis for our model. The information on the failure mode number of each component within the system was implemented in the behavior models to simulate function faulty states in Matlab Simulink and Wolfram SystemModeler.

The importance of this approach in this work is to bridge the gap between conventional tools (such as FMECA) and the simulation environment at early design phases. The individual failure modes are represented as mode numbers in the simulation environment. This makes the approach suitable for predicting faults at the early stages. The FMECA input addresses limitations surrounding its singular use such as being generic in application or being less considerate of the operating conditions of certain components within a system. The limitations are addressed by attaching the failure mode number of each component to a model of the actual system being investigated. The result from this work reduces reliance on arbitrary values given to probability rating, consequence rating and risk priority number (RPN). These are replaced with specific overall system performance values obtained from simulating the failure mode numbers. This crucial step helps in quantifying the actual degree of impact a failure mode has in the particular system being tested, thus reducing the uncertainty from the application of FMECA.

A model of the system's failure space from component failure information is modeled to capture various individual operational states within our system using abstract, state-based descriptions of component behavior and failure behaviour in the simulations of models. Different failure mode scenarios representing the different faults that would occur in the system were all injected into the simulations. From the failure mode database, series of single fault to multiple faults tests were introduced into the simulation environments by changing the abstract, nominally performing quantitative states value of components to correspond with faulty states. Investigations on how these faults are propagated and their paths across other interacting components thereby causing a change in operational states were carried out.

TABLE 4.3: AN EXCERPT OF THE FAILURE MODE DATA (FMECA) OF THE ELECTRIC VEHICLE POWERTRAIN

Component	Failure mode	Effect(s)	Mode Number	Cause(s)	Probability rating (1-9)	Consequence rating (1-9)	Risk Priority Number
Battery	Damaged recharging contact	No output power	3	Improper installation, wear and tear, loose connections, manufacturing faults	2	9	18
	Cell(s) damage	Reduced output power	1	Manufacturing faults, mishandling during installation, overheating	2	7	14
	Encasement/ cover impaled	Reduced output power	1	Manufacturing faults, mishandling during installation, overheating	1	9	9
	Damaged discharging contact	No output power	3	Improper installation, wear and tear, loose connections, manufacturing faults	2	9	18
	Worn-out Battery	No output power	3	Normal wear and tear	1	7	7
	Partial contacts	Reduced/ inconsistent output power	2	Improper installation, cable wear and tear, loose connections, wiring faults, manufacturing faults	2	9	18
	Over discharge	Excessive output power	4	Manufacturing faults	2	9	18
	Under discharge	Inconsistent output power	2	Manufacturing faults	2	7	14
	Overheating	Reduced output power	1	Improper installation, loose wired connections, wiring faults, manufacturing faults	2	9	18
	Overcharging	Reduced output power	1	Wear and tear, wrong connection to charging supply, wiring faults, manufacturing faults	2	9	18

TABLE 4.3 (Cont.): AN EXCERPT OF THE FAILURE MODE DATA (FMECA) OF THE ELECTRIC VEHICLE POWERTRAIN

Component	Failure mode	Effect(s)	Mode Number	Cause(s)	Probability rating (1-9)	Consequence rating (1-9)	Risk Priority Number
Power Inverter	Failure to convert DC to AC	No output power	3	No power from battery, wrong connections, windings and coil damage, manufacturing faults	1	9	9
	Failure to transfer EE	No output power	3	No power from battery, wrong/loose connections, wiring faults, manufacturing faults	2	9	18
	Damaged Inverter switch	No output power	3	Wear and tear, manufacturing faults	2	9	18
	Partial contacts	Reduced/ inconsistent output power	1, 2	Improper installation, cable wear and tear, loose connections, wiring faults, manufacturing faults	2	9	18
	Loose Connections	Reduced/ inconsistent output power	1, 2	Improper installation, cable wear and tear, loose connections, wiring faults, manufacturing faults	2	7	14
	Old Inverter	Reduced/ inconsistent output power	1, 2	Normal wear and tear, manufacturing faults	1	7	7

4.8 FFIP Simulation

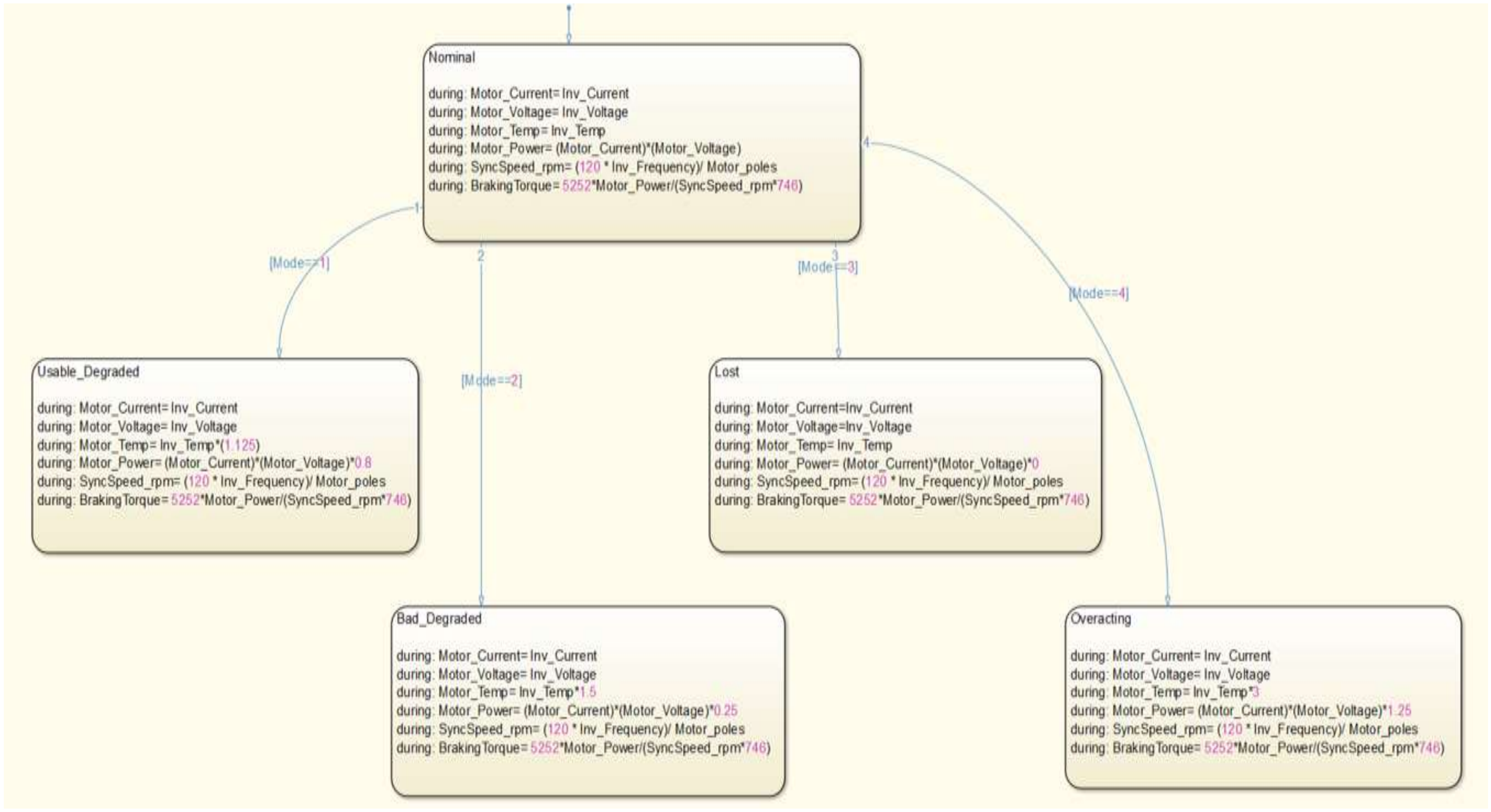
The FFIP simulation of the system uses information from the functional model and the failure database into behavior models represented as simple state machines. The models were built using MATLAB Simulink Stateflow tools. Using state machines allows discrete and continuous modelling of system components within a times simulation. It also allows an easy assessment of the impact of individual component modification.

Figure 8 shows a behavioral model of an electric motor performing a transmitting function as it exists in the detailed fidelity Function 3 described above. Inside the behavioral component model, the inputs consist of different input flows that each component needs to perform its in-built operations, as defined by the designer.

In the case of the electric motor, there are several inputs needed to be considered for operation. There are also heat flow losses which are external flows to the environment. The Figure shows the executable state machine in accurate detail. As the model is operating at a high level of abstraction, with minimal parameter definition, physics, engineering and logical reasoning are employed. For example, In the case of the electric motor behaviour models, four operational modes were considered. These modes are enumerated as.

1. Nominal - the energy out of the module is at its expected level.
2. Overacting - the module is providing more energy than expected to the rest of the system.
3. Lost - the module is providing no energy to the rest of the system.
4. Degraded - the module provides energy to the rest of the system, but is less than required.

This type of modelling and analysis was employed for all other component models in the powertrain system



```
Nominal

during: Motor_Current= Inv_Current
during: Motor_Voltage= Inv_Voltage
during: Motor_Temp= Inv_Temp
during: Motor_Power= (Motor_Current)*(Motor_Voltage)
during: SyncSpeed_rpm= (120 * Inv_Frequency)/ Motor_poles
during: BrakingTorque= 5252*Motor_Power/(SyncSpeed_rpm*746)
```

```
Usable_Degraded

during: Motor_Current= Inv_Current
during: Motor_Voltage= Inv_Voltage
during: Motor_Temp= Inv_Temp*(1.125)
during: Motor_Power= (Motor_Current)*(Motor_Voltage)*0.8
during: SyncSpeed_rpm= (120 * Inv_Frequency)/ Motor_poles
during: BrakingTorque= 5252*Motor_Power/(SyncSpeed_rpm*746)
```

```
Lost

during: Motor_Current=Inv_Current
during: Motor_Voltage=Inv_Voltage
during: Motor_Temp= Inv_Temp
during: Motor_Power= (Motor_Current)*(Motor_Voltage)*0
during: SyncSpeed_rpm= (120 * Inv_Frequency)/ Motor_poles
during: BrakingTorque= 5252*Motor_Power/(SyncSpeed_rpm*746)
```

```
Bad_Degraded

during: Motor_Current= Inv_Current
during: Motor_Voltage= Inv_Voltage
during: Motor_Temp= Inv_Temp*1.5
during: Motor_Power= (Motor_Current)*(Motor_Voltage)*0.25
during: SyncSpeed_rpm= (120 * Inv_Frequency)/ Motor_poles
during: BrakingTorque= 5252*Motor_Power/(SyncSpeed_rpm*746)
```

```
Overacting

during: Motor_Current= Inv_Current
during: Motor_Voltage= Inv_Voltage
during: Motor_Temp= Inv_Temp*3
during: Motor_Power= (Motor_Current)*(Motor_Voltage)*1.25
during: SyncSpeed_rpm= (120 * Inv_Frequency)/ Motor_poles
during: BrakingTorque= 5252*Motor_Power/(SyncSpeed_rpm*746)
```

FIGURE 4.6: A BEHAVIORAL MODEL OF THE ELECTRIC MOTOR IN AN ELECTRIC VEHICLE AT FIDELITY FUNCTION 3

4.9 FFIP Results

A number of results were obtained using the combination of models and tools in the approach discussed above.

Nominal scenario test indicates the presence of no known faults or abnormal conditions affecting the simulations while other scenarios will have certain degree of faults as obtained from the failure database.

TABLE 4.4: PARAMETERS USED IN MODELING COMPONENT BEHAVIOR

Mode Number	Function Healthy States	Efficiency Values
0	Nominal	100%
1	Usable_Degraded	80%
2	Bad_Degraded	25%
3	Lost	0%
4	Overacting Wheel	125%

4.9.1 Single Fault Scenario 1: Differential

The first scenario presented shows the results of the system simulations using different failure mode numbers for the EV's Differential failures while other components were healthy.

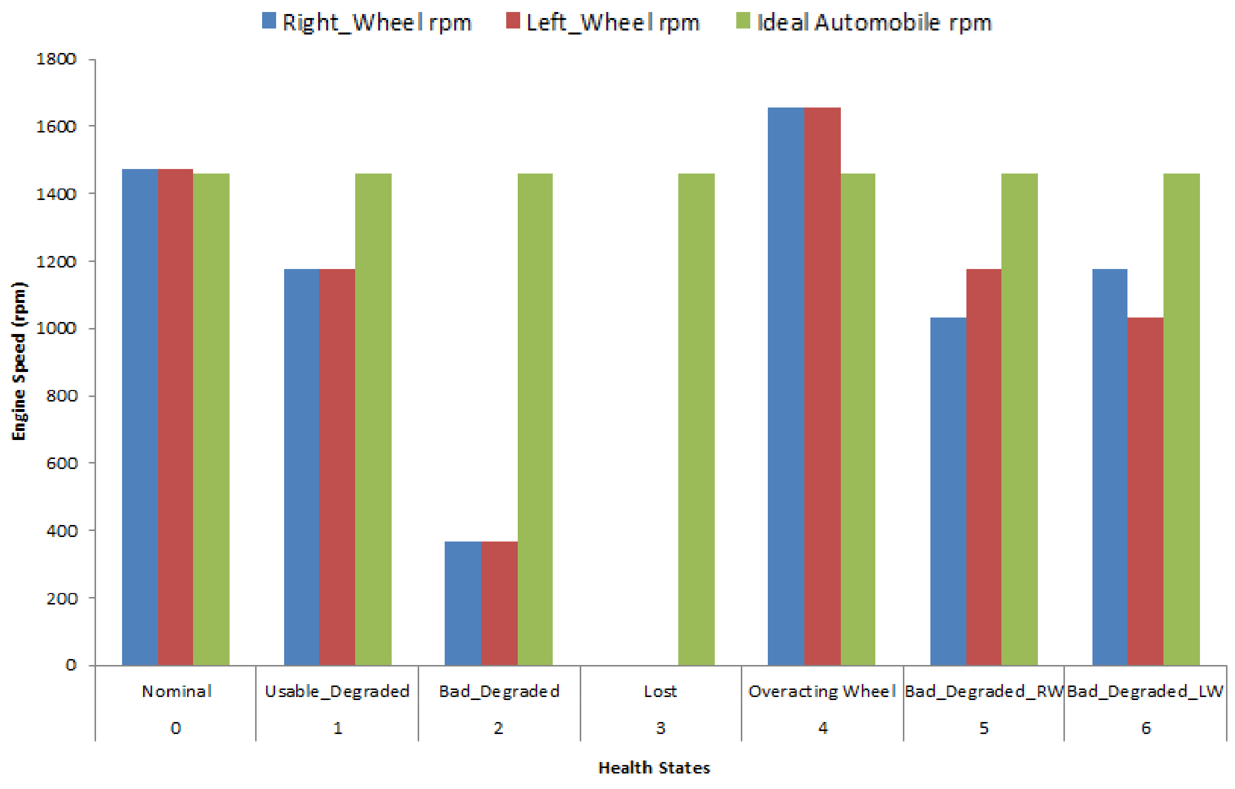


FIGURE 4.7: DIFFERENTIAL FAILURE PLOTS, ALL OTHER COMPONENTS ARE HEALTHY

For example, a left wheel torque differential output will resemble a flat tire on the left side of the vehicle. Information of the other possible deformed conditions are obtained from the failure database and implemented in the model to analyse the possible effects. Abnormal environmental conditions were not created as inputs to all the models to limit uncertainties in our predictions.

4.9.2 Single Fault Scenario 2: Battery

The scenario shows the results of the system simulations using different failure mode numbers for the EV's Battery failures while other components were healthy.

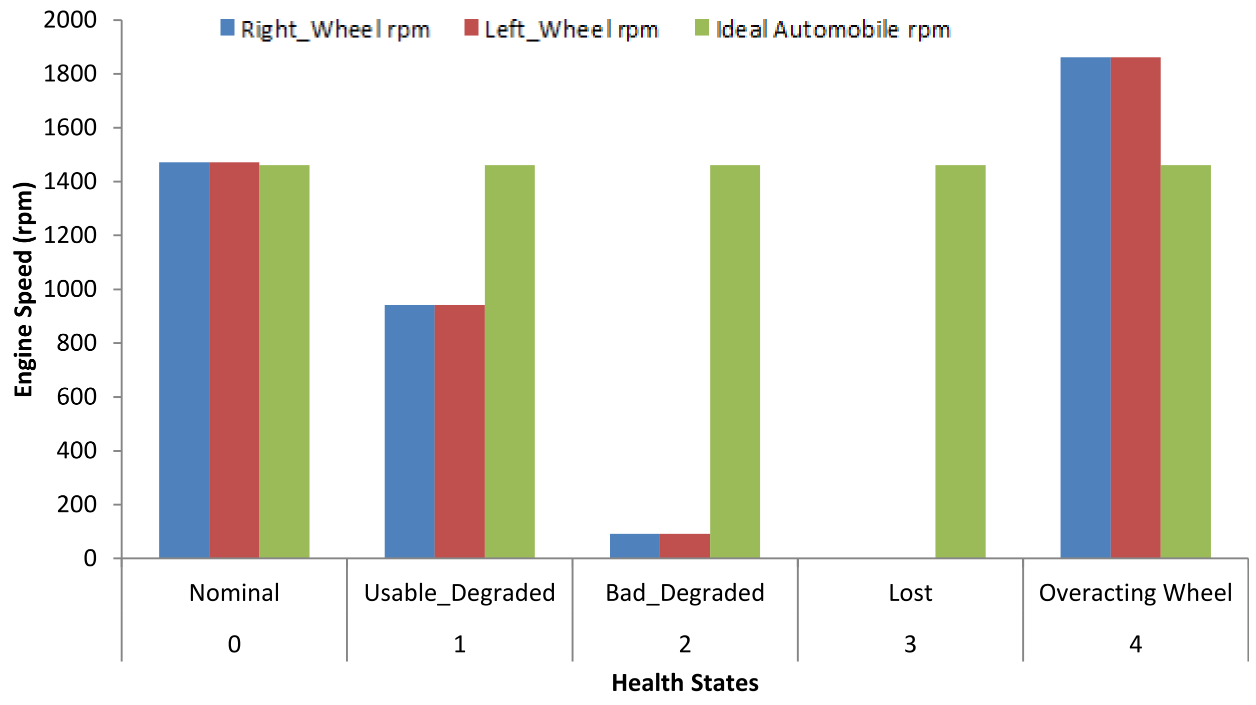


FIGURE 4.8: BATTERY FAILURE PLOTS, ALL OTHER COMPONENTS ARE HEALTHY

From the output results of the Single Fault Scenario 1 and Single Fault Scenario 2, it can be observed that there is a huge and significant drop in the wheel speed due to the battery being degraded than there was at any degradation at either wheel.

4.9.3 Double Fault Scenarios:

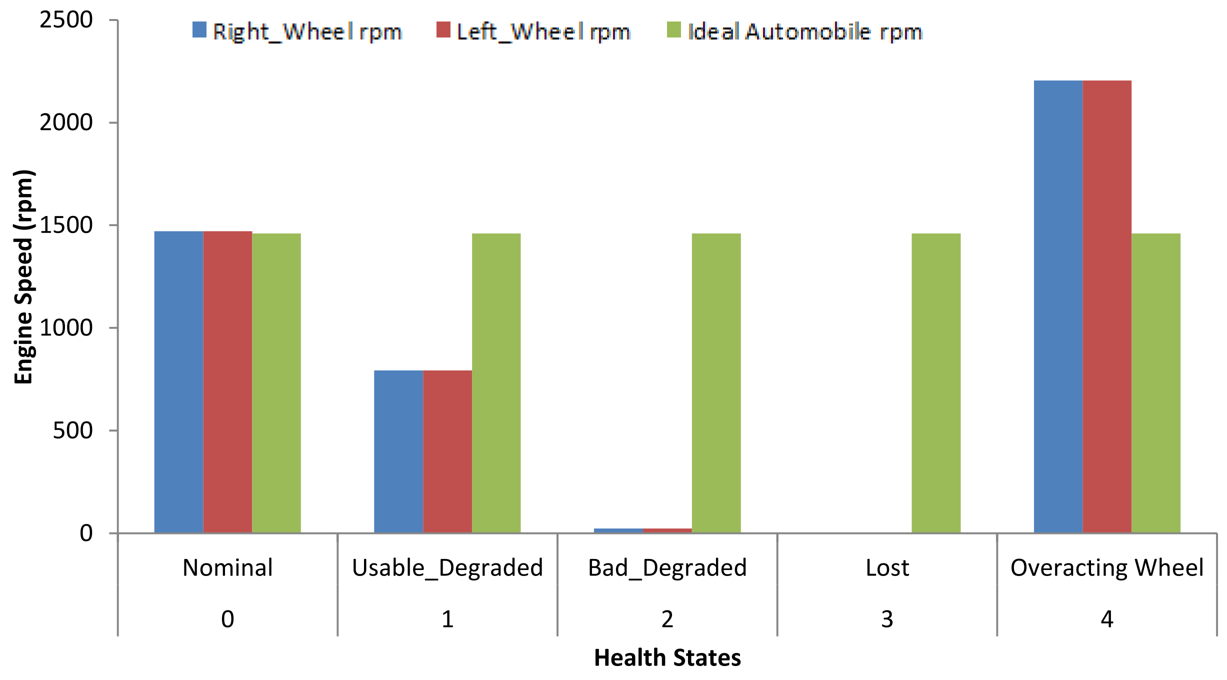


FIGURE 4.9: DOUBLE FAILURE IN BATTERY AND INVERTER

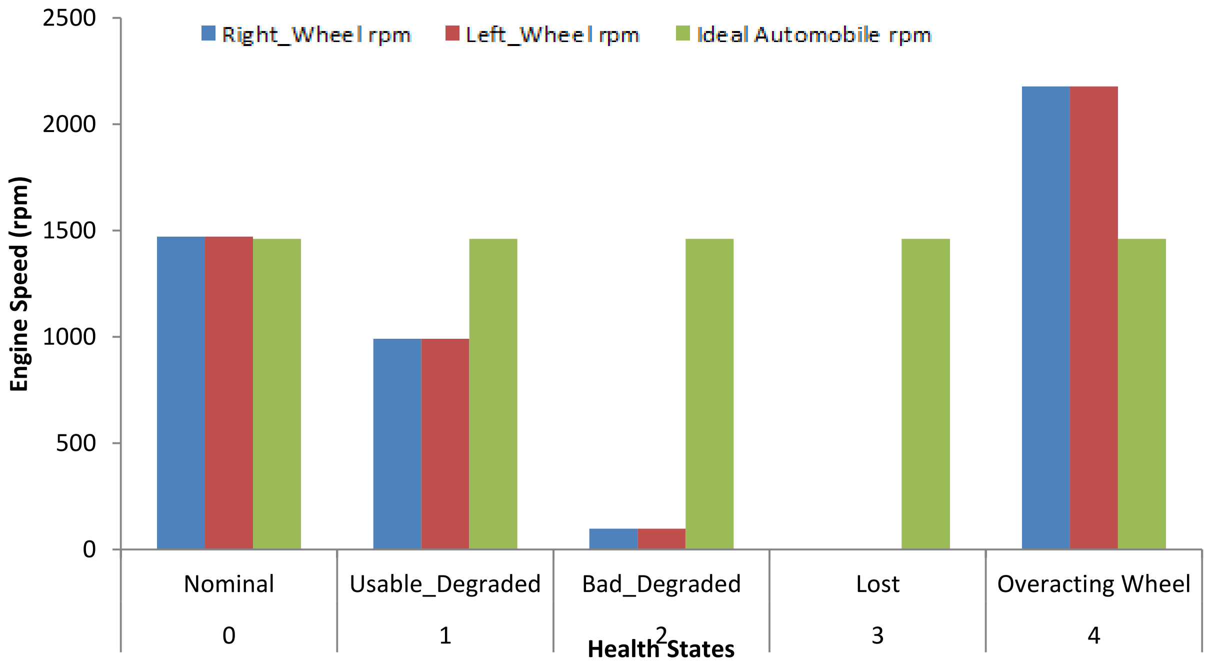


FIGURE 4.10: DOUBLE FAILURE IN THE INVERTER AND ELECTRIC MOTOR

From the output results of the Double Fault Scenarios, it can be observed that there is a higher impact caused by a combination faulty Battery and Inverter than there is with a combination of the Inverter and Electric motor when the latter are at faulty states.

4.9.4 Triple Fault Scenarios:

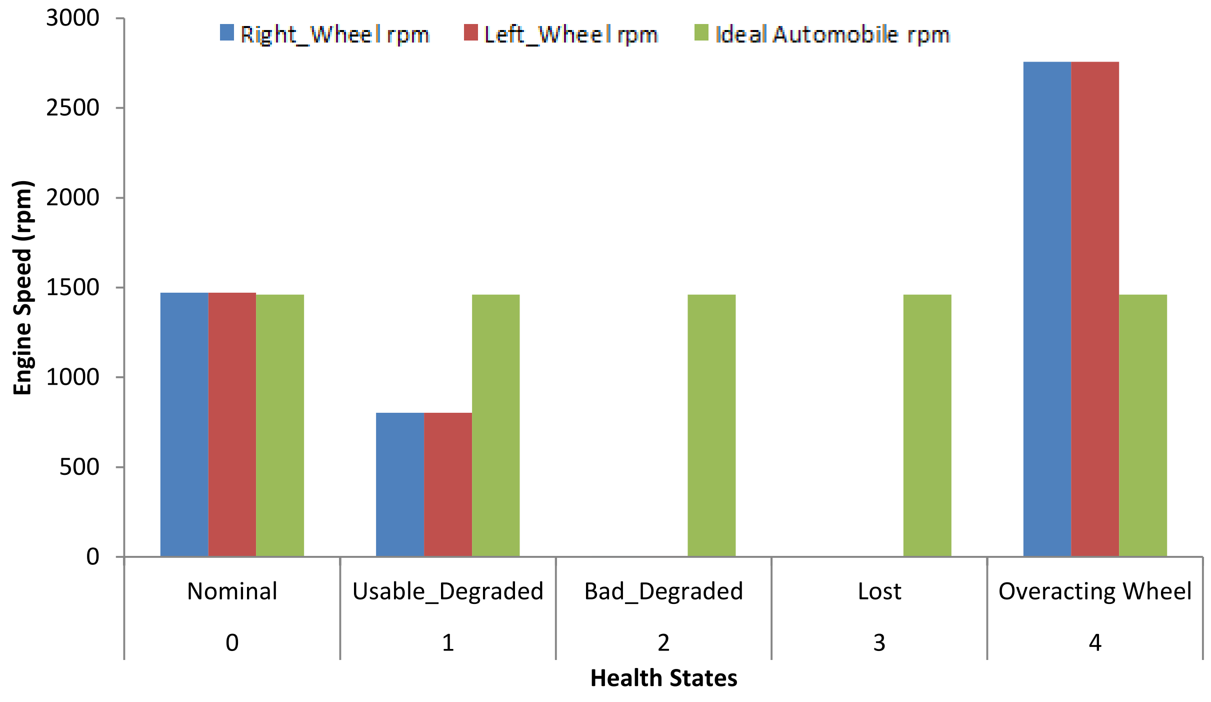


FIGURE 4.11: TRIPLE FAILURE IN THE BATTERY, INVERTER AND ELECTRIC MOTOR

From the output results of the Triple Fault Scenarios, it can be observed that there are huge drops in the energy supply to the wheels than there are with Single and Double Faults. The degraded values with triple faults will render the electric vehicle in-operable at this state.

4.10 Significance of Design Details at the Conceptual Stage

At the conceptual stage in the design of the electric vehicle, minimal information on the intended components in the make-up of the vehicle system is known. However, failure modes of the electric vehicle can be identified from the functional models of the system. In order to mitigate failures early at this stage, a detailed model of the system at various abstraction levels will help in adequately predicting such faults as presented above.

The usefulness of the approach presented in this work goes beyond evaluating system performance in the presence of faults within the inherent components. Using the functional and behavioral models of the electric vehicles, parametric variations of component data can also be carried out during design selections. This allows the system designer to examine possible trade-offs needed during component selection to optimize system performance in the electric vehicle.

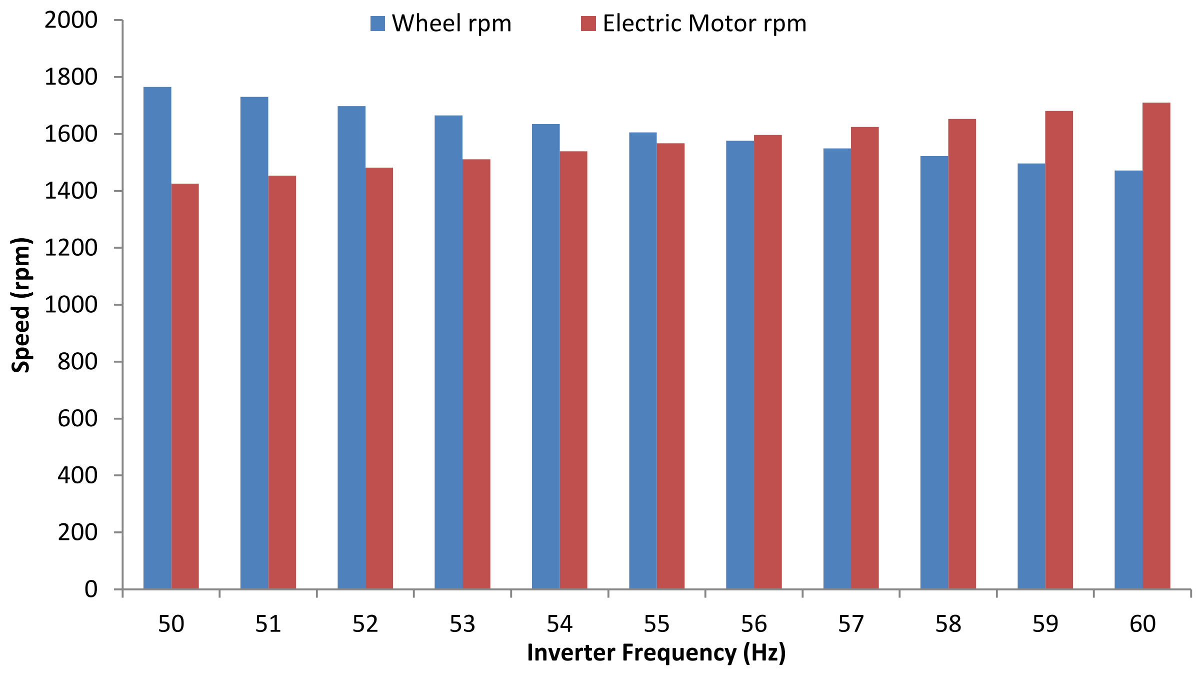


FIGURE 4.12: VARIATION IN SPEED AT DIFFERENT INVERTER FREQUENCY

Figure 4.12 shows the possible rpm speed values that can be obtained from the wheel and the electric motors from 50 Hz – 60 Hz of inverter readings in the electric vehicle. By observing the values above, equilibrium in the wheel speed and motor speed can be obtained using a 55Hz inverter. However, a particular frequency may not independently be isolated as optimal due to the requirements from other parts within the system. The model will serve as an alternative guide to using generic tables in selecting components for the chosen system.

Other trade-off analyses were carried out on the Electric vehicle's component parameters. The effect of varying the gear ratio is shown below.

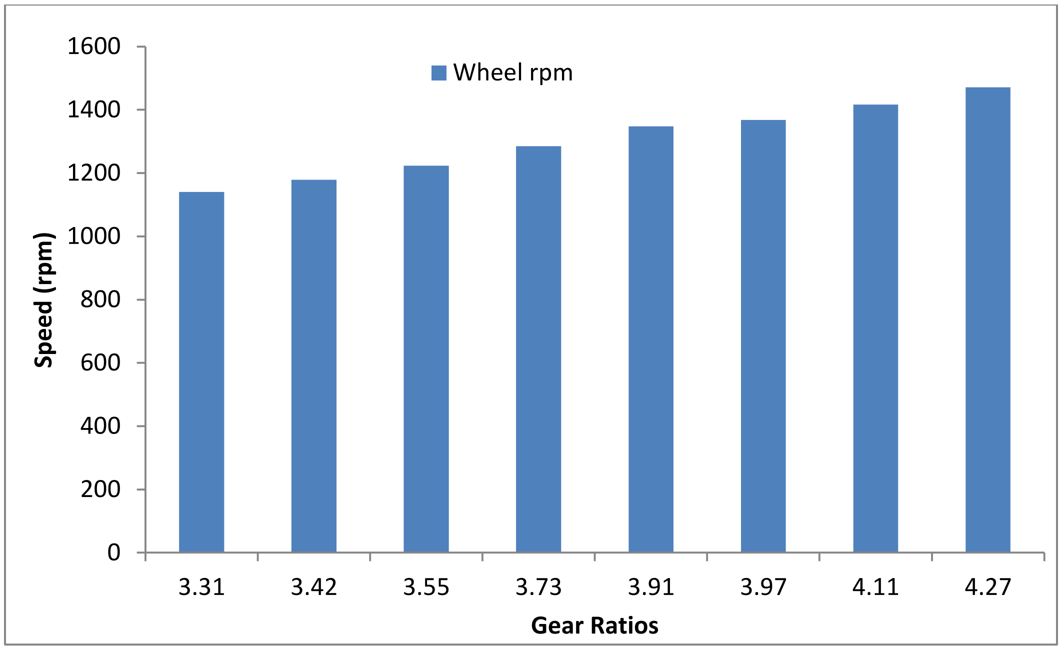


FIGURE 4.13: EFFECTS OF VARYING GEAR RATIO

Figure 4.13 shows that the speed obtained from the electric vehicle is increased with a corresponding increase in the gear ratio selected.

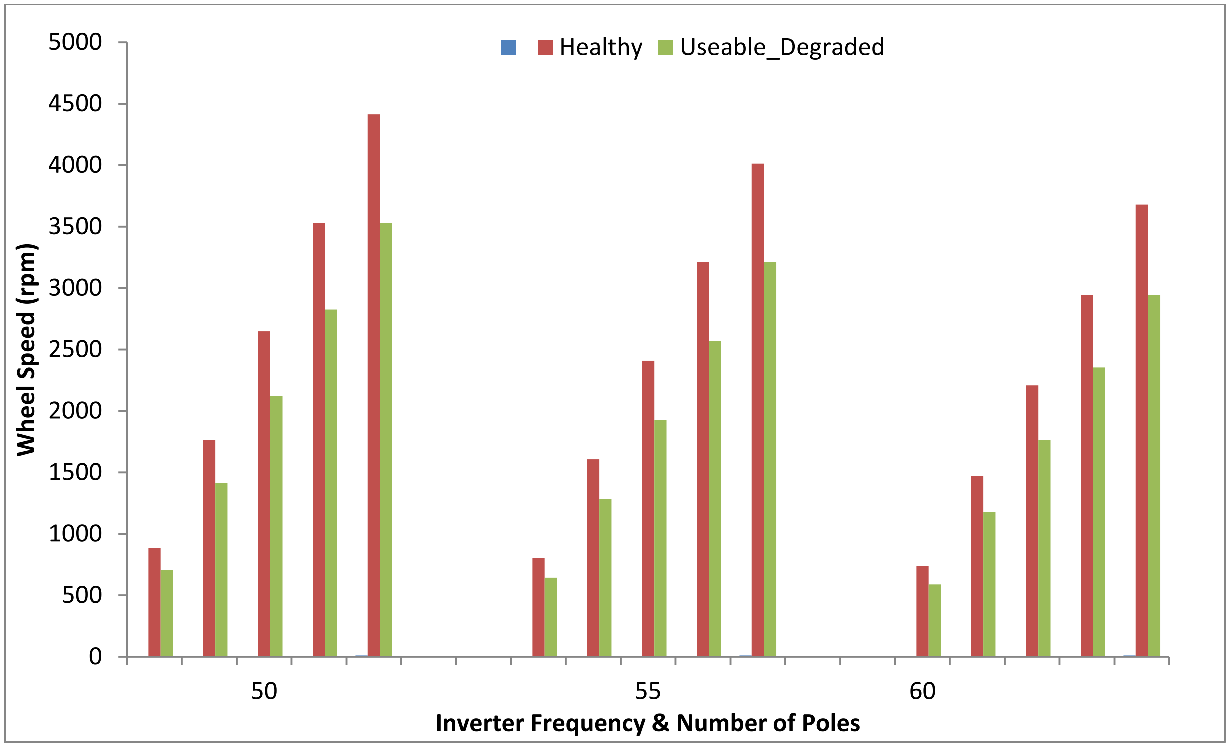


FIGURE 4.14: EFFECTS OF MULTIPLE PARAMETERS ON SYSTEM PERFORMANCE

A larger scenario of parameter variation within the electric vehicle is shown in Figure 4.14. The figure shows the results of the wheels speeds obtained using a gear ratio of 4.27 in the electric vehicle while varying the health state, the inverter frequency and the number of poles (2, 4, 6, 8, and 10) of the inverter. From the figure 4.14, it is observed that there is an approximately 4% difference between the “Usable_degraded” health state of the electric vehicle when operating at 50 Hz and the “Healthy” health state of the electric vehicle when operating at 60 Hz.

The details highlighted in the examples above make the adoption of the approach presented in this work suitable for use at the conceptual stage.

4.11 Impacts of Flow Fidelity on Behavior Models

Early in design stages, most components are often chosen without a prior knowledge of their performance before being integrated into the intended system. The level of detail to which designers may model these components for simulation can always vary depending on previous experience with such components. Regardless of this knowledge, an appropriate level of detail showing the intended behavior of components and their corresponding failure behavior needs to be captured to sufficiently conclude on components' overall impact on the system in which they operate.

In this section, the electric motor from the electric vehicle model is used for illustration. The principal aim of an electric motor is to convert an electrical energy input to a mechanical energy output. However, the electric motor used in most electric vehicles often provides several other operational outputs needed by other parts of the vehicle. Different interactions would normally take place within a system's components for it to execute its overall functions. These interactions can be adequately captured using discrete and continuous modeling methodologies. Here, we explore different levels of discrete and continuous simulations in order to draw conclusions on what level of fidelity provides an adequate report on component performance. When creating models of a component such as the electric vehicle motor, the designer will need to expertly determine what level of information is needed to adequately quantify the behavior of such component.

Based on components physics and the engineering knowledge available, various functional flows can be chosen to represent detailed interactions that take place from the input end to the output end of the component. The level of fidelity chosen to model such a component will affect the degree of accuracy and information content obtained from behavioral analysis results. This method also helps prioritize needed design simulations by optimizing computational cost and time.

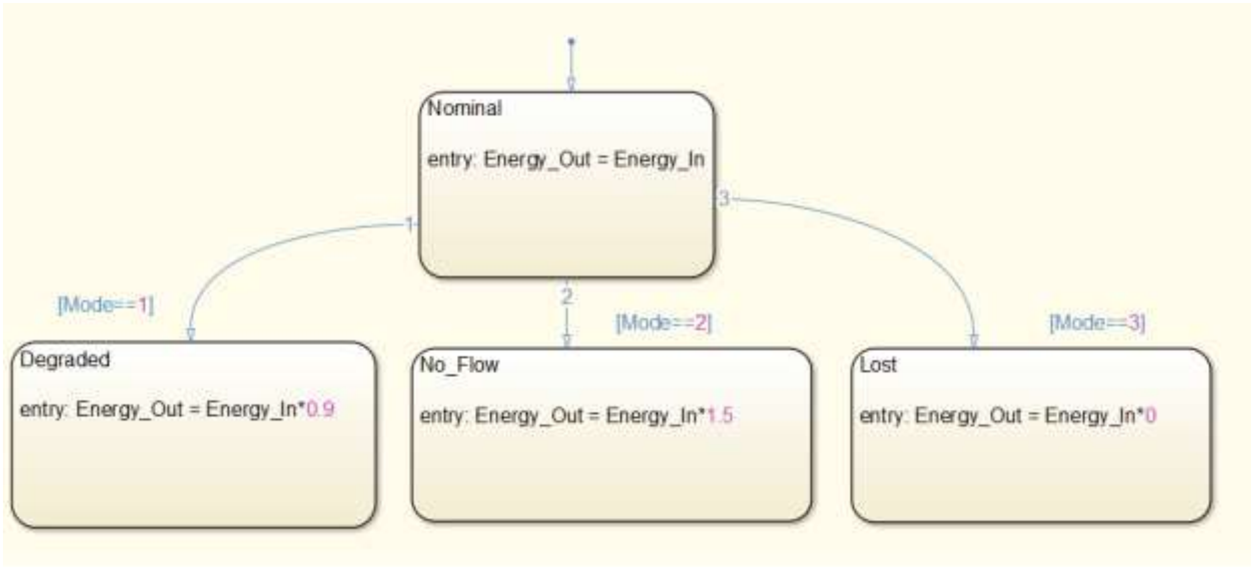

Nominal
entry: Energy_Out = Energy_In
1
2
3
[Mode==1]
[Mode==2]
[Mode==3]
Degraded
entry: Energy_Out = Energy_In*0.9
No_Flow
entry: Energy_Out = Energy_In*1.5
Lost
entry: Energy_Out = Energy_In*0

FIGURE 4.15: E-MOTOR BEHAVIOR MODEL AT FIDELITY LEVEL OF FUNCTION 1

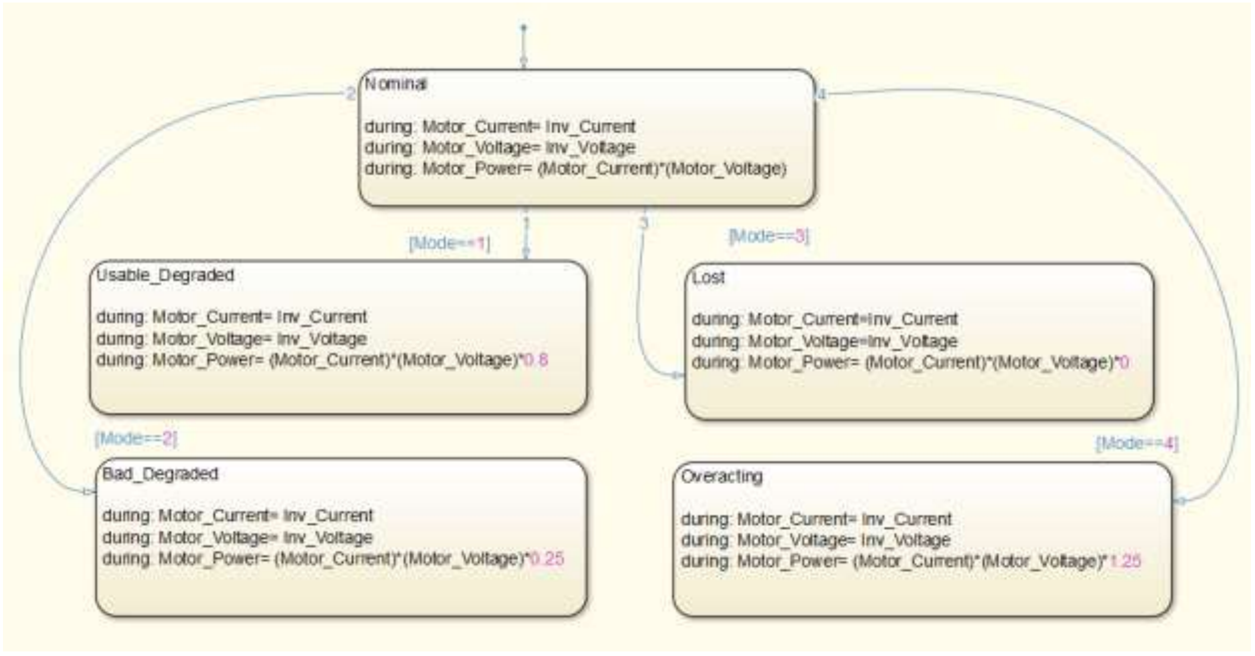

Nominal
during: Motor_Current= Inv_Current
during: Motor_Voltage= Inv_Voltage
during: Motor_Power= (Motor_Current)*(Motor_Voltage)
1
2
3
4
[Mode==1]
[Mode==2]
[Mode==3]
[Mode==4]
Usable_Degraded
during: Motor_Current= Inv_Current
during: Motor_Voltage= Inv_Voltage
during: Motor_Power= (Motor_Current)*(Motor_Voltage)*0.8
Lost
during: Motor_Current=Inv_Current
during: Motor_Voltage=Inv_Voltage
during: Motor_Power= (Motor_Current)*(Motor_Voltage)*0
Bad_Degraded
during: Motor_Current= Inv_Current
during: Motor_Voltage= Inv_Voltage
during: Motor_Power= (Motor_Current)*(Motor_Voltage)*0.25
Overacting
during: Motor_Current= Inv_Current
during: Motor_Voltage= Inv_Voltage
during: Motor_Power= (Motor_Current)*(Motor_Voltage)*1.25

FIGURE 4.16: E-MOTOR BEHAVIOR MODEL AT FIDELITY LEVEL OF FUNCTION 2

Different system information is obtained while using the component model representations utilized above. A tertiary function abstraction can effectively be utilized with the three levels of flow and behavior abstractions in creating a detailed component model. The tertiary levels for all modeling abstractions provide the design engineer with the most detailed information on system health states and overall performance. Time-based behavioral models using software tools such as ADAMS and Modelica make use of tertiary behavior abstractions. These models allow in-built physics that includes design criteria such as material types, properties, and operating cycles as inputs during simulation and analysis of the system. Other considerations such as the development of heuristics for all potential failure space in which the intended system will operate are also provided at this level.

This research explores how failure prediction in systems at the early design stages despite the presence of some design constraints and limited knowledge of the intended system. By using either the primary or secondary flow or behavior abstraction for modeling component behavior, Figure 4.15, the following information can be obtained:

- Component and system faults can be predicted from the output information on health states (such as "Healthy", "Degraded", etc.)
- There is limited information on how changing parameters will affect the system.

While the use of the tertiary level of flow abstractions together with primary or secondary behavior abstractions, Figure 4.6 and Figure 4.16, provide the following information:

- Component and system faults can be predicted from the output information on both the health states (such as "Healthy", "Degraded", etc.) and the actual reported values from the components and systems during operation

- Design selections and parameters can also be varied using this level of modeling
- Detailed information on how changing parameter variables affect the system performance. This information helps when creating design redundancies and alternative routing of power to supplement failed components

4.12 Dynamic System Model

Upon the completion of an abstract-based simulation of function and behavior of components using simulink, the results obtained are usually compared with actual physical prototype tests for verification. In some cases, the knowledge or expertise of the designer in quantifying the needed parameters necessary to provide details on the system's space (both nominal and failure) may be limited. There are usually certain levels of oversights during the early design stage. A dynamic simulation tool such as the Wolfram/SystemModeler software which uses a Modelica library of components with in-built physics equations to effectively capture the general physics associated with most commonly used engineering components and systems. This tool helps the designer to create actual prototypes of the desired system in a simulation environment while having exactly the same working principles as physical prototypes.

This work effectively assembles the relevant components needed to build an electric vehicles powertrain using SystemModeler based on the general requirements set for the design that were stated above. Most of the built-in components have their physics set exactly similar to real life components. Some other needed components can also be built as an assembly of parts that make up the component while taking into account their constituting physics. SystemModeler also has the added advantage of having built-in reliability modules (for example Weibull, Exponential and ChiSquare distributions), suitable for different components simulation.

The failure modes of the components were injected to the system model by creating a set of tables with actual values representing nominal, degraded, Lost and overacting. These values can be preset and changed by the designer based on engineering judgement. The actual values needed for the vehicle to function normally are set as the nominal values while the deviations are percentage difference from this value. The purpose of this type of simulation is to get very precise response of the system to the set variables of the component before an actual physical prototype is built and the results can be compared.

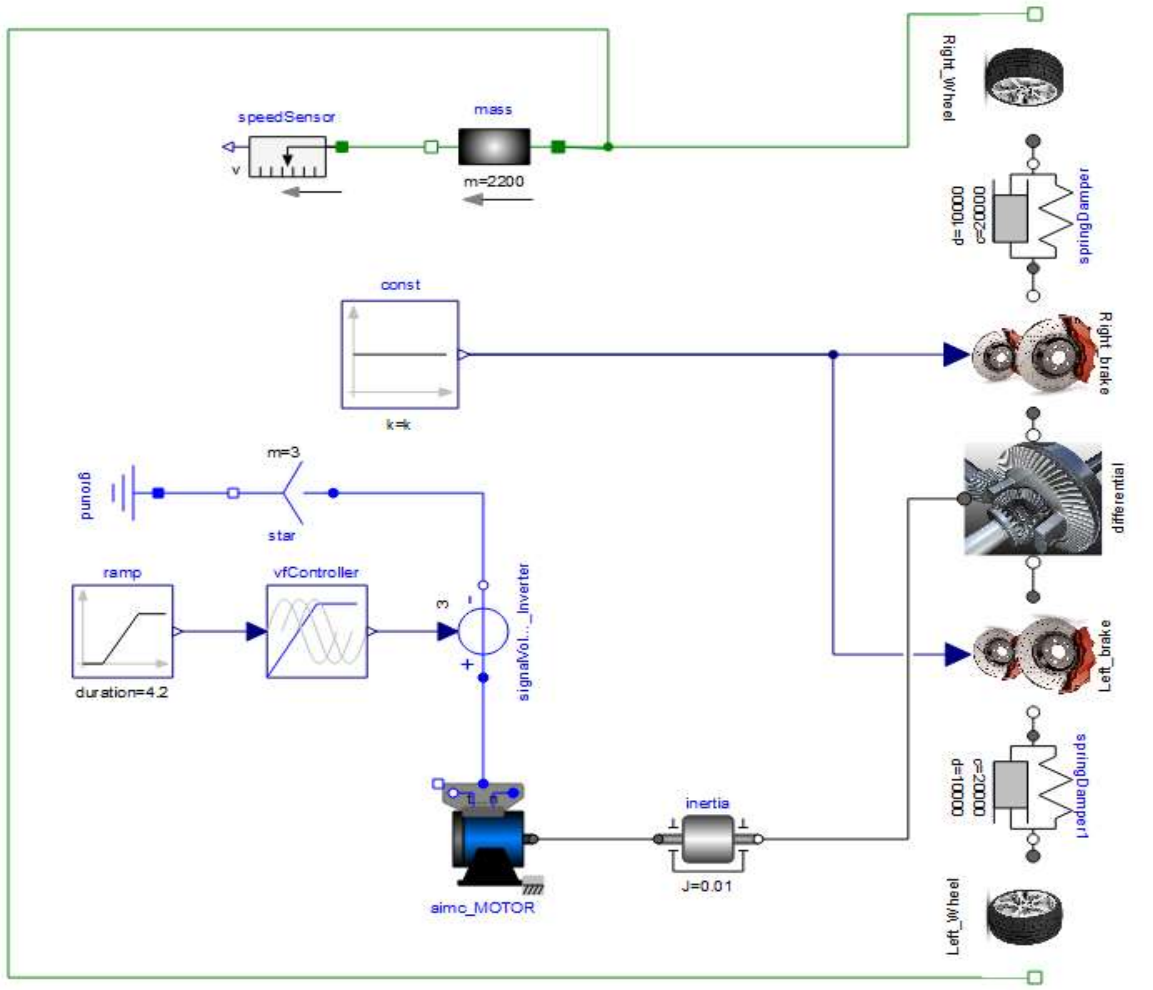


FIGURE 4.17: DYNAMIC SIMULATION MODEL OF THE ELECTRIC VEHICLE USING SYSTEMMODELER

4.12.1 Dynamic System Model Simulation and Results

The output voltage from the battery was initially set at the nominal state using actual values from the specification above. However, various degraded failure conditions were set by percentage deviations from the otherwise healthy condition of the component. The results of the dynamic simulations on the entire system using different battery voltage are shown in Figure 4.18

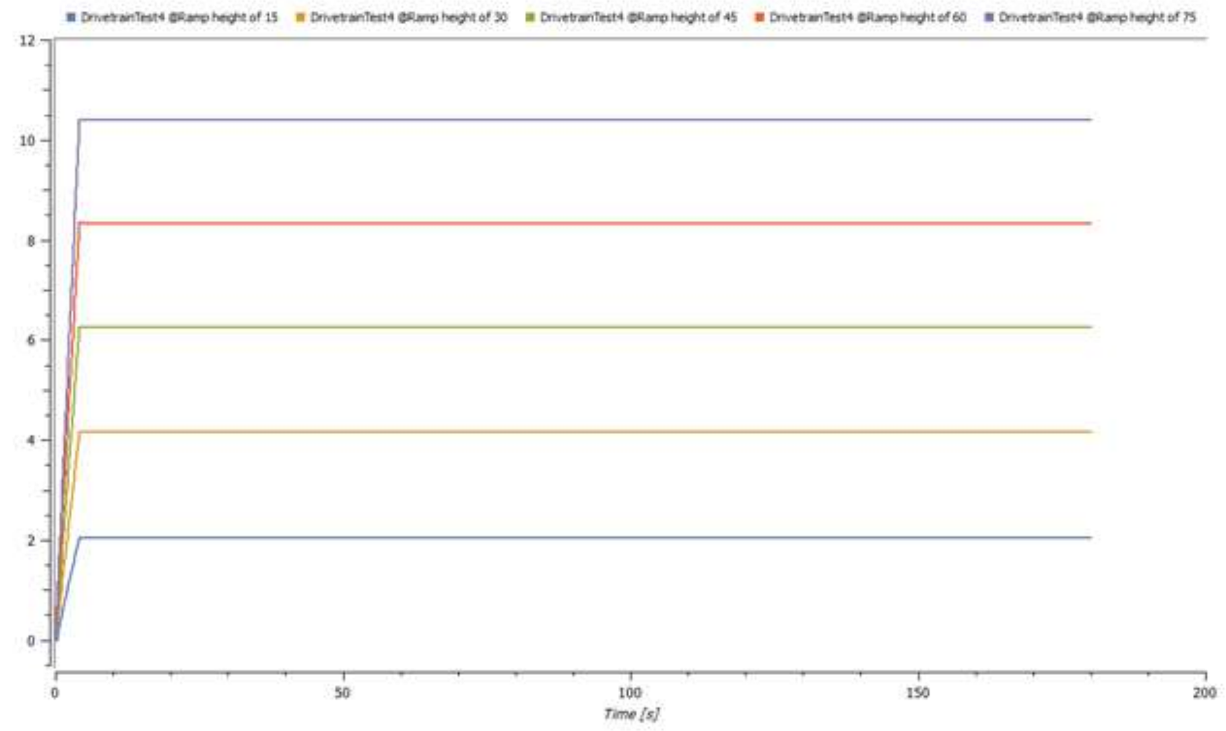


FIGURE 4.18: RESULTS OF SYSTEMMODELER SIMULATION

The dynamic simulation provides a time-based impact of degradation from the healthy state for all system components and the system itself. The plots of the results in Figure 4.18 show the time taken for the electric vehicle to accelerate to reach top speed within a certain time. Reduced voltage supply from the battery limits the ability of the vehicle to accelerate as other components draws from the same low output. Other than being able to identify the existence of faults in the battery, these types of results have the potential of being useful to designers or drivers when planning missions such as driving uphill or downhill based on battery readings. Similar tests for failure in other components are also explored to study their impact on the system performance.

4.13 Conclusion

This work uses the FFIP simulation to predict the performance of a system through the combination of traditional failure analysis tools and system simulation tools. Different levels of abstraction and model details were explored in order to determine the potential health status of a system based on the information available. From the results of the research, the primary level of abstraction details for function, flow and behavior, can provide information on possible health states of the system. However, using the secondary and tertiary behavior abstraction model detail will provide predictions on the health states and actual performance values. Information on impacts of varying component parameters in the intended system is also provided using the higher abstraction levels. The methodology provided is mostly conditioned and suitable for failure analysis at the conceptual stage.

The FFIP simulation fails to capture the time-sensitive degradation introduced by the cascading failure effect while the Dynamic simulation used in this work effectively does. For example, the FFIP simulation successfully observes the degraded health state of a particular function but does not have a means to determine when a degraded state or lost state starts. The ability of the behavioral model simulation to mimic the severity or degree to which a functional impact is affected is determined by the amount of detail present in the behavioral descriptions.

<u>References</u>

[1] Buede, D.M., 2009, "The engineering design of systems: Models and methods," John Wiley \& Sons, New York.

[2] Hunter, S. C., Jensen, D. C., Tumer, I. Y., 2016, "The Impact of Abstraction and Fidelity Levels on the Usefulness of Early System Functional Models," ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. V01BT02A018-V01BT02A018.

[3] Roza, Z.C., 2005, "Simulation fidelity theory and practice," TU Delft, Delft University of Technology.

[4] Kurtoglu, T., Jensen, D. C., and Tumer, I. Y., 2010, "A Functional Failure Reasoning Methodology for Evaluation of Conceptual System Architectures," 21.

[5] Kurtoglu, T., Johnson, S., Barszcz, E., 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, PHM'08,

[6] Kurtoglu, T., and Tumer, I. Y., 2008, "A Graph-Based Fault Identification and Propagation Framework for Functional Design of Complex Systems," 130(5) .

[7] Tumer, I. Y., and Smidts, C. S., 2010, "Integrated Design and Analysis of Software-Driven Hardware Systems," 60pp. 1072-1084.

[8] Jensen, D. C., Bello, O., Hoyle, C., 2014, "Reasoning about System-Level Failure Behavior from Large Sets of Function-Based Simulations," Artificial Intelligence for Engineering Design, 28(04) pp. 385-398.

[9] O'Halloran, B. M., Haley, B., Jensen, D. C., 2014, "The Early Implementation of Failure Modes into Existing Component Model Libraries," 25(3) pp. 203-221.

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

[11] Hirtz, J., Stone, R., McAdams, D., 2002, "A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts," 13pp. 65-82.

[12] Umeda, Y., Ishii, M., Yoshioka, M., 1996, "Supporting Conceptual Design Based on the Function-Behavior-State Modeler," Artificial Intelligence for Engineering Design, 10(4) pp. 275-288.

[13] Stone, R. B., and Wood, K. L., 2000, "Development of a Functional Basis for Design," 122(4) pp. 359-370.

[14] Hutcheson, R., McAdams, D. A., Stone, R. B., 2007, "Function-based behavioral modeling," Proceedings of the ASME Design Engineering Technical Conferences; International Design Theory and Methodology Conference},

[15] Joshi, A., and Heimdahl, M. P. E., 2007, "Behavioral Fault Modeling for Model-based Safety Analysis," In HASE 07, IEEE Computer Society,   IEEE, pp. 199-208.

[16] Balci, O., 2003, "Verification, validation, and certification of modeling and simulation applications: verification, validation, and certification of modeling and simulation applications,"

Proceedings of the 35th conference on Winter simulation: driving innovation, Winter Simulation Conference, pp. 150-158.

[17] Burnett, E. L., 2008, "A Proposed Model Fidelity Scale," Proceedings of AIAA Modeling and Simulation Technologies Conference and Exhibit, Lockheed Martin Aeronautics Company.

[18] Hancock, P.A., Vincenzi, D.A., Wise, J.A., 2008, "Human factors in simulation and training," CRC Press.

[19] Hunter, S. C., Jensen, D. C., Tumer, I. Y., 2016, "The Impact of Abstraction and Fidelity Levels on the Usefulness of Early System Functional Models," ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. V01BT02A018-V01BT02A018.

[20] Larminie, J., and Lowry, J., 2004, "Electric vehicle technology explained," John Wiley & Sons,

CHAPTER FIVE

REASONING ABOUT SYSTEM-LEVEL FAILURE BEHAVIOR FROM LARGE SETS OF FUNCTION-BASED SIMULATIONS [1]

This chapter presents modified excerpts of a published co-authored work on the analysis and reasoning about FFIP simulation results. The electric vehicle health states clustering analysis from the research in Chapter 4 are included in this chapter.

A version of this chapter has been published in the Artificial Intelligence for Engineering Design Journal.

[1] Jensen, D., Bello, O., Hoyle, C., & Tumer, I. (2014). Reasoning about system-level failure behavior from large sets of function-based simulations. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 28(4), 385-398. doi:10.1017/S0890060414000547

5.1 Introduction

The primary objective of this work is to develop a design-stage simulation and analysis tool set that uses simulation data to reason about the functional robustness of systems to potential component faults and fault propagation. This type of approach is intended to enable designers to compare potential system architectures, identify component and subsystem behaviors that lead to undesired system states, and assess the impact of complex fault scenarios. In order to achieve this high-level objective there are three specific objectives that this presented method addresses. These are:

1. Characterize the impacts of a large space of the potential complex failure scenarios. (In what types of ways does the system fail?)

2. Identify the system-level importance of the sets of potential system failures. (What does each type of failure mean in terms of system functionality?)
3. Determine how this analysis can be used to make system design decisions. (Can we use this data for a systems view of functional robustness?)

By addressing the first objective, this method moves beyond single scenarios analysis and begins to develop a system-level characterization based on simulation of component behavior. The result of completing the first objective is distinct types of system failure analogous to failure modes for the system. However, since these are identified through simulation and data analysis, the types of system failure must be related to the system-level functionality. In this way, objective two enables this method to link top-down and bottom-up analysis methods. Finally, the third objective begins to address how this approach can fit within the overall systems design processes.

5.2 Background

This section discusses the three technical areas used in this paper and presents some detail of the example system.

The FFIP section of this article which is the source of the data on which the analysis and clustering methods are applied has been extensively discussed in the previous chapters of this work. A brief background on the method of clustering data using a k-means algorithm is provided. Finally, a categorical data clustering approach for identifying an underlying probabilistic model for the structure of the data, namely, Latent Class Analysis is presented.

5.2.1 Data Clustering

Separating data into clusters or partitions has been a useful activity in the data mining community to elicit meaning from large data sets [2]. Starting with the classification of human traits and personality in the 1930-40s, clustering analysis continues to be an important tool to enable machine learning. Multiple methods and algorithms have been developed based on different perspectives on the meaning of a cluster [3]. There are three main approaches to clustering with multiple methods and algorithms supporting them.

Hierarchical clustering assumes that some category or classification captures all the data and that data points can further be sub-classified into more specific groups in a tree structure. In biology, the Linnaeus taxonomy of living things is an example of hierarchical clustering. Hierarchical methods often relate one or more data points by their similarity.

In contrast to hierarchical methods, partitioning methods separate the data space into different clusters without implying a higher level relationship between those clusters. Data points are related based on a measure of the distance between values. Algorithms that implement partitioning identify centroids of the clusters and then group all data points into a predetermined number of clusters based on their distance from that centroid. K-means clustering is one method of data partitioning that evaluates the Euclidean distance between data points [4].

Two significant issues of k-means clustering are that the number of clusters must be selected first and that data points may only have membership in one cluster. To address the first issue, heuristic rules such as choosing k based on the square root of half the data set size can provide an initial assessment [5]. Evaluation of the correctness of the value of k can be done through heuristic metrics as well. Variations of k-means known as soft or fuzzy clustering methods use a similar approach but instead provide membership percentages.

The third category of data clustering methods is model-based. These methods assume some structure to the data and try to find the correct statistical model to match that structure. Methods in this category use different means of estimating and finding the maximum likelihood of the data fitting the parameters of a statistical model [6, 7].These methods assume that the reason some data points are related to other data points is due to some unobserved (or latent) variable. Unlike k-means, data points have a probability of being within a particular cluster based on their dependence to that unobserved variable. There are many variations of model-based clustering depending on the form of the data and the likely form of the clusters. For the analysis of function-based failure simulation data, the most appropriate model-based method is Latent Class Analysis. The details of this analysis and the justification for its use in this work are presented next.

5.2.2 Latent Class Analysis

Social scientists have used the concept of latent classes since the 1950s [8]. 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. 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 [9].

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 this 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 (**Y**) of manifest variable states y, denoted P(**Y** = y), is a weighted average of the C class-specific probabilities P(**Y** = 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 1.

$$P(\mathbf{Y} = y) = \sum_{X=1}^{C} P(X = x)\, P(\mathbf{Y} = y / X = x) \qquad (1)$$

Further, the manifest variables within a class, Y_l are assumed to be locally independent. Therefore, Equation 2 defines the probability of observing a pattern in the L manifest variables within a class.

$$P(\mathbf{Y} = y / X = x) = \prod_{l=1}^{L} P(Y_l = y_l / X = x) \qquad (2)$$

Using the political example above, (**Y**) is the pattern of answers associated with a political group answering the specific questions y. This pattern is independent within each of the discrete political groups in X.

As with k-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 either the Akaike Information Criterion (AIC) or the Bayesian Information Criterion (BIC). These are metrics to estimate 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 3, where **K** is the number of parameters and **L** the likelihood function for the statistical model. The BIC formulation is similar but accounts for the sample data size.

$$\text{AIC} = 2\mathbf{K} - \mathit{ln}\,(\mathbf{L}) \tag{3}$$

LCA was chosen as a clustering method over other clustering methods because the manifest variables are the discrete health 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.

5.2.3 Example System Case Study

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 example system will be used to simulate numerous fault scenarios, identify the set of functional impacts for each scenario and apply the clustering algorithms to find patterns of system failure behavior.

This EPS example is an early design-stage model that uses batteries 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 [10]. In previous work, various potential design architectures were compared using a quantified interpretation of the FFIP results [11]. The example used in this work expands upon a similar but less complex example [12].

As seen in Figure 5.1, the concept for the EPS is a fault tolerant software controlled hardware system. At the system level, three operational states are recognized. Specifically, “Nominal”, when both load banks of AC and DC loads are operational; “Degraded”, when only one of the load banks is operational; and “Lost”, when neither load bank is operational. The purpose of the software control is to automatically maintain operation at a nominal state if possible and a degraded state otherwise. By evaluating the voltage levels in both the load banks and both battery banks the controller decides to open or close Relays 1 through 4. The first rule implemented in the software control is that no two batteries can be connected together. For example, Relays 1 and 4 cannot both be closed while there is power available from both batteries or an electrical over current will occur. After this rule, the controller observes the voltage and relay position sensor values to determine which relays to open or close to ensure continued operation. In a fault scenario, the controller can decide to swap power so that the first battery powers the second load and vice versa or simply to shut down one line and run at a degraded state. The control logic is implemented with a truth table where values of sensors correspond to specific relay positions. The control attempts to keep the system in the best operating state as described in Table 5.1. In this table the term “Batt1→ Load1" indicates that Battery Bank 1 is powering Load Bank 1.

TABLE 5.1: OPERATIONAL STATES THE SOFTWARE CONTROL ATTEMPTS TO MAINTAIN

Nominal		**Degraded**		**Lost**
State 1	State 2	State 3	State 4	State 7
Batt1 → Load1 Batt2 → Load2	Batt1→ Load2 Batt2→ Load1	Batt1 → Load1	Batt2 → Load2	No Action
		State 5	State 6	
		Batt1 → Load2	Batt2 → Load1	

This fault tolerant example system enables the identification of high-level system goals such as maintain load operation and illustrates fault propagation over both software and hardware components. This example system is complicated enough to demonstrate the clustering methods yet still provides clarity in the impact of complex faults. The FFIP analysis has also been demonstrated on a more complicated system (nuclear power generation [13, 14]).

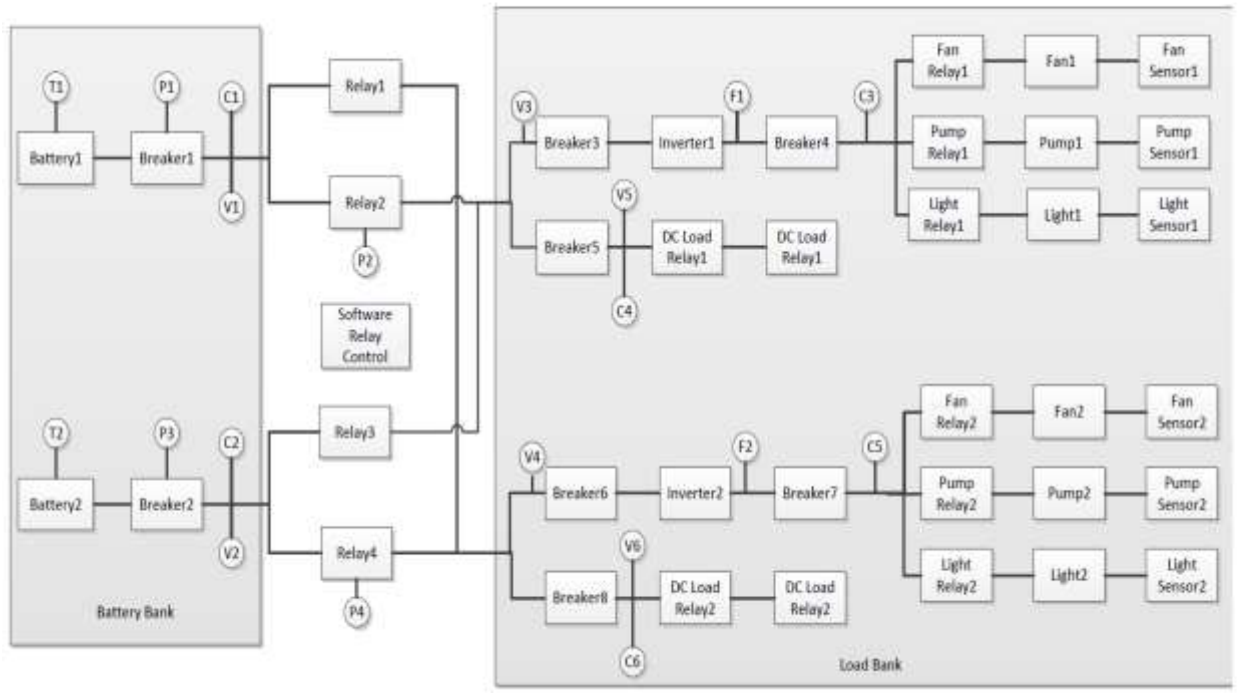


FIGURE 5.1: ARCHITECTURE OF THE ELECTRICAL POWER SYSTEM (EPS) USED FOR FUNCTION-BASED FAILURE ANALYSIS AND RESULTS CLUSTERING

5.3 Methods

The development and justification of the functional effect analysis using the FFIP methodology is documented in previous work [11, 14-16] and will not be repeated here. Because the motivation of this work is to use data analysis techniques to identify underlying system behavior, we begin with collecting the analysis results from the FFIP-based simulation. Other methods of design analysis and simulation could be used instead. The two things that are needed to apply these techniques is a large number of behaviors to simulate (many scenarios) and multiple data points to describe each scenario.

FFIP provides this by the ability to simulate single and multiple fault scenarios as well as variations in ow parameters. Further, for each scenario simulated, the result is the health state of each component-level function in the system. These function health states are the variables that describe the system state in response to the simulated scenario. In the following sections we discuss the simulation and collection of functional effect failure data and the application of the similarity clustering and probabilistic latent class analysis.

5.3.1 Identifying the Functional Impact of Component Faults and Interactions

The impact 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 are triggered and the resulting changes in system dynamics are allowed to propagate. The output of each simulation is the function health state of each component-level function.

For example, one scenario includes triggering the failure behavior for both batteries. To simulate this scenario, the system simulation begins with all components operating nominally. Then after 25 time steps the first battery's operating mode is changed to “Failed-Disconnected." The effect of this change is the loss of current and voltage from that component. After 50 time steps the second battery's operating state is changed in the same way. The effect of these changes is allowed to propagate through the system. In this example, the software controller attempts to switch between sources by changing which relays are closed. Finding no solution that provided power to the loads, the software controller by default opens all relays as a failure safety measure.

After 100 time steps, the simulation is ended and the final function health state for each component-level function is recorded as the result for that scenario. The injection of failures at 25 and 50 time steps is arbitrary. Through analysis of numerous simulations it was found that the state machines used need four to eight time steps to reach a steady state. Further, reducing the time between failure mode insertions resulted in no change to the final system state. However, the order of the fault mode changes did affect the final system state results for many scenarios (excluding the one above). Therefore, every order of faults is also simulated. Because this system has 58 component-level functions, the result of simulating a scenario is a vector where each element corresponds to the health state of each of the 58 functions. These function health states are recorded as integers from 1 to 4 to ease data handling.

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. While simulating three or more scenarios is possible, for this example system the limited number of components resulted in few unique system states for more than two failure scenarios. 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 or simply using proximity. An alternative to this approach is simulating fault modes based on the relationship between causes and symptoms of faults [17]. This latter approach is based on triggering failure modes in components with fault symptoms (e.g. leaking) which are of the same type as fault causes (e.g. exposure to liquid). Finally, the clusters generated using the approach may provide guidance in identifying fault modes that should be simulated together in an iterative approach.

Function failure analysis results are collected from each scenario in a matrix where each row is a separate scenario and the columns correspond to the resulting identified health state of the component 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 193 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 37,299 fault simulation records. Both fault orderings were included because it is possible that the order of faults may change the system level effects.

5.3.2 Pre-processing to Enhance Clustering Effectiveness

The clustering methods demonstrated in this work are applied to find similarities and structure between different fault scenarios. However, the first level of grouping is to identify which fault scenarios resulted in identical functional results. These represent scenarios that cannot be functionally distinguished from each other. For example, faults in two loads that both cause high current draw can trip a breaker. The large number of combination of two load faults results in a large set of identical faults, that is, they all result in the same tripped breaker and subsequent loss of power. This grouping is accomplished through a simple sorting algorithm which groups identical scenario results into bins. Selecting one scenario result from each bin represents the set of unique system states.

When applied to the EPS example system the 37,299 total scenarios were sorted and 3,509 unique system states were identified. The significant reduction reflects a large number of identical functional impacts. Many of these identical impacts are related to faults in the sensors which all had five failure modes but resulted in little effect to the system because the controllers that use those sensors were not simulated. The exception to this was failures in the sensors used by the controller, where faults did result in a change in the behavior of the system. The unique system states represent one or more failure scenario results and are the data provided to the clustering methods.

5.3.3 Clustering of Results Based on Functional Similarity

The motivation for implementing similarity clustering is to identify groupings of failure scenarios and aid designers in creating robust mitigation methods. For example, if a system designer knows of a particular undesirable system state, then finding all scenarios that lead to a similar functional state can identify if adequate control methods have been implemented. In order to identify the relationship between two system states we must develop a metric of distance between function health states. In data clustering methods, the distance between variables can be determined based on the Euclidean distance between the variable values (Distance = $\sqrt{(a^2 + b^2)}$).

However, the values chosen to represent health states are categorical numbers not nominal numbers, which violates an underlying assumption in the Euclidean formulation. Therefore, we introduce a functional distance metric based on functional impact. A relational table (Table 5.2) is generated to define the similarity between function health states. For this analysis, we identify “Lost" and “No Flow" as having no significant functional difference to the system. Here, designers could choose to increase the distance of off-nominal states to effectively punish and group those scenarios as being worse. Since a low system-knowledge approach is being used for this example, all states have a single unit of difference. For example we can consider a system with two functions and compare the similarity of two fault scenarios. If the resulting system state from scenario 1 is {Healthy; Lost} and the system state from scenario 2 is {Degraded; NoFlow}, then the Euclidean distance between these two using the relation matrix in Table 5.2 is $\sqrt{(1^2 + 0^2)}$ or 1.

TABLE 5.2: RELATIONAL MATRIX FOR IDENTIFYING THE DISTANCE BETWEEN FUNCTION HEALTH 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

Table 5.2 is one way to quantify the qualitative distance between functional health states. The k-means clustering algorithm was also applied to the same simulation results using different distance values and where “No Flow" and “Lost" were not equivalent. The cluster centroid and distances between centroids changes when this scale is changed. However, when comparing the population of scenarios between clusters using different distance matrices, the average error is about 0.5%. This is within the normal variation of the algorithm when repeated with the same relational matrix. As a result of this finding, it is clear that the concept of functional similarity is strongly dependent on the scale used in this relational matrix. However, population of the clusters and the resulting meaning of those clusters are consistent across scales.

5.3.3.1 Results of Similarity Clustering

The total distance is calculated by summing over the distance for each function health state. A weighting for functional importance could be incorporated into this step. However, for this analysis each function is given equal importance. This algorithm identifies the functional similarity using Table 5.2 for each low-level function. Since there is no way to know a priori how many clusters to expect, we repeatedly call the k-means algorithm to cluster the data using

1-10 clusters. Additionally, the algorithm is replicated 100 times for each clustering to avoid local minimums.

There are several recognized methods of identifying the appropriate number of clusters. The first approach implemented is the “knee method" [7], where the within cluster sum of square (WCSS) distance to the cluster centroid is plotted. When additional clusters do not substantially change the WCSS there is no need to further cluster the data. Using the EPS example data, the inflection point appears between 5 and 7 clusters (see Figure 5.2a). This ambiguity results in the need for a second cluster validation method. By comparison of the dispersion of the scenario similarities within a cluster and the dispersion of the impacts of those scenarios it is possible to identify the appropriateness of the clustering groups.

For this work a plot is developed where cluster centroids are plotted against the sum of their function health states normalized by the total number of functions. That is, a vertical value of 1 indicates that all functions are at the healthy state (a nominal scenario). If all component functions in the system had failed in a scenario then the normalized impact would be 4. Vertical position gives an estimate of the scope of the system affected by the fault. Each scenario in a cluster is then plotted based on a horizontal position representing the distance of that scenario to the cluster centroid and a vertical position based on the normalized sum of function health states. Selecting to use five clusters for the k-means algorithm, the plot shown in Figure 5.2b illustrates the variance of the distances from the cluster centroid in the horizontal direction and the variance of the scenario impacts in the vertical direction.

TABLE 5.3: EVALUATING CLUSTER DISTANCE AND IMPACT MEAN AND COEFFICIENT OF VARIATION

Metric	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
Distance Centroid	Mean = 11.99 CV= 0.39	Mean = 7.54 CV= 0.3	Mean = 7.48 CV= 0.90	Mean = 8.03 CV= 0.26	Mean = 5.73 CV= 0.92
Normalized Scenario Impact	Mean = 1.44 CV= 0.10	Mean = 1.11 CV= 0.02	Mean = 1.28 CV= 0.18	Mean = 1.08 CV= 0.02	Mean = 1.31 CV= 0.04

For this example system, Table 5.3 records the mean and coefficient of variation for each cluster for the distance from the centroids and the normalized impact of the scenario. The coefficient of variation (CV) is the ratio of the standard deviation and the mean of a population where larger numbers indicate greater dispersion of the data. For the distance metric, the CV indicates how similar the scenarios in the cluster are to each other. For the impact metric, the CV shows the variation in the impact for scenarios in that cluster based on this data that the scenarios with the least similarity are in clusters 3 and 5. Similarly, the most diverse set of impacts is in found in clusters 1 and 3. Based on this analysis cluster 3 has the potential to have very dissimilar scenarios with somewhat significant differences in total functional impact. Since there was ambiguity in the correct number of clusters between 5 and 7 and the potential for cluster 3 to be subdivided, 6 clusters where selected for the analysis of scenario similarity.

5.3.4 The Latent Class Analysis Method

The second method of grouping the failure results is focused on identifying patterns of failure behavior. For this method a Latent Class Analysis (LCA) is performed on the 3,509 unique fault simulation results using the package poLCA [18, 19] for the statistical software tool R [20]. The poLCA package treats the manifest variables as categorical. The manifest variables in this analysis are the function health states and the latent variable describes the system failure behavior. Similar to the k-means clustering, the number of latent variable classes must be specified prior to the analysis. Therefore, an iterative approach is also taken to fit multiple latent class models with different numbers of classes. In order to avoid local maxima, the poLCA classification algorithm is executed 10 times for each specified number of classes. The correct number of classes is identified as the LC model with the lowest Akaike Information Criterion (AIC) and lowest Bayesian Information Criterion (BIC).

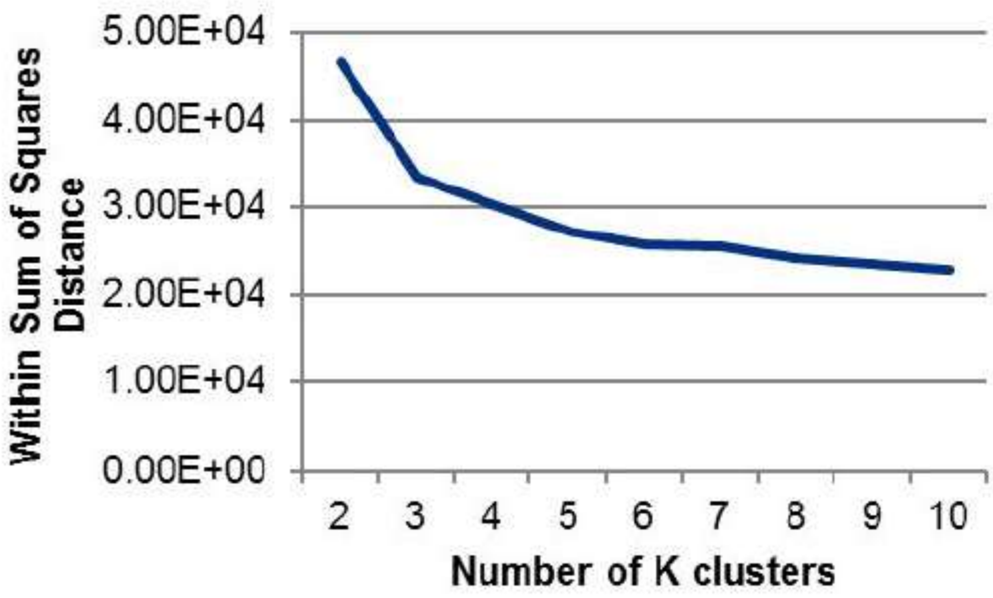


FIGURE 5.2(A): THE SUM OF THE WITHIN CLUSTER SQUARE DISTANCE OF SCENARIOS TO THE CENTROID OF THEIR RESPECTIVE CLUSTER.

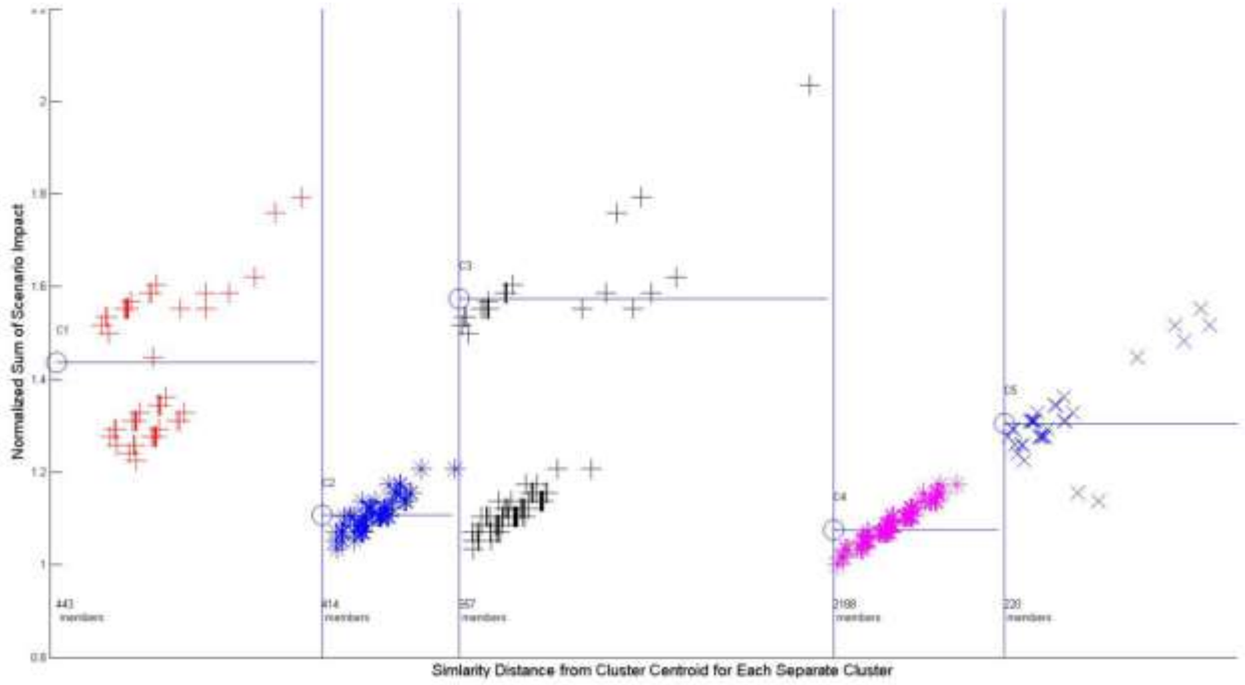


FIGURE 5.2(B): GRAPHING CLUSTERS BASED ON THEIR DISTANCE FROM CENTROID AND TOTAL SCENARIO IMPACT WITH 5 CLUSTERS. CLUSTER 3 HAS BOTH VERTICAL (IMPACT) VARIANCE AND HORIZONTAL (SIMILARITY) VARIANCE AND COULD BE SEPARATED INTO TWO CLUSTERS.

FIGURE 5.2: SUMMARY OF RESULTS FOR APPLYING A MODIFIED K-MEANS CLUSTERING TO THE UNIQUE SYSTEM FAILURE STATES.

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.

5.3.4.1 Results of Model-Based Clustering

The AIC and BIC tend to flatten when evaluating latent models with more classes. Implementing a LCA on the example system data set, minima of AIC and BIC can be seen at 5 classes and 8 classes. Unlike k-means clustering, LCA can identify probabilistic membership of scenarios into each class. Due to the low level of emergent behavior in this system, scenarios were classified into each class with very high confidence. The classification of individual scenarios in both 5 or 8 latent classes was compared and 5 classes was selected due to the tendency to split 100% confident classification in the 5 class model into two or more groups with partial classification in the 8 class model.

The meaning of the different classes is not directly found but must be inferred from the resulting groups. That is, if the system is found to have 5 different classes of failure, providing a description of those failure classes cannot be generated from the analysis but requires expert knowledge. The normal approach in an LCA is to compare the probabilities of observing a particular variable (function) state within a class to develop descriptions for that class. However, given 58 function variables that each have 4 different states, this task can be very challenging and is not scalable to large systems. Instead, by comparing the classification provided by LCA to the clustering found through the modified k-means, these groups can be readily identified. This will be discussed in the next section.

5.3.5 Comparing and Validating Clustering Methods

The modified k-means clustering partitioned all of the unique scenario result states into 6 clusters. Each scenario result then has two properties: 1) The normalized total impact of that scenario; and 2) The distance of that scenario from the theoretical centroid of the cluster in which it belongs. This distance is a measure of functional similarity over the identified 58 functions in the space. Scenarios very near the centroid are the “typical" scenarios for that cluster.

The result of the LCA model is a predictive description of the latent failure behavior and the probabilities of observing a particular function's state. Comparing this model-based approach to the k-means approach has two benefits. First, LCA provides a mathematical validation of the partitioning of the k-means method when the two clustering methods agree. Second, the centroid of the k-means cluster can be used to identify the meaning of the matching LCA cluster.

In Figure 5.3, the k-means clusters are plotted based on total normalized impact and their distance from the cluster centroid. The classification of scenarios by the LCA and the modified k-means was inconsistent for 26 of the 3509 unique scenarios. The scenarios that were classified differently by the two methods are noted with diamonds in Figure 5.3. Because this plot compares similarity and normalized impact, some of the markers overlap. This means that these scenarios are equally different from the cluster centroid and have affect the same number of functions. It does not mean that the final system state of these scenarios is identical.

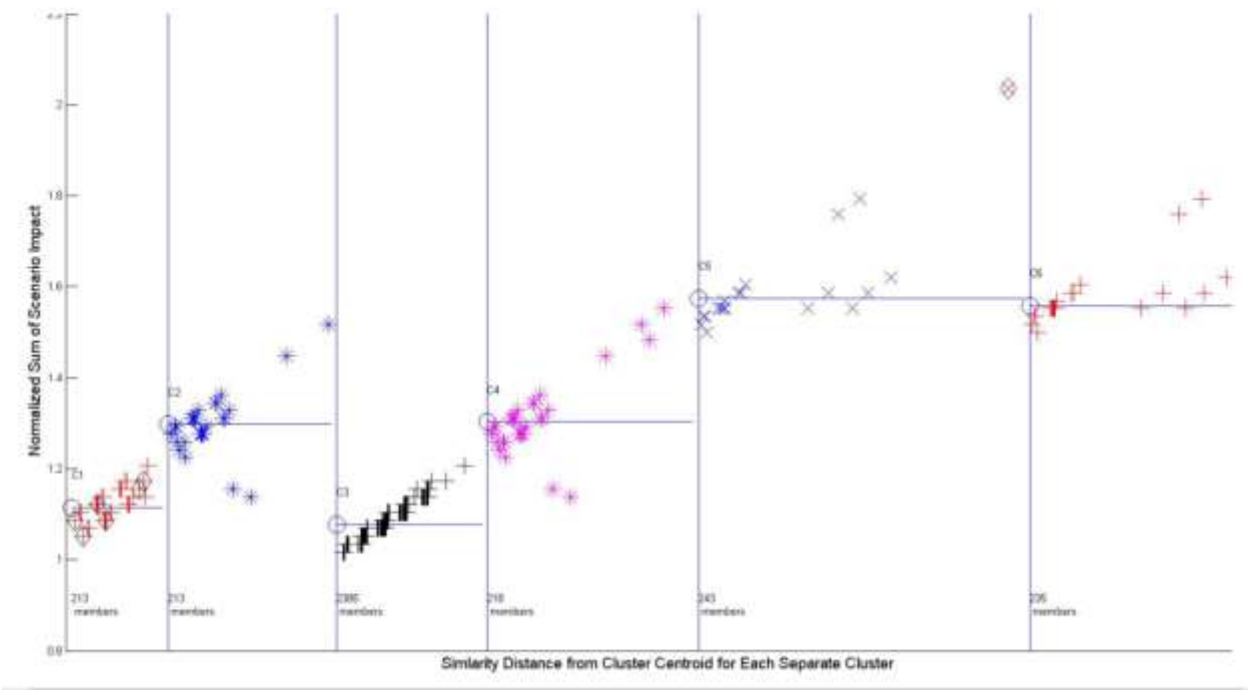


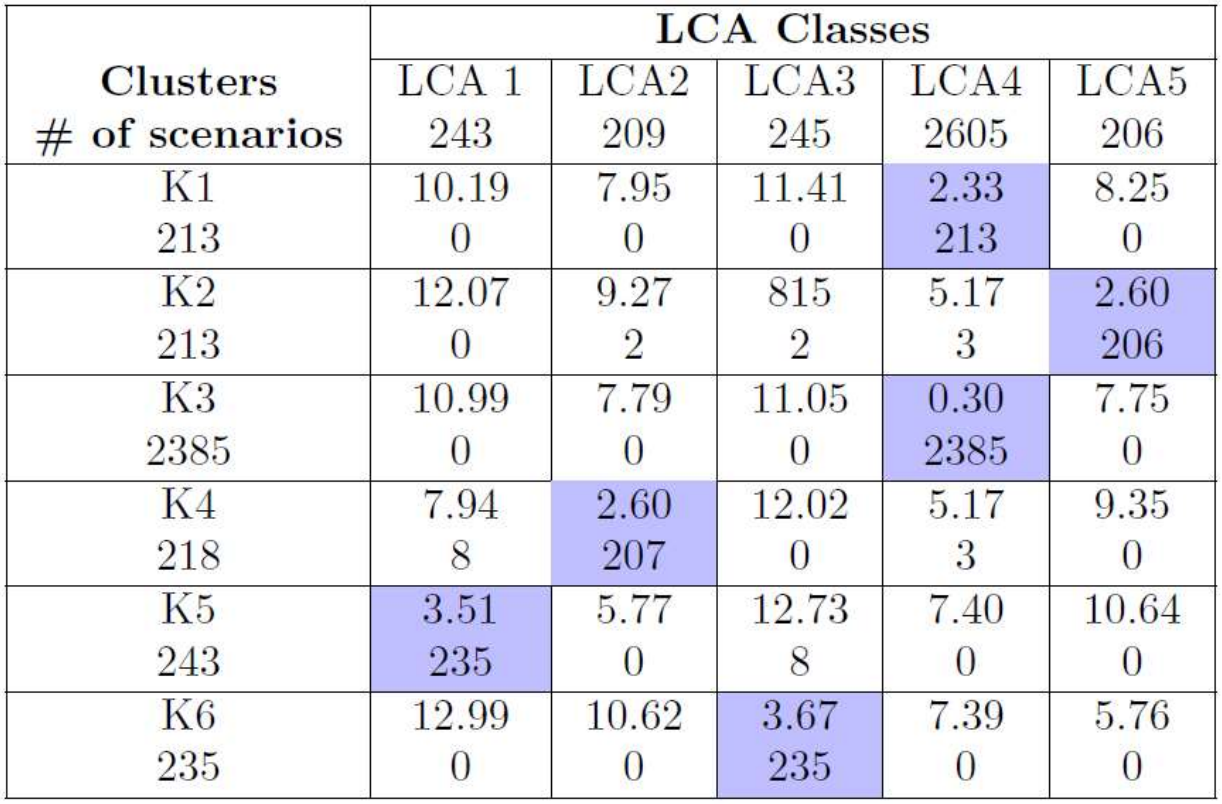
FIGURE 5.3: COMPARING THE CLUSTERING FOUND THROUGH THE K-MEANS AND LCA METHOD. DISCREPANCIES ARE MARKED WITH DIAMONDS. NOTE THAT SOME MARKERS OVERLAP.

There are two metrics for evaluating the consistency of the clusters found by the two algorithms. In Table 5.4 both metrics are shown for the 5 class LCA results and the 6 clusters from the k-means algorithm. First, to compare if the scenario populations are consistent, the union of cluster membership is evaluated. In Table 5.4, the number below each cluster name is the total number of scenarios classified into that cluster or class. The integers within the table show the membership union.

For example, 2 of the scenarios found in the third LCA class are also found in second cluster from the k-means algorithm. The numerical order provided by the algorithm is random. The second metric for comparing clusters is the distance between centroids. Since the LCA gives a probability distribution of health states for each function as the centroid, it cannot be directly compared to the single value centroids from the k-means algorithm. Instead, the centroid of the resulting classification from the LCA is used. That is, if a class contained scenarios 1-3, then the centroid is based on the centroid of those three scenario results and not the probabilistic centroid of the model of that class which the LCA algorithm used to fit scenarios 1-3.

In Table 5.4, the centroid to centroid distance is reported for each cluster and class as a real number in units of the distance between functional states. From Table 5.4, both metrics identify the same overlap in the k-means clusters and LCA classes (as indicated in the colored cells). Using this example it is clear that the fourth LCA class is the combination of first and third k-means cluster.

TABLE 5.4: COMPARING THE CENTROID TO CENTROID CLUSTER DISTANCE AND SCENARIO MEMBERSHIP OVERLAP



Clusters	LCA Classes				
# of scenarios	LCA 1 243	LCA2 209	LCA3 245	LCA4 2605	LCA5 206
K1	10.19	7.95	11.41	2.33	8.25
213	0	0	0	213	0
K2	12.07	9.27	815	5.17	2.60
213	0	2	2	3	206
K3	10.99	7.79	11.05	0.30	7.75
2385	0	0	0	2385	0
K4	7.94	2.60	12.02	5.17	9.35
218	8	207	0	3	0
K5	3.51	5.77	12.73	7.40	10.64
243	235	0	8	0	0
K6	12.99	10.62	3.67	7.39	5.76
235	0	0	235	0	0

5.3.6 Relating Clusters to System-Level Functionality

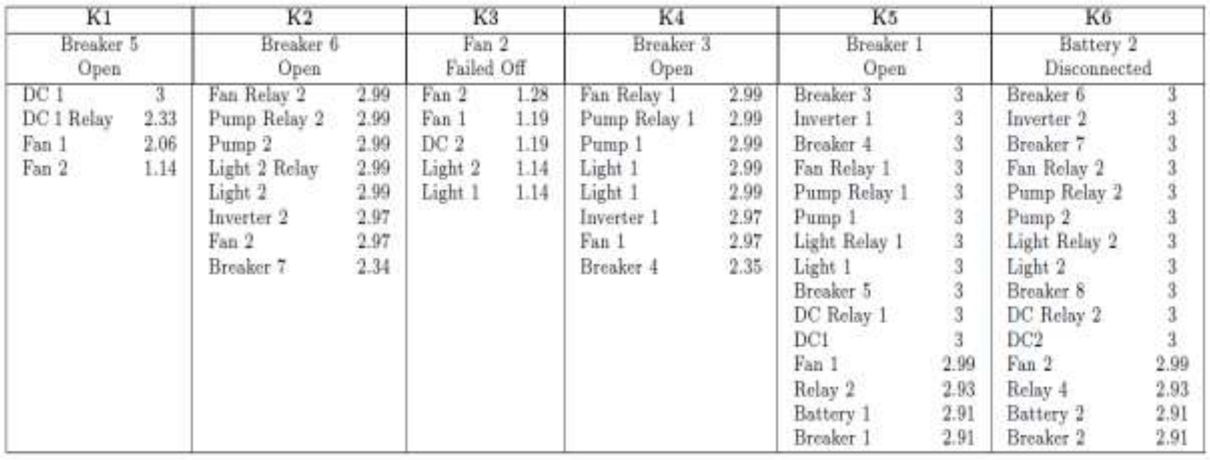
The centroid of each cluster found through the modified k-means analysis represents a point in the functional state space defined by 58 functions. In this space each function may have the value between 1-3 representing Nominal, Degraded, and Lost or Now Flow respectively. By observing what scenario is closest to the cluster centroid and what functional dimensions have the largest impact for a cluster, the meanings of the clusters become apparent.

In Table 5.5 the k-means clusters are sorted so that the highest functional impacts are grouped together. All component functions that do not appear in Table 5.5 have values near 1 and are considered predominately “Nominal” for the scenarios in that cluster. Additionally, the representative scenario for that cluster is also listed in the second row. The non-nominal functions are listed for each cluster along with the centroid's location along that functional axis. By looking at these characteristic functions and the health states for each cluster centroid, the clusters can be described in terms of their dominant system level effects. Thus each cluster is defined by a set of functions in some off-nominal health state.

While the clustering algorithm identifies that there are dependencies between these functions (and thus clusters them together), it can not directly reveal causality. For this reason we take the component-level functions identified in each cluster and use the model to organize the connectivity of the graph shown in Figure 5.4. Care should be taken not to interpret this as the direction of fault propagation. Instead Figure 5.4 shows the relationship between the functional dependencies in the clusters and the physical system architecture.

Finally, as can be seen in Table 5.5, the K3 cluster centroid does not have any characteristic functions in the degraded or lost state. This means that scenarios within this group have few failures that affect multiple functions and there is a minimal dependency between the faulty states of functions. Since the degraded and lost state functions are used to characterize the clusters, the K3 cluster is not included in Figure 5.4.

TABLE 5.5: OFF-NOMINAL FUNCTIONAL IMPACT FOR EACH CLUSTER AND REPRESENTATIVE SCENARIO



K1		K2		K3		K4		K5		K6	
Breaker 5 Open		Breaker 6 Open		Fan 2 Failed Off		Breaker 3 Open		Breaker 1 Open		Battery 2 Disconnected	
DC 1	3	Fan Relay 2	2.99	Fan 2	1.28	Fan Relay 1	2.99	Breaker 3	3	Breaker 6	3
DC 1 Relay	2.33	Pump Relay 2	2.99	Fan 1	1.19	Pump Relay 1	2.99	Inverter 1	3	Inverter 2	3
Fan 1	2.06	Pump 2	2.99	DC 2	1.19	Pump 1	2.99	Breaker 4	3	Breaker 7	3
Fan 2	1.14	Light 2 Relay	2.99	Light 2	1.14	Light 1	2.99	Fan Relay 1	3	Fan Relay 2	3
		Light 2	2.99	Light 1	1.14	Light 1	2.99	Pump Relay 1	3	Pump Relay 2	3
		Inverter 2	2.97			Inverter 1	2.97	Pump 1	3	Pump 2	3
		Fan 2	2.97			Fan 1	2.97	Light Relay 1	3	Light Relay 2	3
		Breaker 7	2.34			Breaker 4	2.35	Light 1	3	Light 2	3
								Breaker 5	3	Breaker 8	3
								DC Relay 1	3	DC Relay 2	3
								DC1	3	DC2	3
								Fan 1	2.99	Fan 2	2.99
								Relay 2	2.93	Relay 4	2.93
								Battery 1	2.91	Battery 2	2.91
								Breaker 1	2.91	Breaker 2	2.91

5.4 Clustering Analysis of the Electric Vehicle (EV) Health States

The ability to group the results of the electric vehicle health states help in identifying common and associated faults among components of the system. This analysis is done to further support the use of clustering analysis in different engineering systems. Once, failure is observed at the system level, the likely scenarios exhibiting similar behavior leading to such faults are effectively predicted by studying the clustering algorithm. A silhouette plot of clusters of the functional health states data from the electric vehicle components is shown below.

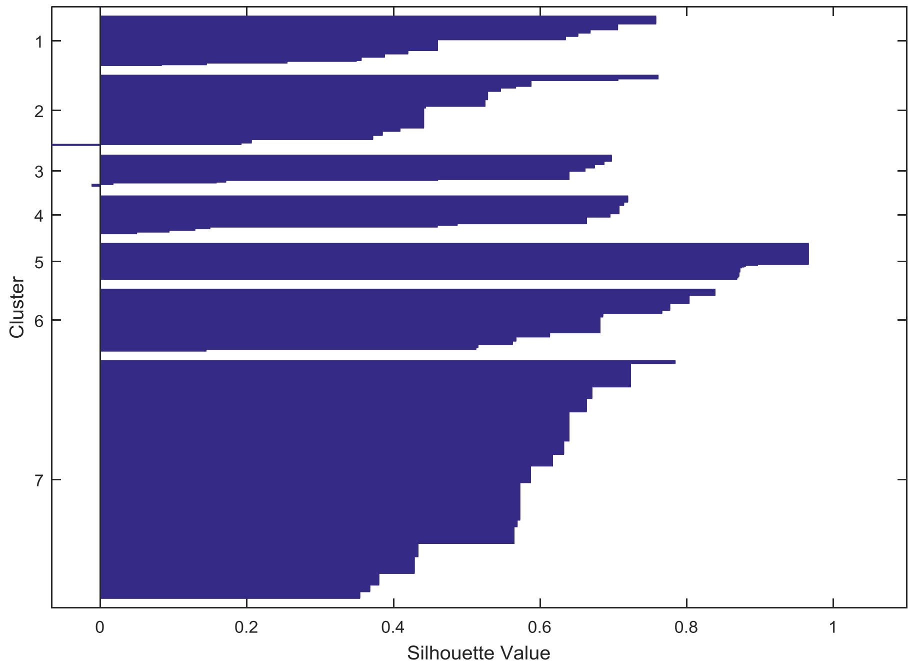


FIGURE 5.4: A SILHOUETTE PLOT OF CLUSTERING THE EV HEALTH STATE DATA

Figure 5.4 shows the similarity of points within each cluster. High silhouette values indicate how close a point is to members of its own clusters and different the point is to neighboring clusters.

Clustering results demonstrate similarities in fault properties among components that have no direct relationships or connections or similarity in the way they function. From EV health state data, it is observed that the differential and wheels failures are more prominent within Cluster 1, which is evident in the relationship between the two components. However, there are about 4 contributions of “Usable_degraded” and “Bad_degraded” failure modes for each of the Battery, Inverter, and the Electric Motor in Cluster 1. This shows that for most system level faults associated to a differential failure or wheel failure, there is a small chance of such system level faults being attributed to failure in any of the other three components highlighted above. This type of relationship can also be seen the cluster within clusters plots in Figure 5.5.

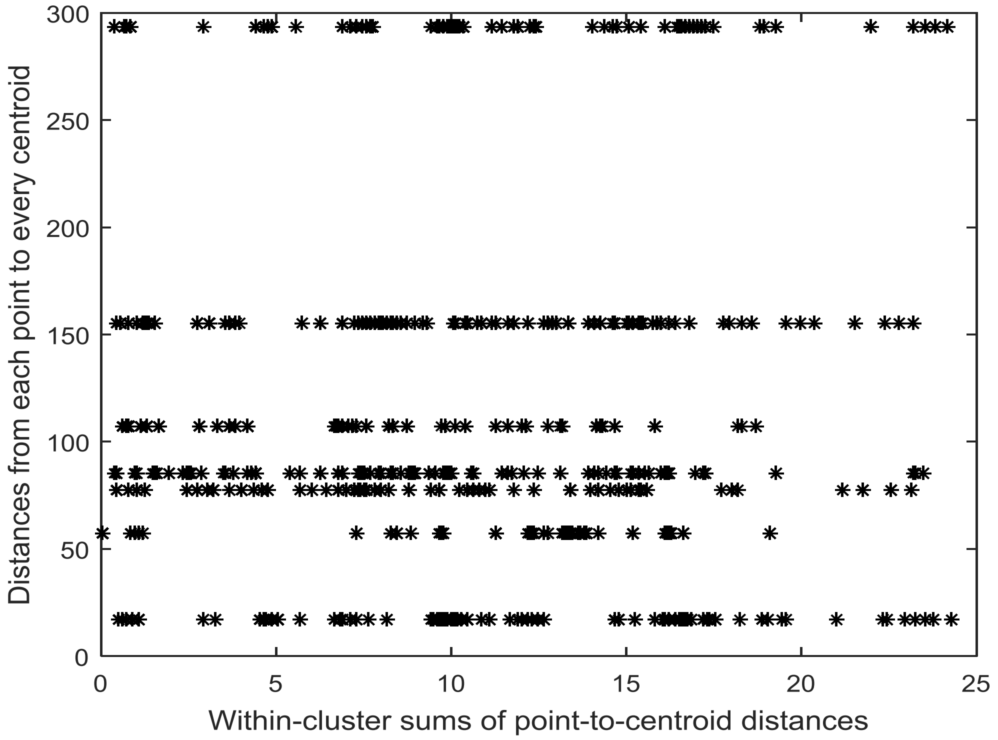


FIGURE 5.5: A PLOT OF CLUSTERS WITHIN CLUSTERS OF THE EV HEALTH STATES

5.5 Results

In this section we will present how the results of conducting the clustering approach address the three objectives of: 1) characterizing the impacts of a large number of failure scenarios; 2) Identifying the system-level meaning of those characterizations; and 3) determining how this analysis can be used to make system design decisions.

The first objective of characterization is accomplished through identifying an underlying pattern of failure behavior exhibited in the system states that result from numerous fault simulations. This underlying pattern of behavior is found through applying the Latent Class Analysis (LCA) to the set of unique systems states. The result of applying the LCA to the 3,509 unique systems states that result from fault scenario simulation for the example system best fit a model with 5 discrete classes of system failure. Further, the probability of scenarios fitting exactly one of the five classes is very high (most are 100%). This confirms that five different patterns of system failure emerge from the simulation of combinations of component fault behavior.

Because the LCA approach fits a structure to the data, each class is fully defined by the probability of a function being at a health state. The health state of a function as a result of simulating a scenario is deterministic and has a known value after simulation. However, the class of system failure is a model where each function has a probability of being at each health state. The system-level failure behavior classes are the result of the interactions of component behaviors. For this reason the five classes represent emergent failure behavior observed at the system level in the scenarios simulated. This does not represent all potential emergent behaviors of the system.

The clustering algorithm uses the simulation data and thus if the behavior is not present in the simulation it will not be identified by the algorithm. However, due to the large number of scenarios that form the data for each class model, this approach does provide some confidence that this system will not experience significantly different behavior. While the LCA-based clustering was able to address the first objective by finding underlying classes of system behavior to characterize scenarios, those classes must also be related to the system level functions of interest.

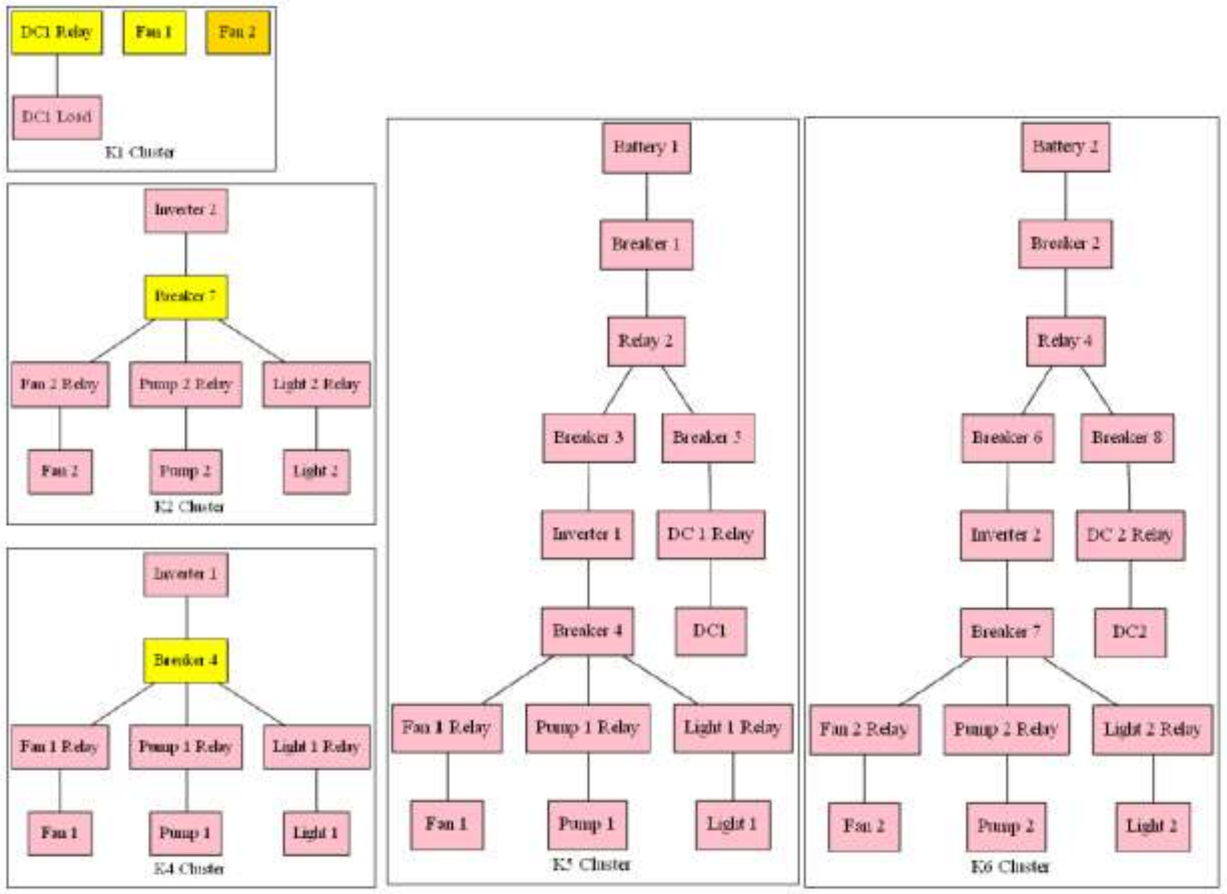


FIGURE 5.6: THE CLUSTERS IDENTIFIED THROUGH THE MODIFIED K-MEANS AND LCA ARE MAPPED TO THE SYSTEM MODEL.

The second objective, to identify the system-level meaning (for designers) of the classes of behavior, is accomplished using a k-means clustering on scenario impact similarity. By using the cluster centroids, each cluster is described with a set of functions and their health state. Limiting the focus to degraded and lost functionality provided five of six clusters that can be used to relate the system functionality to the scenario clusters. Figure 5.4 shows the characteristic functions and their health states for each cluster and uses the system model to identify physical connections. The third cluster centroid did not exhibit consistently degraded or lost functionality and is not included.

By comparing the system model to the cluster's representative functions, the relation to system-level functions begins to emerge. For example, the scenarios classified in Cluster 4 are predominately scenarios affecting the first load bank. When certain fault scenarios result in loss of power to that load bank the function of those components is lost or degraded. For this simple system this demonstrates that, without a prior knowledge of component connectivity, the clustering approaches identified behavior-based connections. For more complex systems with emergent behavior, these connections could be identified in components in different subsystems where interactions may be harder for designers to predict.

The third objective of this work was to determine whether the discrete failure behavior of the system identified through the clustering analysis could be used for system-level design decision making. As described in Section 2.4, the example system is designed to be fault tolerant where the software control attempts to operate as many of the loads as possible. The software control was designed to recognize and operate the system at the best available of the 7 potential states identified in Table 5.1. Comparing these 7 control action states to the clusters provides an assessment of the effectiveness of the system architecture and control. Table 5.6 shows how the degraded control states address faults from certain clusters.

One example of a design decision that could be made after application of this analysis is to redesign the architecture and control to address the individual load faults that are seen in Cluster 3. The application of this approach has shown that the current control method addresses four of the fundamental failure behaviors of the system, but has no specific action states to address the other two.

TABLE 5.6: RELATION OF DEGRADED SOFTWARE CONTROL STATES TO SCENARIO CLUSTERS

State 3 Batt1 → Load1	State 4 Batt2 → Load2	State 5 Batt1 → Load2	State 6 Batt2 → Load1
Cluster 6 Cluster 2	Cluster 5 Cluster 4	Cluster 4	Cluster 2

Finally, the small set of scenarios that k-means classified in Cluster 5 and that the LCA grouped in cluster 6 (see Figure 5.3), correspond to scenarios where both battery banks could provide no power. These special scenarios that are hard to cluster indicate important scenarios for the system designer to investigate. For this system, scenarios where both batteries are disconnected (and other similar scenarios) are unrecoverable by the software control. Based on the probability, and the consequence of those faults, designers may want to redesign the system redundancies.

5.6 Conclusion

This paper proposed two different approaches for clustering the results of a function-based failure analysis method in the early design stage. In contrast to others methods which focus on single faults or single failure scenarios, the goal of this work is to characterize a design's overall failure behavior. The results of implementing these clustering approaches on an example fault tolerant, software-controlled electrical power system (EPS) an the electric vehicle demonstrates the ability to both identify system-level failure behavior and utilize the classification of that behavior for decision making during the design process.

The first clustering approach was a modified k-means algorithm where the distance between failure scenarios was determined based on the functional similarity of the impact of those scenarios. This method partitions the fault scenarios into discrete clusters. Each cluster has a centroid which is the representative set of functions and their health states for that cluster. The second clustering approach was a model-based method that used Latent Class Analysis (LCA) to identify a latent variable with a set of discrete classes. The latent variable is a single unmeasurable variable that describes the system's failure state or failure modes. The LCA provides a probabilistic model that is used to characterize the system behavior. By comparing these methods the k-means clustering was mathematically validated when the scenario groupings agreed with the LCA classifications. Further, the challenge of describing the system failure modes found through LCA is addressed by using the centroids of the corresponding k-means clusters.

The example EPS describes how the designed control addressed some but not all of the system failure behavior modes. When informed by other variables such as cost, this could be used in a multi-objective decision making process. A future challenge that this work can address is that large-scale system modeling may be impossible at the component fidelity level. However, the LCA classes are models of the system state and could be used as abstractions for the component details. For example, the EPS can be described as having a few nominal modes and the identified five failure modes. This simplified model can then be incorporated into a larger model without the need to specify low-level component behavior.

Additionally, more work is needed in applying the presented methodology to complex systems to develop a relationship between the completeness of the analysis and the number and types of failures to simulate. The objective of this work is to aid designers in identifying the potential system-level failure behaviors and use the classification of those behaviors to improve system design. By using data analysis techniques on large sets of design-stage analysis data, designers can make better risk-informed decisions and provide stake-holders with safer systems.

References

[1] Jensen, D., Bello, O., Hoyle, C., & Tumer, I. (2014). Reasoning about system-level failure behavior from large sets of function-based simulations. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 28(4), 385-398. doi:10.1017/S0890060414000547

[2] Han, J., Kamber, M., and Pei, J., 2006, "Mining Frequent Patterns, Associations, and Correlations," Data Mining: Concepts and Techniques (2nd Ed., Pp.227-283).San Francisco, USA: Morgan Kaufmann Publishers.

[3] Estivill-Castro, V., 2002, "Why so Many Clustering Algorithms: A Position Paper," ACM SIGKDD Explorations Newsletter, 4(1) pp. 65-75.

[4] Lloyd, S., 1982, "Least Squares Quantization in PCM," IEEE Transactions on Information Theory, 28(2) pp. 129-137.

[5] Mardia, K. V., Kent, J. T., and Bibby, J. M., 1980, "Multivariate Analysis (Probability and Mathematical Statistics)".

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

[7] MacKay, D.J., 2003, "Information theory, inference and learning algorithms," Cambridge university press.

[8] Lazarsfeld, P.F., and Koch, S., 1959, "Latent Structure Analysis in Psychology: A Study of a Science," New York: McGraw-Hill.

[9] Vermunt, J.K., and Magidson, J., 2004, "Latent Class Analysis in The Sage encyclopedia of social science research methods," Sage Publications, Inc, pp. 549-553.

[10] Poll, S., 2007, "Advanced Diagnostics and Prognostics Testbed," 18th International Workshop on Principles of Diagnosis},

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

[12] Jensen, D., Hoyle, C., and Tumer, I. Y., 2012, "Clustering Function-Based Failure Analysis Results to Evaluate And Reduce System-Level Risks," ASME 2012 International Design Engineering Technical Conference and Computers and Information in Engineering Conference.

[13] 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},

[14] Sierla, S., Tumer, I. Y., Papakonstantinou, N., 2012, "Early Integration of Safety to the Mechatronic System Design Process by the Functional Failure Identification and Propagation Framework," pp. do:10.1016/j.mehatrons.2012.01.003.

[15] 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) pp. 051401.

[16] 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," San Diego, CA.

[17] 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},

[18] Linzer, D. A., and Lewis, J., 2011, "poLCA: An R Package for Polytomous Variable Latent Class Analysis," 42(10) pp. 1-29.

[19] Linzer, D.A., and Jeffrey Lewis, 2011, "poLCA: Polytomous Variable Latent Class Analysis," R package version 1.3.1. http://userwww.service.emory.edu/~dlinzer/poLCA.

[20] Team, R.D.C., 2011, "R: A Language and Environment for Statistical Computing," R Foundation for Statistical Computing, Vienna, Austria.

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

The work carried out in this research contributes new knowledge into how quality failure information is to be extracted from complex systems especially at the conceptual design stage or when there is limited information about the intended system.

The key aspects of this dissertation are (1) Understanding the significant contributions of existing function-based failure analysis methodologies, (2) Building accurate system model representations that capture actual failures at the conceptual stage, (3) Generating failure scenarios using failure analysis models to help forecast system performance, (4) Analyzing predicted system results to identify, qualify and quantify faulty component behaviors in order to improve system designs. These issues have all been addressed in this dissertation.

Chapter 3 provides a novel grouping for the existing Function Failure Identification and Propagation (FFIP) related researches that are available to the research community. The function-based failure analysis research were grouped as (1) Graph-Based, (2) Functional Failure Reasoning (FFR) Architecture, (3) Behavior Descriptions, (4) System Representation, (5) Results Analysis, (6) Capturing Emergent Behavior, (7) Socio-Impact (Human), and (8) Software Implementation. A review of the various researches has been presented as a form of FFIP history showing how the framework has contributed to improving prognostics and health management (PHM) methodologies.

Chapter 4 provides a detailed description of the recommended failure analysis methodology to be carried out in exploring a system for vulnerabilities. The impact of model details and abstractions on system modeling has been presented by carefully identifying key structures of system modeling. A description and importance of the Functional and Behavioral models were presented in Chapter 4. The modeling choices used in behavior modeling contributes significantly to the output obtained from the system. Also a synthesized conventional/traditional failure analysis tool (FMECA) that automatically introduces a mode number into the simulation environment has also been introduced.

The Primary, Secondary and Tertiary levels of abstraction for Function, Flow and Behavior models were investigated and their findings presented. It was concluded that at the earliest stage of the design process, using the primary levels of abstraction for modeling would give information on the predicted functional health states of the system. However, from the secondary levels of behavior abstraction to the tertiary level, adequate information on a system's functional health states, actual system performance and the effects of varying component parameter choices can be obtained. This information is suitable to designers at the early design stages before committing resources to certain designs.

Chapter 5 presented how to reason about FFIP simulation results using clustering analysis and latent class analysis. The two analyses that were utilized were explained in detail. Significant focus in this chapter was made on characterizing a design's overall failure behavior. The results of implementing the clustering approaches on the electrical power system (EPS) and the electric vehicle demonstrates the ability to both identify system-level failure behavior and utilize the classification of that behavior for decision making during the design process.

Other research focuses that can be carried on from this dissertation are recommended. The recommended researches for future work include: The investigation of failure in systems using continuous time-based system modeling in order to establish instances of fault initiation in the failure space; Utilizing a cyber-physical testbed for investigating actual system performance and for validating the FFIP framework; Establishing measures of confidence analysis on FFIP simulation results; Quantifying uncertainty in the parameter used in FFIP simulation models; and Investigating failure predictions in autonomous vehicles using function-based failure analysis.

Overall, this research effectively applies design theory and methodology concepts in designing, simulating and analyzing systems in order to help designer’s decision-making in building safer and more reliable systems.