12-2014

Utilizing Failure Information for Mission Assessment and Optimization for Complex Systems

Charlie DeStefano
University of Arkansas, Fayetteville

Utilizing Failure Information for Mission Assessment and Optimization for Complex Systems

Utilizing Failure Information for Mission Assessment and Optimization for Complex Systems

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Mechanical Engineering

By

Charlie DeStefano
University of Southern California
Bachelor of Science in Mechanical Engineering, 2010

December 2014
University of Arkansas

This thesis is approved for recommendation to the Graduate Council.

ABSTRACT

This paper presents a new failure analysis method, Failure Identification for Mission Analysis (FIMA), which performs a general failure analysis for the overall state of a system, as well as a mission-specific analysis that identifies how failures may have differing effects on the various mission tasks that a system must complete. The FIMA method is capable of being implemented at any point in the design process. During early design stages, the FIMA method will identify various qualitative failure scenarios based on programmed functional relationships and any number of initial failures wished to be simulated. The functional relationships for this method are unique in that along with traditional function-based failure modes, they also include manufacturing-based failure modes in each component's performance model. The models are then used to determine fault propagation paths as well as each failure scenario's criticality on the overall system performance. During later design stages, the FIMA method will introduce the usage of physics-based governing equations to more accurately identify the system's behavior during different failure scenarios. The FIMA method is unique in its ability to identify a specific failure scenario's effects on a system's overall performance and then apply this failure information to specific mission tasks. The FIMA method uses multiple metrics to determine the effects of a given failure scenario on a potential mission plan and then uses other unique metrics to assess and optimize a new mission plan based on the remaining tasks and the remaining functionality of the system's components. This method is demonstrated in two different theoretical case studies with experimental validation to be conducted in the future. The results of the first case study will show how the FIMA method is able to automatically identify a large variety of possible failure scenarios and their varying effects on the overall system's performance, while the second case study will show the FIMA method's mission analysis

capabilities by using multiple unique metrics for mission comparisons and optimizations during various potential failure scenarios.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

In a time when every new technology seems to be vastly more complex and sophisticated than its predecessor, it is obvious that the methods dedicated to modeling and analyzing these technologies must also become more complex and sophisticated. Technological advancements have allowed for the creation of many complex systems capable of performing a variety of tasks a variety of different ways, simplifying processes and helping automate many industries. Unfortunately, along with the benefits of being able to produce complex behaviors, complex systems also produce complex failures, and herein lies possibly the biggest challenge currently faced in understanding complex systems; complex failure analysis can be incredibly difficult due to the fact that the different functions of a complex system may all experience significantly different effects from the same failures. Therefore, the various possible use-cases of a complex system must be taken into account for accurate failure analysis. For example, if an airplane's landing gear fails in the open position, this failure would affect the performance while cruising, however, during takeoff and landing the failure would be inconsequential. Unfortunately, most current failure analysis methods are ill equipped for such comprehensive complex system analysis due to their mission independence. A mission is defined here as the system's high-level objective; for example, an airplane getting from point A to point B would be its mission. Mission tasks, on the other hand, are the discrete actions that must be done to complete a mission; for example, taking off, cruising a particular route, and landing would be three abstract mission tasks for an airplane. Current methods typically only identify potential failures and their likely effects on component and system-level behaviors in a general sense, but do not identify how the faulty behaviors will affect the system's ability to complete specific mission tasks, i.e. they do not take into account the three stages of flight in the previously stated example, and therefore, would not accurately be able to determine the severity of a failure on the various mission tasks.

According to the Failure Identification and Mission Analysis (FIMA) method presented in this paper, accurate complex system failure analysis depends on three factors: the internal system state, the external system state, and the internal system's mission. An internal system is any collection of interconnected functional entities, or components, such as an airplane. An external system is the environment with which the internal system interacts, and a mission is a task, or set of tasks, to be performed by the internal system, within the external system, during a specific time period. While the internal system boundaries and overall mission do not change for a given complex system, the external system boundaries and current mission tasks may frequently change depending on the system's current use. For example, an airplane would represent an internal system, and, regardless of what functions it is performing at a given time, it is always an airplane. Similarly, the airplane's mission of getting from point A to point B does not change. However, the current mission task changes from taking off to cruising to landing, and the external system changes with the changing weather that the airplane may fly through during its trip.

Understanding the possible complex uses of a complex system is where current methods primarily fall short in providing comprehensive failure analysis. Most current failure analysis methods and tools, many of which are discussed in Chapter 2, typically only identify potential failures and how they might affect component and system-level behaviors. These methods do not consider a failure's effect on specific missions however. This mission-independence greatly limits their analyses because during one mission task a failure may cause drastic changes to the system's performance, while during another, that same failure might not be noticed at all, and thus, it is vital to know how the complex system is to be used in the future, after a failure occurs, in order to accurately identify the high-level effects of the failure. For example, if a failure occurs in the drivetrain of a vehicle and causes the vehicle to no longer be able to turn left effectively, it might seem that the vehicle has suffered a critical failure because one of its main functions is removed. However, if the vehicle's mission was only to transport something down and back a straight path, then the turning failure would be inconsequential. Unlike current methods, the FIMA method presented in this paper would be able to recognize this and instead identify the failure as manageable for this specific mission. Likewise, for the same vehicle system, if the mission or external system were changed to include turns or obstacles, the proposed method would look for all possible paths that only require right turns and if no such solutions existed, then, and only then, would the failure be considered critical.

As well as knowing which functions are still achievable, it is also important to know if there are any functional or control redundancies that could help restore the lost functionalities; a functional redundancy is the utilization of healthy components in a new fashion in order to compensate for reduced or lost functionalities of unhealthy components, and a control redundancy is a parameter change to maintain nominal performance [1,2]. Understanding a system's functional and control redundancies is the first step towards optimizing a system's robustness. Robustness is a system's ability to adapt to failures and is vital in being able to extend its own lifespan and get the most use before any external involvement is needed, such as the replacement of faulty components. Robustness depends on a system's ability to adapt its behavior or its mission plan after a failure occurs in order to increase its overall functional efficiency. The human body is an excellent example of a robust complex system as it is constantly adapting to failures through the use of redundancies. For example, if you sprain your ankle, the body's pain sensors will feel the failure and adjust its functionality by walking slower and with a limp; putting more weight on the healthy leg is a functional redundancy and walking slower is a control redundancy. If these adjustments were not made and a normal walking style at a normal pace was continued then the injured ankle would be more susceptible to further injury and eventually total failure. Unfortunately, non-living complex systems cannot feel pain and thus will attempt to continue operating at full capacity on unhealthy components, increasing the likelihood and rate of further degradation until a total failure occurs. Therefore, new strategies, such as those presented in this paper, should be implemented in order to identify when, where, and which redundancies are needed.

The FIMA method described in this paper is broken down into three phases of modeling and analysis to be used in the design process depending on how much system information is known; these phases are (1) qualitative, (2) quasi-quantitative, and (3) quantitative. The qualitative analysis phase focuses on abstract failure modes, such as 'Nominal,' 'Degraded,' and 'Defective,' and is designed to determine potential failure scenarios, identify the difference between initial and propagating faults, and ultimately, provide a qualitative complexity gauge for conceptual design comparisons. The quasi-quantitative phase also focuses on abstract failure modes but allows for different degrees and types of degradation, such as a channel being '10% too big/small.' This phase also incorporates physics-based equations to more accurately define a system's behavior during nominal and faulty conditions, which is then used to assess and optimize the system's performance for specific mission plans. Lastly, the quantitative phase implements specific failure modes, such as '20% Wear,' and utilizes a physical testbed for experimental validation. By providing support for all three phases of analysis, the FIMA method will allow for a much more streamlined analysis process by allowing the same adaptable model, which simply expands and becomes more comprehensive with each new learned piece of information about the system, to be used throughout the entire design process.

A major issue with most current failure analysis methods is that during the design process the majority of failure information is only used to improve a system's reliability, i.e. its probability to function given time and scenario before a failure occurs [3,4]. Unfortunately, however, no matter how reliable, all systems are guaranteed to eventually fail, and therefore, it is important for designers to begin using a system's robustness [5,6], i.e. its ability to adapt to failures, just as seriously as its reliability, and the proposed FIMA method attempts to help with this problem. Utilizing the FIMA method allows for comprehensive models and simulations of failure scenarios in complex systems, but also allows for specific mission assessments and optimizations based on a system's robustness. This paper will explain the detailed theory and methodology for the qualitative and quasi-quantitative phases of the FIMA method, as well as providing example case studies for each phase, in Chapters 3 and 4, respectively, where MATLAB/Simulink software was used to create the models. The general description of the fully quantitative phase will then be discussed in the "Future Recommendations," found in Chapter 6.

CHAPTER 2

BACKGROUND

The proposed method attempts to expand upon the research in model-based failure analysis by incorporating failure modes for not just components, but also manufacturing processes and environmental influences. Function-based modeling enables a broad approach to failure analysis as it can be applied during early design stages when a system's detailed information may not yet be known. Function-based modeling can be used to examine multiple faults [7-32], and some can also help simulate propagating faults [7,8,19,32]. By using function-based models, various components of different technology types, manufacturing processes, and physical connections, or lack of connections, between components can all be represented in the same failure simulation analysis. The functional performances of the individual components, as well as the overall functional performance of the system, are then identified through discrete failure scenario simulations. This paper's research is based on this foundation of function-based design and failure analysis.

Function-based modeling starts with a very abstract description and gets progressively more detailed. Function-based modeling classifies system functions into groups and subgroups, with abstract functional relationships connecting the high-level groups, and more detailed functional relationships connecting the low-level subgroups. These function groups are repeatedly divided into more and more detailed subgroups until the functions are detailed enough that specific components can be chosen to satisfy them [15]. For the purposes of the FIMA method, components are already chosen, however, the designer is responsible for determining their detailed functional relationships based on how each component's function is related to the desired overall system performance. Then, depending on how much information is known, each component can be represented by either qualitative or quantitative equations. When little information is known, qualitative math is used so that abstract values can be given to various components to represent certain abstract failure states with different degrees of degradation, which then allows for qualitative comparisons to be made between components [33]. For example, if a value of 5 is said to represent nominal performance and 0 represents defective performance, with the in between values representing degradations, then, if component(A)=5 and component(B)=2 then component(B) is said to be more degraded than component(A). Once sufficient information is known, however, more quantitative analysis with actual quantitative equations can be used to determine specific outputs. For example, a simple circuit would be given the equation V=I*R, but if the resistor lost 10% of its functionality, the new equation would become V=I*(.9R).

Two of the main failure analysis methods currently used in industry are Failure Modes and Effects Analysis (FMEA) [34,35,36,37] and Failure Modes, Effects and Criticality Analysis (FMECA) [38,39]. However, these methods do not use function-based modeling. These methods use only static failure definitions to identify the causes, effects, probabilities, and criticality of known failures that a system may experience. However, the analysis is explicitly on how a component's failure will affect that single component's performance, while the propagating effects on the other components' and the overall system's functions are not explored. Moreover, different use-cases, or missions, are not explored and therefore, these methods are limited when it comes to analyzing complex systems with complex behaviors. Also, because these are static methods that do not incorporate behavioral models, they rely on a fair amount of detailed information to be known about the system in order to provide any useful analysis, which means they cannot typically be applied until late in the design process.

On the other hand, existing methods that do explore model-based failure analysis include the Function-Failure Design Method (FFDM) [13,21], Function-Failure Identification and Propagation (FFIP) [8,19], and Risk in Early Design (RED) [22,24,25]. These methods focus on function-based modeling to identify the behavioral effects of a system's possible failure modes, as well as how failures might propagate through the system. Function-failure methods generally only care about whether or not an individual component is experiencing a functional failure, such as a valve not opening all the way, and how this affects the overall system performance, while the causes of these failures may be ignored. Therefore, less detailed information is required to make these models, allowing them to be implemented early in the design process. Some methods, such as FFIP, also provide a more expansive analysis by including physics-based behavioral equations in order to more accurately define failure effects on system performance. Unfortunately, these methods, similar to FMEA and FMECA, do not consider the various potential missions that a system may be asked to perform, and therefore, they provide only a limited understanding when it comes to complex systems and their complex behaviors.

Other forms of analysis include dependency-based methods, such as Fault Tree Analysis (FTA) [40], which is a top-down approach that starts with a component's possible failures and works downward to determine all possible causes for each failure. The process begins by identifying the different ways that component functions may be limited. For example, if analysis were being done on a vehicle, the process would begin by identifying potential high-level failures, such as a transmission failure. Then, the approach would be to work downward with less abstract descriptions to identify the different possible causes for this failure, such as the gears being broken, the clutch being jammed, or through power loss, until all possible causes and the responsible components have been described in sufficient detail. This same style of identifying causality will be used by the FIMA method, however it will be used for determining functional relationships for fault propagation purposes, where instead of choosing effects and working downward to identify causes, the FIMA method chooses high-level system functions and works downward to identify components that provide these functions. Also, while this paper explores modeling with MATLAB/Simulink, the modeling of functional relationships could also be done using formal modeling languages, such as SysML. [41,42]

Other methods introduce functional and control redundancies, as well as other criteria to help enhance a system's robustness [1,2,3,4]. Redundancies, as previously mentioned, are when healthy components take on extra responsibilities and functionality in order to compensate for unhealthy components. Self-maintenance machines are one such technology that use control and functional redundancies [1,2], however they only identify these once the design is already completed and cannot identify any of the backup procedures without first knowing the potential failure scenarios or failure probabilities, and therefore cannot be applied until such information is known, which is typically not until late in the design process.

CHAPTER 3

PHASE I: QUALITATIVE ANALYSIS

3.1. Phase I: Overview

3.1.1. General Theory

Phase I of the FIMA research was previously published and presented by the author at the 2014 ASME IDETC/CIE conference as a stand-alone research project, and therefore, all information in Chapter 3 can be also found in that conference paper referenced in the "List of Publications" section [DeStefano & Jensen. 2014].

The following sections detail the proposed approach for performing qualitative function-based failure analysis on complex systems during early design stages. This analysis can be used during any design stage, however, typically it would only be used during early design stages of novel systems when detailed information about the system is not known, i.e. governing equations or failure modes. If such detailed information were already available it would likely be more beneficial to skip straight to Phase II for quasi-quantitative analysis. Despite only providing a qualitative analysis, Phase I of the FIMA method provides valuable abstract failure information, such as possible failure scenarios, fault propagation paths, critical versus manageable failure scenarios, and complexity gauges of the design. Phase I analysis is to be used to compare multiple potential concept designs in hopes of being able to identify the best options, or at least being able to eliminate the worst.

During this phase, the first step is to create an abstract model of functional relationships and dependencies between the system's components. These functional relationships are the framework for further, more detailed analysis later on. The models are not based on internal system structure, but rather only on functionality, as well as any other factors that may affect a system's performance, such as a component's manufacturing process or environmental influences. Structure is not valued here because in complex systems where components can transfer electrical, material, or signal information, just because two components may be next to each other structurally, they do not necessarily have any interaction with one another. Therefore, only a component's functionality is assessed.

The FIMA method uses Simulink state-flow models to identify the functional relationships and the different potential failure modes, and uses MATLAB coding to initiate failure scenario simulations. MATLAB and Simulink were used as the modeling software for this research, however the FIMA method should also be able to be applied using any other state-based, signal processing software. Each system component within the Simulink model is connected to one another based on their functional relationships, i.e. the components that are directly dependent on the performances of the other. Each component has two types of Simulink diagrams: a state-machine diagram and a failure-logic diagram. State-machines allow for a component to switch between any of its potential performance states, such as "Good Performance" or "Faulty Performance," and then provide a performance value based on which state the component is currently located. On the other hand, failure-logics are essentially the reverse, as they provide the state of a component based on its individual performance value as well as the performance values of all of its dependent components. Simply put, state-machines are used to simulate multiple failure scenarios by identifying different combinations of performance states throughout the system, and failure-logics are used to identify where the failures have occurred in the system for a specific scenario. During the Phase I qualitative analysis, component state-machines contain three potential performance states: "Nominal," "Degraded," and "Defective." While having only three performance states is very abstract, as most real-world components have more than one way to be "Degraded" or "Defective," the more that is known about a system, the more detailed and extensive these faulty states can become; this expansion will be explored in more detail in the Phase II quasi-quantitative analysis method found in Chapter 4.

3.1.2. General Methodology

Once a system is chosen for modeling, the first step is to create a top-down functional relationship diagram by identifying the highest-level component, or components, that best express the overall performance of the system and then working downward to determine all dependent functional relationships. For example, a functional relationship of a car would start with the wheels, because they are the component that actually allows the vehicle to move, and then the wheels would be dependent on the driveshaft, which would depend on the power system, and so forth, until every component was represented and related through their functionality. Or for a simpler example, take a three component system where component(A) depends on component(B), which depends on component(C), as shown in Figure 3.1.

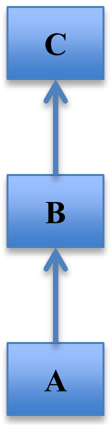


Figure 3.1 – Functional Relationships between three components (C depends on B, which depends on A)

Next, Simulink models are created based on these functional relationships. For qualitative models, the Simulink state-machines only contain abstract performance values based on the components' different performance states and their functional relationships. The functional relationships flow up, so analysis begins with the bottom components, which is component(A) for the aforementioned example. If component(A) is in the "Nominal" state it would be given an abstract performance value, which for the purposes of this example we will set equal to 10. On the other hand, if component(A) is in the "Degraded" state its performance value will be 9, or the "Nominal" performance value minus one, i.e. 10-1=9, and if it is "Defective" it will be 0. Then, component(B)'s "Nominal" state will inherit component(A)'s performance value and then proceed through the same subtraction for its "Degraded" state, while a "Defective" state is always given a performance value of 0. That is to say that if component(A) is "Nominal," i.e. has a performance value of 10, and if component(B) is also "Nominal," then its performance value would also be 10 or it's "Degraded" value would be 9, but if component(A) is "Degraded," i.e. has a performance value of 9, then if component(B) is "Nominal" its performance value would be 9 and its "Degraded" value would be 8, and this process progresses for any number of components (A-Z). So, for the example three-component system in Figure 3.1, if all three components were "Nominal" then they would all three have performance values of 10. However, if component(A) and component(B) are degraded from extended use, but component(C) is brand new, then component(A) would be labeled "Degraded" with a performance value of 9, component(B) would be labeled "Degraded" with a performance value of 8, and component(C) would be labeled as "Nominal" with a performance value of 8. This subtraction method is used so that if a single component becomes "Degraded" the reduced performance value will propagate to the next dependent component, however it will not necessarily be labeled as "Degraded," it would simply have a lower “Nominal” value; a component is only labeled “Degraded” if its decreased performance value is caused by an initial internal failure rather than by fault propagation. By making this distinction, we can easily identify if a component is affected by an initial or propagating failure based on the combination of its performance state and performance value. This allows for a quick check of how many initial faults are present within the system by comparing a component’s reduced performance value with the designated nominal value. For example, if a component, such as component(C) in the previous example, shows a “Nominal” state with a performance value of 8, instead of 10, then this indicates that there are 2 preceding failures in the system, and the component’s failure-logic diagram could then be used to identify where these initial failures occurred. More detailed examples of state-machines and failure-logic diagrams will be explored in the following case study.

Once the Simulink model is completed, the next step is to develop a code in MATLAB that can simulate the Simulink model to automatically generate possible failure scenarios by identifying all possible combinations of the programmed failure modes. A detailed example of this modeling process will be explored in the case study discussed in the following section.

3.2. Phase I Case Study: MEMS DNA Sequencer

3.2.1. Case Study: Overview

Along with advanced technologies becoming more and more complex, many are also becoming smaller and smaller; one such product area is known as MEMS, or micro-electromechanical systems. Many of these devices deal with electrochemical machines and quantum physics, which can cause difficulty in understanding potential failure modes during early design phases. This lack of detail when it comes to electrochemical devices and quantum physics is a perfect example of when the FIMA method’s Phase I qualitative analysis can be highly beneficial. Therefore, the case study presented in this paper will introduce such a device: a MEMS Nanochannel DNA sequencing device [43,44].

The device, i.e. the internal system, as seen magnified in Figure 3.2, is comprised of a microchannel inlet and a microchannel outlet connected by a nanochannel, represented by the dashed line, along with transmitting and receiving electrodes on either side of the nanochannel; the image depicts five electrodes, however, this paper’s model only deals with one for the sake of simplicity. Including all five electrodes in the model would more accurately represent the system’s robustness due to the enhanced functional redundancy, however for the purposes of Phase I, the goal is to simply show that the qualitative process works for basic failure identification and mission assessment at even such a low level of detail. More detailed analysis that will incorporate redundancies for mission optimization is reserved for the case study seen during Phase II in Chapter 4.

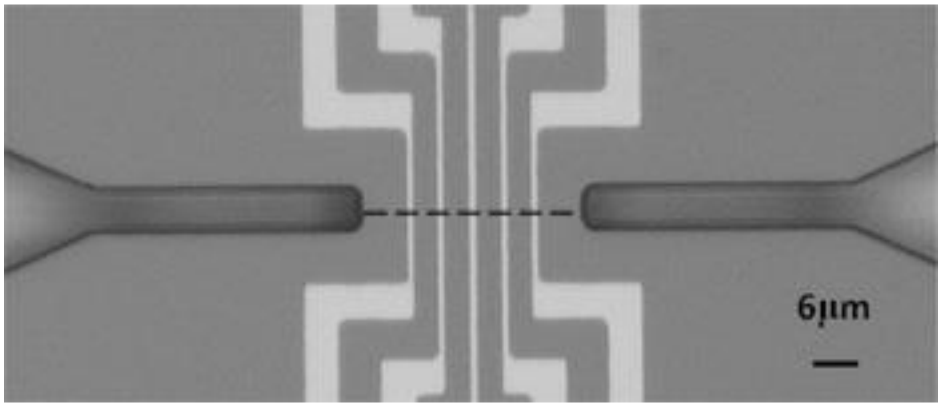


Figure 3.2 - MEMS Nanochannel DNA Sequencer. [43,44]

The modeled device uses the quantum principle of electron tunneling to detect and sequence a strand of DNA that traverses through the nanochannel, however, in the current developmental stage of this device, nanobeads are used to represent DNA strands. The theory behind this device is that because DNA consists of four nucleotide bases, thymine (T), adenine (A), cytosine (C), and guanine (G), each with a different internal resistance, when the transmitting electrode emits a current at a passing DNA, or nanobead, the receiving electrode will register different voltages based on which nucleotide base is passing by, blocking the transmitted current [41,42]; this goal of identifying and sequencing the DNA would be considered the system's mission. Due to only a limited number of components, this device appears to have a fairly low complexity. However, the case study will show that the Phase I model is able to simulate a very large number of possible failure scenarios, which will illustrate how even seemingly simple devices can still have more failure scenarios than a human designer could ever think of within a reasonable timeframe, without the use of simulations.

3.2.2. Case Study: Methodology

First, the system's functional relationships had to be determined. Because this is a device with no moving parts, it can be more difficult to understand how failures might propagate, as an individual component has no means of influencing any other component through physical contact, rather only through complex combinations of electrical, material, or signal flows. However, it is because of this fact that this device is a perfect example for the proposed FIMA method's qualitative analysis. Applying the proposed method to this device demonstrates how structural relationships do not matter, but instead, it is only functional relationships that matter for determining the system's overall performance. In this case study, the focus is on how each component's functional performance affects the final measurement of the DNA, therefore making the receiving electrode the highest-level component. A simple representation of the functional relationships for the device can be seen in Figure 3.3. The diagram also displays how the Simulink model will be approached; a component's influence flows up, meaning components in the diagram are dependent on the performance of those connected from below.

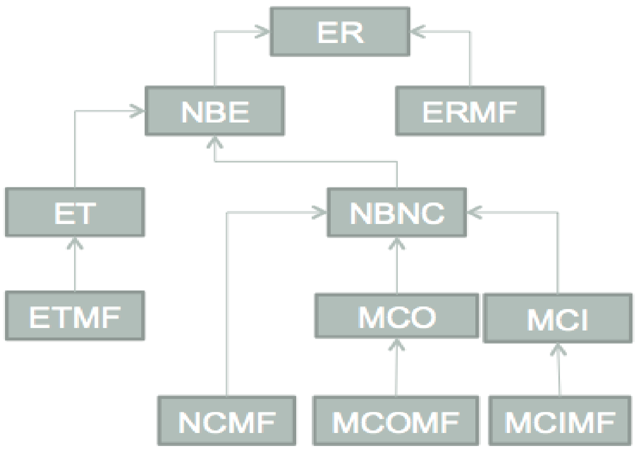


Figure 3.3 - Functional Relationship Diagram for Nanochannel DNA Sequencer

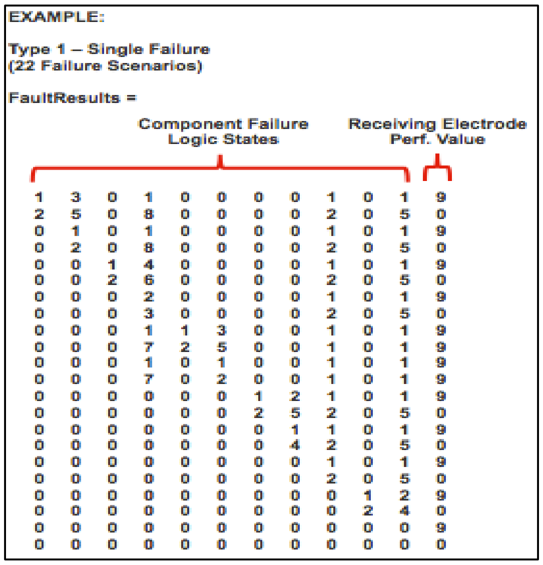
Figure 3.3, identifies how the functional relationships between the components' manufacturing and performance states will be modeled, where a state-machine and a failure logic will be created for each block. Beginning at the low-level components and moving up, it can be seen that the state of the microfabrication of the nanochannel (NCMF) directly affects the performance state of the nanochannel (NBNC). The performance state of the nanochannel is also affected by the microfabrication and performance states of both the microchannel inlet (MCIMF/MCI) and the microchannel outlet (MCOMF/MCO). This is due to the fact that if the inlet or outlet is faulty, such as by being clogged, the result would respectively be either no nanobeads being able to enter or no nanobeads being able to exit the nanochannel, both of which would produce a non-nominal state within the nanochannel. Next, the state of the nanochannel, along with the microfabrication and performance state of the transmitting electrode (ETMF/ET), combine to determine the conditions of the nanobead (NBE). For example, if the nanochannel is not the right size or if the transmitting electrode does not emit the correct current then the nanobead will not be able to be accurately measured. Finally, the performance state of the receiving electrode (ER) is affected by its microfabrication (ERMF) and the nanobeads' conditions when they are between electrodes. Accurately identifying these functional relationships is vital in being able to properly detect the difference between initial and propagating failures. For example, if the nanochannel were "Degraded," such as not being the correct size, this would result in an inaccurate measurement taken by the receiving electrode without any failure to the receiving electrode itself. Similarly, if the transmitting electrode is "Degraded" and emits a faulty current, then the receiving electrode will again have inaccurate measurements without any failure to itself, or, lastly, the nanobead itself could cause an inaccurate receiving electrode measurement, such as by being damaged or moving too fast. All three listed examples of propagating failures, as well as initial failures of a degraded manufacturing or performance of the receiving electrode itself, will result in inaccurate measurements, however, by accurately identifying the functional relationships and through the use of Simulink state-machines and failure-logic diagrams the FIMA method should be able to effectively identify the root cause of the faulty measurements.

A Simulink diagram is created using state-machine and failure logic diagrams instead of the blocks used in the functional relationships diagram. An overview of the entire Simulink model can be seen in Figure A1 in the Appendix, and examples of two different performance state-machines and a failure-logic diagram can be seen in Figures A2, A3 and A4, respectively. State-machine and failure-logic diagrams consist of two main parts: state-blocks and transition lines. The values and equations in each state-block represent the output performance states and values for that component, and the values and equations on each line represent the inputs that determine which state-block to activate, i.e. the state-machine begins at the default value, represented by a line open at one end and connected to a state-block on the other, then the model observes the inputs, finds a matching transition line, and then transitions the component into that state.

After the Simulink model is finished, the next step is to write a MATLAB code that simulates all combinations of potential performance states found in every component, where each possible combination is known as a failure scenario. As seen in Figure 3.3, the overall system is comprised of 11 parts, counting both manufacturing processes and functional performance for each internal system component, as well as the nanobeads. However, the Simulink model includes a state-machine and failure-logic diagram for each component and therefore, there are 22 parts, as seen in Figure A1. The system inputs, seen as the small circles feeding into the left side of each component in Figure A1, represent the performance states being designated for each component during any given scenario. The system outputs, seen as the small circles exiting from the right side of the components, represent the components' failure-logic and performance values. If all 11 components failed simultaneously, there would be a massive number of possible failure scenarios, and while simulating this would be possible, it does not seem probable as a real-world scenario. So, for the purposes of this paper, the MATLAB code is designed to simulate all possible failure scenarios based on the programmed functional relationships for single, double, and triple type failure scenarios i.e. all failure scenarios initiated by any combination of one, two, or three initial component failures, respectively.

3.2.3. Case Study: Results and Discussion

After all simulations were complete, it was determined that there were 22 possible failure scenarios for single failures, 220 failure scenarios for double failures, and 1320 failure scenarios for triple failures. Also, by examining the performance states of the highest-level component in the functional relationship diagram, i.e. the receiving electrode, it was determined that for the single type simulations, 9 of the 22 failure scenarios, or roughly 41%, resulted in critical failures. Critical failures are classified as when the receiving electrode's performance value is 0, either through initial or propagating failures, and manageable failures are when the receiving electrode's performance state is not 0; an example of the output data set for single failures can be seen in Figure 3.4, where each row is a different failure scenario and the columns are the failure-logic states for each component and the performance value for the receiving electrode. For the double type simulations, 144 of the 220 failure scenarios, or roughly 65%, resulted in critical failures, and for the triple type, 1056 of the 1320 failure scenarios, or roughly 80%, resulted in critical failures. The reason every "Defective" performance state did not result in a critical failure in this example was because the microchannel outlet was given time-based failures. For example, if the microchannel outlet was full or experienced a blockage, the effect would not instantly be seen in the nanochannel. Only after a certain amount of time would the blockage back up into the nanochannel, and until that time, accurate measurements would continue to be made.



```
EXAMPLE:

Type 1 – Single Failure
(22 Failure Scenarios)

FaultResults =

                  Component Failure          Receiving Electrode
                    Logic States                 Perf. Value

   1   3   0   1   0   0   0   0   1   0   1   9
   2   5   0   8   0   0   0   0   2   0   5   0
   0   1   0   1   0   0   0   0   1   0   1   9
   0   2   0   8   0   0   0   0   2   0   5   0
   0   0   1   4   0   0   0   0   1   0   1   9
   0   0   2   6   0   0   0   0   2   0   5   0
   0   0   0   2   0   0   0   0   1   0   1   9
   0   0   0   3   0   0   0   0   2   0   5   0
   0   0   0   1   1   3   0   0   1   0   1   9
   0   0   0   7   2   5   0   0   1   0   1   9
   0   0   0   1   0   1   0   0   1   0   1   9
   0   0   0   7   0   2   0   0   1   0   1   9
   0   0   0   0   0   0   1   2   1   0   1   9
   0   0   0   0   0   0   2   5   2   0   5   0
   0   0   0   0   0   0   0   1   1   0   1   9
   0   0   0   0   0   0   0   4   2   0   5   0
   0   0   0   0   0   0   0   0   1   0   1   9
   0   0   0   0   0   0   0   0   2   0   5   0
   0   0   0   0   0   0   0   0   0   1   2   9
   0   0   0   0   0   0   0   0   0   2   4   0
   0   0   0   0   0   0   0   0   0   0   0   9
   0   0   0   0   0   0   0   0   0   0   0   0
```

Figure 3.4 – Single Type Failure Scenario Results

Ultimately, the proposed method was able to simulate the case study model for all single, double, and triple failure scenarios. The example device used in the case study, which was only comprised of five main components, might intuitively be thought of as a fairly simple design, without that many possible ways of failing. However, this paper has demonstrated how even seemingly simple devices can, in fact, have a vast array of possible failure scenarios, many of which result in critical failures. This shows how with so many possibilities for failure, even on a seemingly simple design, a human designer could potentially overlook critical failure scenarios that may eventually cause costly redesigns down the line. Using the proposed method, along with some additional clustering analysis of the results, designers should be able to determine valuable failure information early on in the design process. Such information includes how each component's functional performance relates to the design's overall performance, how many failure scenarios are possible, how many failure scenarios are critical vs. manageable, which components are most sensitive, i.e. involved in the most failure scenarios/most critical failure scenarios, which manufacturing processes are most critical, and many others. Also, despite only one design having been talked about in regards to this device, this qualitative analysis would be very useful when there are multiple potential designs in question.

In general, designers want to limit complexity as much as possible in their designs, and this type of qualitative analysis provides two very useful pieces of complexity information: (1) the higher the involvement of a component, the higher the component complexity, and (2) the more possible failure scenarios, the higher the overall device complexity. With these two criteria, a designer can compare two or more potential device designs to determine which design has the most balanced component involvement and the fewest failure scenarios. Unfortunately, this device did not provide a very adaptable system model, as it only had one possible mission, i.e.

sequence DNA, and did not have any functional redundancies, due to limiting the number of electrodes within the model, and therefore, the full extent of the FIMA method's mission assessment and optimization capabilities were not completely explored in this case study. Nevertheless, the Phase I qualitative analysis method has effectively laid the ground work for further, more comprehensive analysis, which will be explored in the following chapter where the quasi-quantitative analysis method will be explained, as well as demonstrated on an adaptable complex system in order to effectively show the mission analysis capabilities of the FIMA method.

CHAPTER 4

PHASE II: QUASI-QUANTITATIVE ANALYSIS

4.1. Phase II: Overview

4.1.1. General Theory

While Phase I of the proposed FIMA method effectively laid the framework for more detailed models and simulations, as well as effectively provided a variety of valuable early-stage failure information, the main benefits of the FIMA method begin with the Phase II quasi-quantitative analysis. During Phase II, additional information, such as a system's behavioral equations and conditional functional objectives, are to be added to the already created, qualitative model. During this phase, both the Simulink models and MATLAB code must be updated to allow for more complex behaviors and mission plans. Governing equations are added to each Simulink state-machine in order to more accurately describe how different types of failures may influence the system's behavior. Such behavioral equations are essential in understanding how failures at varying levels of severity, not just "Degraded" and "Defective," may propagate through complex systems.

During the Phase I qualitative analysis the only concern was *if* a failure occurred, not *how* it occurred and therefore, only one "Degraded" and one "Defective" performance states were created. This was due to the assumption that only limited behavioral knowledge of the components would be known during the early design stages. Once this information is known, however, each "Degraded" and "Defective" performance state will likely need to be separated into multiple states to identify different types of degradation that a component may experience, such as a channel becoming too small or too big, possibly from clogs or wear, respectively, each of which would have different effects on the overall behavior of the system. Next, the models are updated to allow for different degrees of failure severity. For example, the qualitative analysis would only have "Degraded" and "Defective" states, however, during the quasi-quantitative analysis a "Degraded" failure's severity can be simulated anywhere on a scale of 0-100% degraded for the component's functionality, as well as its speed if applicable. Also, if a "Defective" failure occurs, a failed position can be identified, such as the airplane's landing gear from the earlier example, failing open or closed. Being able to identify different severity and types of failures is very important in being able to understand how various mission tasks will be affected. For example, if the landing gear becomes "Defective" and fails in the open position after takeoff, the failure would still be manageable as the cruising performance would only be degraded, however, if the landing gear fails in the closed position, this would result in a critical failure as now the landing task would not be achievable without crashing.

4.1.2. General Methodology

Once all Simulink and MATLAB updates are completed, the user can begin simulating specific failure combinations for general system analysis, as well as specific mission analysis. First, the user will be prompted by MATLAB to input the health state of each component; the health state includes whether a failure has occurred and, if so, what type of failure it is, and lastly, how severe it is. With this information, the updated behavioral models will be able to calculate the remaining functionalities of the overall system. Next, the user can input specific mission tasks, such as move from point A to point B to point C. Then, the program identifies the remaining functions that are capable of completing each individual task based on the system's current health. If the remaining functions are not capable of completing the mission tasks, then the program will indicate that the mission is not possible and will specify which parts are responsible for losing that specific capability. If the mission is possible, the program will indicate this along with what, if any, redundancies were needed. Redundancies are based on the optimization portion of the program.

The MATLAB code's optimization is based on trying to balance the failures throughout the system by looking at each component's health and all possible remaining solutions to the individual mission tasks, and then ranking the faulty components from most degraded to least degraded. Then, if a system has three parts for example, the program looks at the top 20 solutions that limit the necessary functionality for the most degraded component, from which the top 10 solutions are then chosen for the second most degraded component, from which the top solution for the least degraded component is finally chosen as the "best" solution. By performing this type of optimization, the goal is to create a balanced rate of degradation by forcing the least degraded components to compensate for the most degraded, but still limiting these compensations as much as possible. This is done to extend a system's lifespan by keeping it from suffering a critical failure in one part, while all other parts are still healthy. For example, a system would be able to get much more use if all parts were 90% degraded before one of them finally failed, as opposed to one part failing when all the other parts are only 20% degraded. This optimization is used to create the "best" course of action to complete specific mission tasks, but the "best" course of action is defined within the MATLAB code based on the necessary importance of certain aspects of the mission. For example, a system could be optimized to complete a mission in the shortest amount of time, or it could be optimized to repeat a mission the most possible times before a critical failure occurs; for the case study described in the following section, the system was optimized for the latter. Lastly, the optimization procedure will not only help balance failure degradation among components but will also help make all mission plans more robust.

Lastly, metrics within the MATLAB code are created to provide a system's Overall Coverage Rating (OCR), Mission Time, and Mission Robustness Ratings (MRR). Overall Coverage Rating is the ratio of a faulty system's remaining possible functionalities versus a nominal system's possible functionalities. This OCR value will identify how much of a system's functionality was eliminated by the system's current failure scenario. The Mission Time value will be the time it takes to complete all of the mission's tasks based on the optimized solutions. Lastly, the Mission Robustness Ratings are essentially the same as the OCR, however, there is an MRR for each individual mission task in order to identify which tasks are most affected by the current failure scenario. The OCR and MRR values are then used to compare and improve mission plans for specific failure scenarios, based on which missions are more robust and therefore, which will be better able to handle further system degradations. In the following sections, this method will be applied to an adaptable robot arm system.

4.2. Phase II Case Study: 3-Linkage Robotic Arm

4.2.1. Case Study: Overview

Despite the fact that the same model is meant to be used and expanded through each of the three FIMA phases, because the system used in the Phase I case study is a real system that is currently still only in the early design stage, a new 3-linkage robotic arm system, with a Base Joint that rotates on the X-Y axis, and three Arm Joints that rotate on the r-Z axis, as seen in Figure 4.1, was chosen for the Phase II case study. This robotic arm system was chosen because it has a well understood behavior with known governing equations, as well as the fact that it is an adaptable complex system with multiple functional redundancies and mission possibilities, which will serve as an excellent example for the FIMA method's mission analysis capabilities. The robotic arm system described here is meant to represent a potential manufacturing robot that might be found on a factory floor assembly line that would have missions of moving objects between different positions.

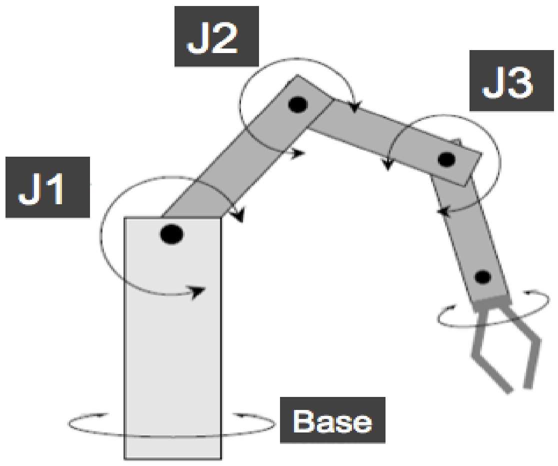


Figure 4.1 – 3-Linkage Robotic Arm Assembly

Unfortunately, when everything is working nominally there is no simple way of determining how good a mission plan truly is, and it is only when specific failures begin to appear that any accurate mission analysis can really take place. For example, one mission plan might only require moving an object a short distance, but one of the locations is at the robot's maximum reach, while another mission plan has the robot moving an object much further distances, but the locations are closer to the base. While the first mission plan may be quicker and require fewer movements, therefore, making it seem like the better mission, just a small degradation is all that would be necessary to make the maximum reach unachievable, making the first mission plan impossible, while the second mission plan would go virtually unaffected. This shows how the differentiation between how various missions will react to a given failure scenario is of the utmost importance because depending on the type of failures that occur, a system may need to re-optimize or possibly even completely change a mission plan. This analysis problem is why the ability to identify the "best" mission based on the Mission Robustness Ratings for each mission task is one of the major unique contributions of the FIMA method. Therefore, this robustness analysis will be addressed in much greater detail in this case study.

4.2.2. Case Study: Methodology

The first step of Phase II was to create an updated, quasi-quantitative Simulink model with state-machines and failure-logics for each component: the Base Joint, Joint1, Joint2, and Joint3. Because all of the components are the same type of mechanism, i.e. joints, the state-machines were all able to be nearly identical, differing only in their governing equations' nominal values; the Base is defined as having a nominal movement range of 0 to 180 degrees, Joint1 can range from 0 to 90 degrees, and Joint2 and Joint3 can each range from -180 to 180 degrees. Each joint was sampled every 1 degree. Sensitivity analysis was done by altering the sampling size to every 3 degrees, as well as every 6 degrees, for each of the use-cases in section 4.2.3. For the 3-degree sampling size the differences in the resulting OCR, MRR, and Mission Time values were minor (average differences of less than 1% for the OCR values, roughly 2% for the MRR values, and roughly 1 minute for the Mission Times), however, for the 6-degree sampling size the difference in results were quite significant and unpredictable. Each linkage was then given a length of 3 feet and the nominal rotational speed of each was defined as 30 degrees per second. Next, for simplicity sake, during this case study it was assumed that there were no obstacles within the arm's movement range, i.e. no external system failures. Also, the linkages were identified as connecting off-axis in order to allow the arm to rotate in on itself. These criteria were all chosen arbitrarily for this example and would likely differ depending on the type of arm assembly and quality of components. Also, these criteria could easily be changed to include obstacles or exclude certain types of arm movements by adding limitations within the MATLAB code. Lastly, all arm coordinates were then calculated using the following forward kinematic equations within the MATLAB code:

$$\mathbf{r = L1*cos(Theta1)+L2*cos(Theta1+Theta2)+L3*cos(Theta1+Theta2+Theta3)} \tag{1}$$

$$\mathbf{Z = L1*sin(Theta1)+L2*sin(Theta1+Theta2)+L3*sin(Theta1+Theta2+Theta3)} \tag{2}$$

$$\mathbf{X = LR*sin(Theta0)} \tag{3}$$

$$\mathbf{Y = LR*cos(Theta0)} \tag{4}$$

where the three arm joints are located in the r-Z coordinate plane, and the base is located in the X-Y coordinate plane. Also, L1, L2, and L3 are the lengths of the three arm linkages, LR is the total length of the arm in the r-direction, and Theta0, Theta1, Theta2, and Theta3 are the joint angles for the Base, Joint1, Joint2, and Joint3, respectively. Examples of the different state-based governing equations within the Simulink state-machines used for updating each joint's movement range, i.e. adjusting the minimum and maximum rotational angles as well as each joint's movement speed for given failure scenarios, can be seen for the Base Joint in Figures A5-A7.

The Simulink model and MATLAB code are related in such a way that the input data for the Simulink model will come from the first portion of the MATLAB code and user inputs, then this information will be processed and outputted from Simulink back into the second portion of the MATLAB code. The Simulink inputs consist of multiple variables for each component that are dependent upon the user's responses to prompts generated by the MATLAB code. The user inputs the health of each component, as well as the degree of failure and type of failure that they wish to have simulated; the types of failure for this system are movement and speed-based. A joint's movement range can be "Defective," resulting in the joint being stuck at a user-specified angle, or it can be "Degraded," anywhere from 0-100% that can then be applied to either a Lower, Middle, or Upper limitation. For example, a 10% Lower limitation for a range of 0-180 degrees would result in a new range of 18-180 degrees, a 10% Middle limitation would result in a new range of 9-171 degrees, and a 10% Upper limitation would result in a new range of 0-162 degrees. Likewise, a joint's speed can also be "Degraded" anywhere from 0-100%. Also, along with the user-inputted, failure-based speed degradation, a joint's speed is also programmed to decrease linearly over time depending on the component's lifespan rating, i.e. if a joint has a lifespan of 10,000 180 degree movements with a speed of 30 degrees per second, then if that joint moves 180 degrees 5,000 times it will now only be capable of moving at 15 degrees per second.

The Simulink model first processes the current state of each component based on the user's inputs and then provides output data, such as updated performance values and new minimum and maximum achievable angles and speeds, that will then be processed by the MATLAB code to determine the Overall Coverage Rating, as well as the graphical representation of all functionalities for the overall system, which can be seen in Fig. 4.2; the top two plots represent the overall coverage of the arm for a nominal system on the X-Y and r-Z axes, respectively, and the bottom two plots represent the remaining coverage for a random faulty system. The example faults present in the bottom plots were: a 20% Middle limitation for the

Base, a 25% Middle limitation for Joint1, a 40% Upper limitation for Joint2, and a 35% Lower limitation for Joint3.

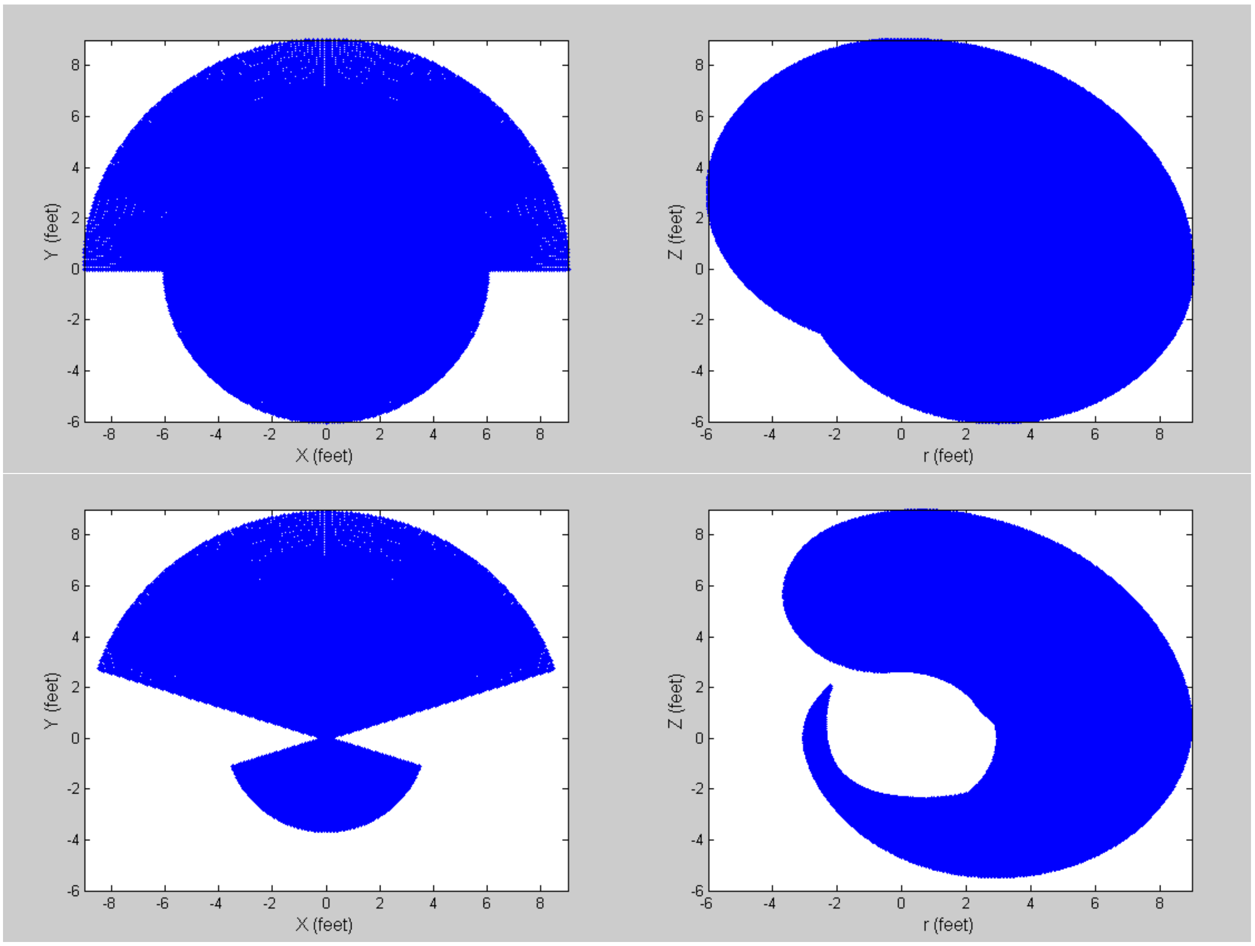


Figure 4.2 – Possible Movement Coverage for 3-Linkage Robot Arm (Top: Nominal, Bottom: Degraded) (Left: X-Y axis, Right: r-Z axis)

Next, the user will be asked to input various mission details, such as the various tasks, i.e. moving an object from point A to point B in the [X,Y,Z] coordinate plane, as well as how many cycles of these tasks need to be completed. Each [X,Y,Z] location was given a margin of error of 0.2 feet based on the assumption that the arm's claw would be at least slightly bigger than the object it is picking up. These user inputs will then result in mission-specific output data that will be compared with the overall system output data to determine mission feasibility, to optimize the mission plan, and to identify any redundancies or repairs that may be needed. An example of the plot generated comparing the original, nominal arm angles to the degraded but optimized arm angles for a given mission can be seen in Fig. 4.3; the mission tasks were to move between two arbitrarily chosen points, [3,4,4] to [2,2,5], and the degraded plot was for the same example failure scenario as seen in Fig. 4.2, where Joint2 is the most degraded component and therefore, the movements were optimized for Joint2.

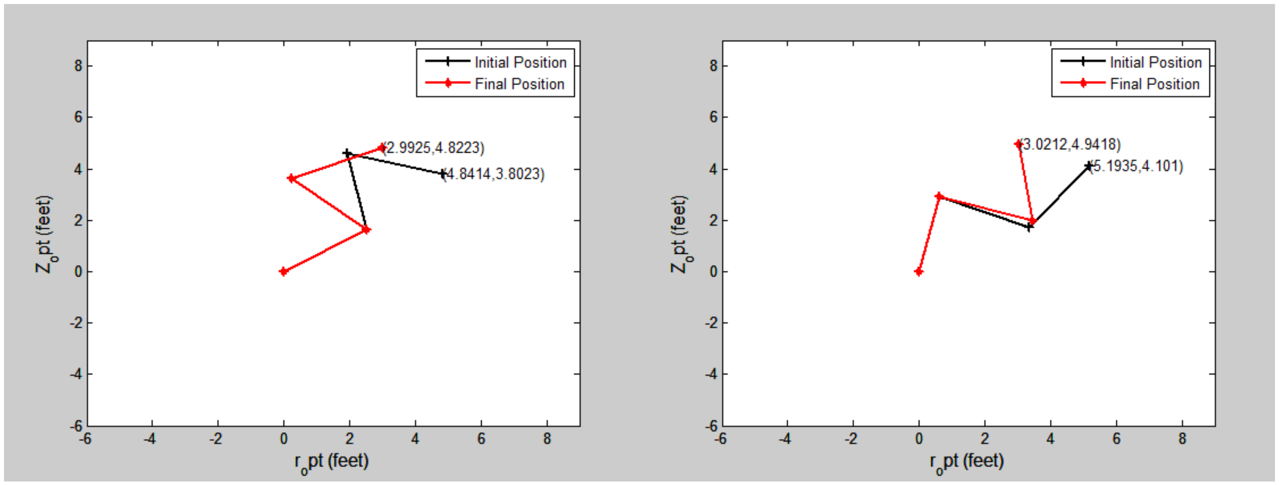


Figure 4.3 – Nominal (Left) vs. Optimized for Degradation (Right) arm positions on the r-Z axis

For this system, two use-cases were explored in the following section. The first is using the FIMA method for comparing two different missions during the same failure scenarios, and the second is utilizing the failure data to optimize a set mission plan to handle further failures by altering the position of the entire robot.

4.2.3. Case Study: Results and Discussion

4.2.3.1. Use-Case 1: Mission Comparisons

The first use-case of the FIMA method's quasi-quantitative analysis was to evaluate different mission plans, i.e. different sets of tasks, or initial and final positions, for different failure scenarios in order to show that by using the Overall Coverage Rating (OCR) and the Mission Robustness Ratings (MRR) the FIMA method can accurately identify which mission plan is best. The mission data for this use-case can be seen in Table 4.1. This mission data includes three different failure scenarios, where three failure factors for each component are identified: Percent Degraded-Range, Limitation Type, and Percent Degraded-Speed, respectively. Each scenario is then evaluated for two different mission plans: A and B. Each mission plan is responsible for two tasks: moving the robotic arm from an Initial position to a Final position, and these missions are to be repeated 250 times. The outputs for each mission are the Mission Feasibility (including which component the mission's optimization was based), the total Mission Time, and the Mission Robustness Ratings for both mission tasks, i.e. the initial and final points.

Table 4.1 – Mission Data for Use-Case 1

	Failure Scenario #1		**Failure Scenario #2**		**Failure Scenario #3**	
Base	**0%, None, 0%**		**0%, None, 0%**		**0%, None, 0%**	
Joint1	**5%, Upper, 1%**		**10%, Upper, 1%**		**45%, Upper, 1%**	
Joint2	**12%, Middle, 1%**		**24%, Middle, 1%**		**48%, Middle, 1%**	
Joint3	**9%, Lower, 1%**		**18%, Lower, 1%**		**47%, Lower, 1%**	
OCR	**75.4%**		**56.1%**		**15.1%**	
	A	**B**	**A**	**B**	**A**	**B**
Initial	**[3,4,5]**	**[2,3,4]**	**[3,4,5]**	**[2,3,4]**	**[3,4,5]**	**[2,3,4]**
Final	**[-2,3,6]**	**[-3,4,6]**	**[-2,3,6]**	**[-3,4,6]**	**[-2,3,6]**	**[-3,4,6]**
Cycles	**250**	**250**	**250**	**250**	**250**	**250**
Feasibility	**Y, FO-J2**	**Y, FO-J2**	**Y, FO-J2**	**Y, FO-J2**	**Y, FO-J2**	**Y, FO-J2**
Time	**24.01 min**	**46.68 min**	**23.73 min**	**50.63 min**	**24.57 min**	**37.86 min**
MRRi	**94.3%**	**95.2%**	**89.1%**	**55.8%**	**27.6%**	**27.3%**
MRRf	**95.2%**	**91.4%**	**90.1%**	**86.0%**	**32.0%**	**30.4%**

For Failure Scenario #1, the Overall Coverage Rating for the arm is 75.4%, which indicates that roughly a quarter of the system's total functionality has been lost. Next, looking at the two mission plans, both are feasible and both were functionally optimized for Joint 2, which is what was expected due to the fact that Joint 2 was the most degraded component. Finally, the mission time, MRRi, and MRRf values are evaluated. MRRi and MRRf are the Mission

Robustness Ratings for each of the mission tasks, i.e. the initial and final positions. For the time comparison, the shorter the Mission Time the better. However, the shortest mission is not always the most robust and this is where the Mission Robustness Ratings' importance is seen. As mentioned earlier, the individual Mission Robustness Ratings are indicators of how the system handles specific failure scenarios for its various mission tasks, and it is desired that both MRRi and MRRf values are larger than the OCR due to the fact that the OCR indicates the overall, average robustness, and therefore, larger MRR values would signify that the mission plans have above average robustness. As seen in Table 4.1 both missions have relatively high MRRi and MRRf values, implying that neither mission was very affected by Failure Scenario #1, and they are also above the OCR value, which as previously mentioned, is desired. However, when directly comparing mission A to mission B, mission A is better all-around, as it not only can complete the necessary 250 cycles faster, but the mission tasks are more robust on average than those for mission B. Even after only the first failure scenario, mission A can be identified as the preferred mission plan, however to show that this assumption holds true for further degradations, Failure Scenario #2 and #3 were simulated. As expected, mission A remains faster and more robust than mission B for all scenarios. In Failure Scenario #2, mission A becomes significantly better in all categories than mission B. However, in Failure Scenario #3, while mission A is still better, the different components' degradations are becoming balanced through optimization, and, as expected, the optimization has also begun to balance each mission's robustness ratings, as well as helping to decrease each of their mission times, reducing them both below even their far less degraded Failure Scenario #1's times.

4.2.3.2. Use-Case 2: Mission Adjustments

The second use-case for the FIMA method's quasi-quantitative analysis was to demonstrate that by using the OCR and MRR values for a specific failure scenario, a mission plan could be greatly improved; both the failure scenario and mission plan were arbitrarily chosen for this study. Unfortunately, because certain mission plans might not be able to be altered, such as a robot picking up a bolt and then placing it on a specific area of a vehicle coming down the assembly line, the position of the entire robot itself might need to be altered in order to increase the system's robustness. Therefore, it is assumed that the arm assembly is capable of being moved on the X-Y plane, such as by being placed on wheels, in order to optimize its position relative to the initial and final positions it must reach. As seen in Table 4.2, the original mission plan is again responsible for two tasks of moving the robotic arm from the initial position to the final position, 250 times, and the output variables for each mission are the same as for use-case 1: Mission Feasibility (including which component the mission's optimization was based), total Mission Time, and Mission Robustness Ratings for both mission tasks.

Table 4.2 – Mission Data for Use-Case 2

Base	**0%**			
Joint1	**15%, Lower, 1%**			
Joint2	**15%, Lower, 1%**			
Joint3	**20%, Middle, 1%**			
OCR	**57.6%**			
	Original	**(Shift: -2Y)**	**(Shift: +3X)**	**(Shift: -1Y)**
Initial	**[-1,1,1]**	**[-1,3,1]**	**[-4,3,1]**	**[-4,4,1]**
Final	**[4,3,-1]**	**[4,5,-1]**	**[1,5,-1]**	**[1,6,-1]**
Cycles	**250**	**250**	**250**	**250**
Feasibility	**Y, FO-J3**	**Y, FO-J3**	**Y, FO-J3**	**Y, FO-J3**
Time	**71.57 min**	**44.46 min**	**34.99 min**	**31.03 min**
MRRi	**17.9%**	**8.3%**	**60.5%**	**80.8%**
MRRf	**50.3%**	**81.8%**	**53.1%**	**82.7%**

As seen in Table 4.2, when the failure scenario listed occurs, the original mission plan is identified as incredibly poor. It is still feasible, however, both MRR values are well below the OCR, indicating that there are far better mission plans available, and this is where the designer would ideally be able to tweak the position of the robot in order to find a more robust mission plan. First, a shift in the negative Y direction was applied, i.e. backing the robot away from the assembly line, and while this adjustment improved the mission time and the MRR of the final position, it reduced the MRR of the initial position. Next, a shift in the positive X direction was applied, and this effectively improved the mission time and both MRR values, however, the MRR value of the final position is still below the OCR, so further improvements can still be made. Finally, another shift in the negative Y direction was made and this resulted in vast improvements to both MRR values and the overall mission time. While further improvements may have been possible through further adjustments, for the purposes of this study, these improvements were sufficient. Ultimately, this study showed that by following the FIMA method, using the OCR and MRR values, a designer could effectively reduce the original mission time by more than half, while also vastly improving the system's mission robustness.

CHAPTER 5

CONCLUSIONS

The Failure Identification for Mission Analysis (FIMA) method proposed in this paper is designed to allow a single, adaptable model to be used throughout the entire design process of a complex system. The method was shown to be able to provide models for qualitative failure analysis during early design stages, and then expand these models for quasi-quantitative analysis, as more information about the system becomes available during the later design stages. By using the FIMA method, designers should no longer be required to create new models or switch analysis techniques throughout the different design stages, making the whole process much more efficient and streamlined than with existing failure analysis methods. Moreover, the FIMA method uniquely allows for the simulations of manufacturing-based failures, as well as traditional function-based failures. However, the biggest and most unique contribution made by the FIMA method is its ability to take a complex system's failure information and use it for mission assessment and optimization.

With the constant advancement of technology and the ever-growing capabilities of complex systems, it is absolutely vital to know what the system is being used for in order to accurately understand the effects of failures on the overall system performance, and the lack of this mission analysis is where current methods fall short. By using the FIMA method, on the other hand, mission assessments and optimizations can be performed in order to balance failure degradations and increase mission robustness for any number of mission plans in an effort to maximize a system's use in between repairs. This unique ability could be especially beneficial for complex systems that are incapable of receiving repairs, such as the NASA rovers exploring

Mars, because even if certain functions are lost due to failures, it is vital to know which functions and mission tasks are still feasible in order to maximize the amount of use the existing rovers can perform before new ones need to be sent.

By utilizing failure information for mission analysis, the FIMA method can provide more comprehensive and useful information than other current failure analysis methods. With next-generation technologies becoming increasingly more complex, it is not enough anymore simply to know how a system will fail. What the system will be doing, what environment it will be doing it in, and what functional adjustments are available must all be accurately identified in order to effectively analyze the effects of complex failures in a complex system, and the FIMA method has been designed to do just that. First, the FIMA method identifies and assesses the potential functions and mission tasks that a complex system may be asked to perform, and then based on various potential failure scenarios, the functions and tasks that are the most and least robust can be identified. Then, by using this information, the FIMA method is able to optimize the system's performance in order to more effectively achieve specific mission plans for any given failure scenario.

CHAPTER 6

FUTURE RECOMMENDATIONS

The benefit of the FIMA method does not end with its ability to provide failure simulations. While initially, during Phase II, the health state of each component and the specific degree of failure must be inputted by the user, in the future, with the addition of actual sensor data, the same MATLAB code that is used for the quasi-quantitative simulations could be used as a diagnostics tool for real-time optimization of real-world physical systems. In this capacity, the code would again not care about the causes of failure, but instead only about the system's functional capabilities that remain. For example, in the manufacturing robot used for the Phase II case studies, instead of the user inputting a "Percent Degraded" value prior to a mission, an actual robot would run a quick system diagnostics check by rotating each individual joint to their minimum and maximum angles at peak speed. Then, instead of the state-machines having to calculate the individual minimum and maximum values and speeds, the sensors would send their data directly back to the code that would then proceed as before to optimize the arm angles based on the different minimums and maximums. Therefore, by using the FIMA method, a designer should be able to use the same model, built congruently with the physical design, from the early, conceptual design phases, all the way to the final detailed phases, and ultimately, into real-world application.

Furthermore, future work on Phase III of the FIMA method will focus on a fully quantitative analysis approach by adding more detailed failure modes to the Phase II models. The quasi-quantitative analysis will always be somewhat abstract, as the specific causes of degradation for certain failure modes are not specified. During a fully quantitative analysis however, each failure mode can be expanded. For example, if a wheel was "Degraded" in the qualitative and quasi-quantitative phases, in Phase III's fully quantitative models, "Degraded" could be expanded into such failure modes as "Traction Loss Caused by Wear" or "Low Tire Pressure," and then "Defective" could be either "Flat" or "Jammed," and each one of these would result in their own behavioral equations as well. Also, the failures could be separated to indicate different internal and external causes, i.e. "Jammed" could be caused by an internal malfunction that is a critical failure and cannot be fixed without a total part replacement, or the jam could be caused by an external failure, such as the wheel being stuck in mud, which would not be a critical failure in the sense that a component needs replacement, it would only be a failure on the system-level objective of movement. Phase III of the FIMA method would then also be able to determine such differences between failures and be able to inform the user the best course of action moving forward; if the first definition of "Jammed" is simulated, the system would produce an error message indicating that the mission cannot be completed and that the broken component must be replaced. If the second definition of "Jammed" is simulated, the system would produce an error message that identifies this failure as an external failure only, and if corrected, perhaps through redundancies from other non-compromised components, such as by switching into 4-wheel drive, would have no long-lasting effects on the system. This ability would allow the model to identify even more potential failure scenarios, as well as effectively label which are critical vs. manageable.

The FIMA method's Phase III quantitative analysis also will be to explore path-planning optimization. During the Phase II robotic arm case study, it was assumed that there were no external obstacles and therefore, the arm was able to move between points in a straight line. However, in more complex cases, it will be necessary not only to know how failures affect the arm's possible angle combinations at mission points, but also how failures affect the arms ability to move around obstacle to get from one point to the other. For example, some internal failures or external obstacles may affect the arm's ability to move left and right, while others may affect the ability to move forwards and backwards, and so depending on the required mission plan, the arm's path between points will need to be optimized, along with the joint's angle combination optimization done in Phase II.

Next, future work on the Phase III quantitative analysis will also include validation of the models through experimentation on a physical testbed. For the case study examined in Phase II, the 3-linkage robot arm, this validation could be done a number of ways. Mission abilities and times could be tested and compared with the failure scenarios and mission plans simulated through control input constraints for each joint's speed and minimum and maximum angles, or by physically replacing the testbed's healthy joints with different types of degraded joints. Degraded joints could be manufactured to have various degrees of wear, jams, or breaks and then based on each of these effects on rotational speeds and minimum and maximum angles, mission plans, arm positions and paths, and the effects of further degradation on the overall system performance could be tested.

Lastly, future work on the FIMA method should include its application to more complex systems with more complex missions in order to show its scalability and its true merit for diverse applications. One such application idea would be to use the FIMA method to create an advanced GPS system. Currently, GPS systems are essentially external system failure analysis tools with mission optimization capabilities. GPS, generally speaking, identifies a mission plan, or route, based on shortest mission time, and then based on external system failures, such as things like traffic jams, missed turns, and construction detours, the system identifies all remaining possible solutions and then re-optimizes based on mission time and provides a new mission plan. However, GPS systems do not consider the condition or capabilities of the vehicle being driven. For example, if a vehicle is about to run out of gas the GPS will simply find the closest gas station, regardless of the path needed to get there, i.e. if there is a gas station half a mile away, but all up hill with multiple stops, and there is another gas station one mile away, but all down hill with no stops, the GPS will still identify the first gas station as the top choice. However, by using the FIMA method, the system could identify the internal system failure of “Low Gas” and understand that the second gas station is better due to there being fewer and less exhaustive functionalities needed to get there. This would be especially useful in the future when there are self-driving cars, as there will not be a human driver that understands coasting down hill requires less gas. Similarly, if there was a quicker path to get somewhere, but some of the roads were dirt roads, a truck might have no problem, but a Ferrari would likely rather take a longer, smoother path, and therefore, knowing the type of vehicle can also influence a mission’s optimization. This is just one of many future possible applications for using the FIMA method to help optimize missions after failures have occurred.

REFERENCES

[1] **Y. Umeda, T. Tomiyama, H. Yoshikawa.** A Design Methodology for a Self-Maintenance Machine. *University of Tokyo,* Japan. 1995.

[2] **Y. Umeda, T. Tomiyama. and H. Yoshikawa**. A design methodology for a self-maintenance machine based on functional redundancy. In *ASME Design Theory and Methodology Conference*. 1992.

[3] **Z. Huang, Y. Jin.** Conceptual Stress and Conceptual Strength for Functional Design-for-Reliability. In *Proceedings of the ASME Design Engineering Technical Conferences; International Design Theory and Methodology Conference,* 2007.

[4] **W. G. Ireson** Reliability Handbook. McGraw-Hill, New York. 1966.

[5] **J. Agte, N. Borer, O. de Weck.** Design of Long-Endurance Systems with Inherent Robustness to Partial Failures During Operations. *Journal of Mechanical Design* Vol. 134, 2012

[6] **E. Kujawski.** Optimization of Critical Systems for Robustness in a Multistate World. *American Journal of Operations Research* Vol. 3. Pg. 127-137, 2013.

[7] **D. Krus and K. Grantham-Lough.** Applying function-based failure propagation in conceptual design. In *Proceedings of the ASME Design Engineering Technical Conferences; International Design Theory and Methodology Conference,*2009

[8] **D. Jensen, I. Tumer, T. Kurtoglu, C. Hoyle. 2012.** Application and Analysis of Complex Systems Using the Function Failure Identification and Propagation Framework.

[9] **J. de Kleer and B. C. Williams.** Diagnosing multiple faults. *Artificial Intelligence.* Vol. 32(1) Pg 97-130. 1987.

[10] **Y. Umeda. T. Tomiyama, and H. Yoshikawa**. Model based diagnosis using qualitative reasoning. *Computer Applications in Production and Engineering*. Pg. 443-450. 1989.

[11] **Tumer, I.Y. and R.B. Stone.** Analytical method for mapping function to failure during high-risk component development**.** In *Proceedings of the Design Engineering Technical Conference*. 2001

[12] **R. Roberts, R. Stone, I. Tumer.** Deriving Function-Failure Similarity Information for Failure- Free Rotorcraft Component Design. In *Proceedings of ASME Design Engineering Technical Conference and Computers and Information in Engineering Conference*. 2002

[13] **R. Stone, I. Tumer, M. Van Wie.** The Function-Failure Design Method**.** In *ASME Journal of Mechanical Design*. Vol. 127. Pg. 397-407, 2005.

[14] R. Stone, I. Tumer, S. Arunajadi. A Framework for Creating a Function-Based Design Tool for Failure Mode Identification. In *Design Engineering Technical Conferences, Design Theory and Methodology Conference*. 2002.

[15] G. Pahl, W. Beitz, J. Feldhusen, K. Grote. Engineering Design: A Systematic Approach. *Springer-Verlag, Berlin.* 2007

[16] P. Frank, E. Alcorta Garcia, B. Koppen-Seliger. Modelling for fault detection and isolation versus modeling for control. *Mathematics and Computers in Simulation* Vol. 53 Pg. 259-271. 2000.

[17] B. Kuipers. Reasoning with Qualitative Models. *Artificial Intelligence* Vol. 59. Pg 125-132.

[18] B. Kuipers. Qualitative simulation**.** *Artificial intelligence* Vol 29(3) Pg 289-338, 1986.

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

[20] T. Kurtoglu, I. Tumer, D. Jensen. A Functional Failure Reasoning Methodology for Evaluation of Conceptual System Architectures. *Research in Engineering Design,* Vol. 21(4). Pg 209-234, 2010.

[21] R. Stone, I. Tumer, M. Stock. Linking product functionality to historical failures to improve failure analysis in design. *Research in Engineering Design,* Vol. 16(2) Pg. 96-108, 2006

[22] K. Grantham-Lough, R. Stone, I. Tumer. Implementation Procedures for the Risk in Early Design (RED) Method. *Journal of Industrial and Systems Engineering,* Vol. 2(2) Pg 126-143, 2008.

[23] K. Wang, Y. Jin. An analytical approach to function design. In *14th International Conference on Design Theory and Methodology*, Pg 449-459, 2002.

[24] K. Grantham-Lough, R. Stone, I. Tumer. The risk in early design method. *Journal of Engineering Design*, Vol. 20(2) Pg 144-173, 2009.

[25] I. Tumer, R. Stone. Mapping function to failure mode during component development.

[26] S. Arunajadai. A function based design tool for failure mode identification and failure-free design. 2002

[27] P. Eremenko. Formal Model-Based Design & Manufacture. *DARPA*. 2013.

[28] S. Uder. Function-based failure mode identification and detection in conceptual design: Extending the function-failure design methodology to electrical systems. *University of Missouri-Rolla*. 2004.

[29] R. Stone, K. Wood. Development of a functional basis for design. *Journal of Mechanical Design.*

[30] R. Nagel, R. Stone, R. Hutcherson, D. McAdams, J. Donndelinger. Function design framework (FDF): Integrated process and function modeling for complex systems. In *Proceedings of the ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference,* 2008.

[31] M. Derelov. Qualitative Modeling of Potential Failures. *Journal of Engineering Design* Vol. 19(3), Pg. 201-225, 2008.

[32] E. Coatanea, S. Nonsiri, T. Ritola, I. Tumer, D. Jensen. A Framework for Building Dimensionless Behavioral Models to Aid in Function-Based Failure Propagation Analysis. *Journal of Mechanical Design* Vol. 133. Dec. 2011.

[33] K. Forbus. Qualitative Physics: Past, Present, and Future. *Qualitative Reasoning Group-University of Illinois at Urbana, Champaign.*

[34] N. Tague. Failure Mode Effects Analysis (FMEA)**.** Excerpted from *The Quality Toolbox.* Second Edition, ASQ Quality Press. Pgs. 236–240. 2004. **http://asq.org/learn-about-quality/process-analysistools/overview/fmea.html**

[35] K. Stillings. Advanced Failure Mode Effects Analysis**.** 2010. **http://www.asq501.org/images/Failure%20Mode%20Effects%20Analysis_Advanced.pdf**

[36] A. Nannikar, D. Raut, R. Chanmanwar, S. Kamble, D. Patil. FMEA for Manufacturing and Assembly Process**.** In *International Conference on Technology and Business Management.* 2012

[37] Performing a Failure Mode and Effects Analysis. *Flight Assurance Procedure no. P-302-720.* www.everyspec.com.

[38] Failure Modes, Effects and Criticality Analysis (FMECA) for Command, Control, Communications, Computer, Intelligence, Surveillance, and Reconnaissance (C4ISR) Facilities. *Department of the Army.* Technical Manual: TM 5-698-4. 2006

[39] Failure Mode, Effects and Criticality Analysis (FMECA)**.** *Reliability Analysis Center*. 1993.

[40] J. Andrews, J. Dugan. Dependency Modelling Using Fault Tree Analysis. In *Proceedings of the 17th International System Safety Conference.* 1999.

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

[42] Object Modeling Group SysML *version 1.2***.**

[43] S. Tung, Z. Wang, D. Wang, N. Jiao, Z. Dong. Nanochannel system fabricated by MEMS microfabrication and atomic force microscopy. *IET Nanobiotechnology***.** 2011.

[44] S. Tung. J. Kim. 2013. Method of Fabricating a Nanochannel System for DNA Sequencing and Nanoparticle Characterization. *U.S. Patent #20130213815*.

LIST OF PUBLICATIONS

1. **C. DeStefano, D. Jensen**. A Qualitative Failure Analysis using Function-based Performance State-Machines for Fault Identification and Propagation during Early Design Phases. In *Proceedings of the ASME Design Engineering Technical Conferences; International Computers & Information in Engineering Conference,* 2014.

APPENDIX

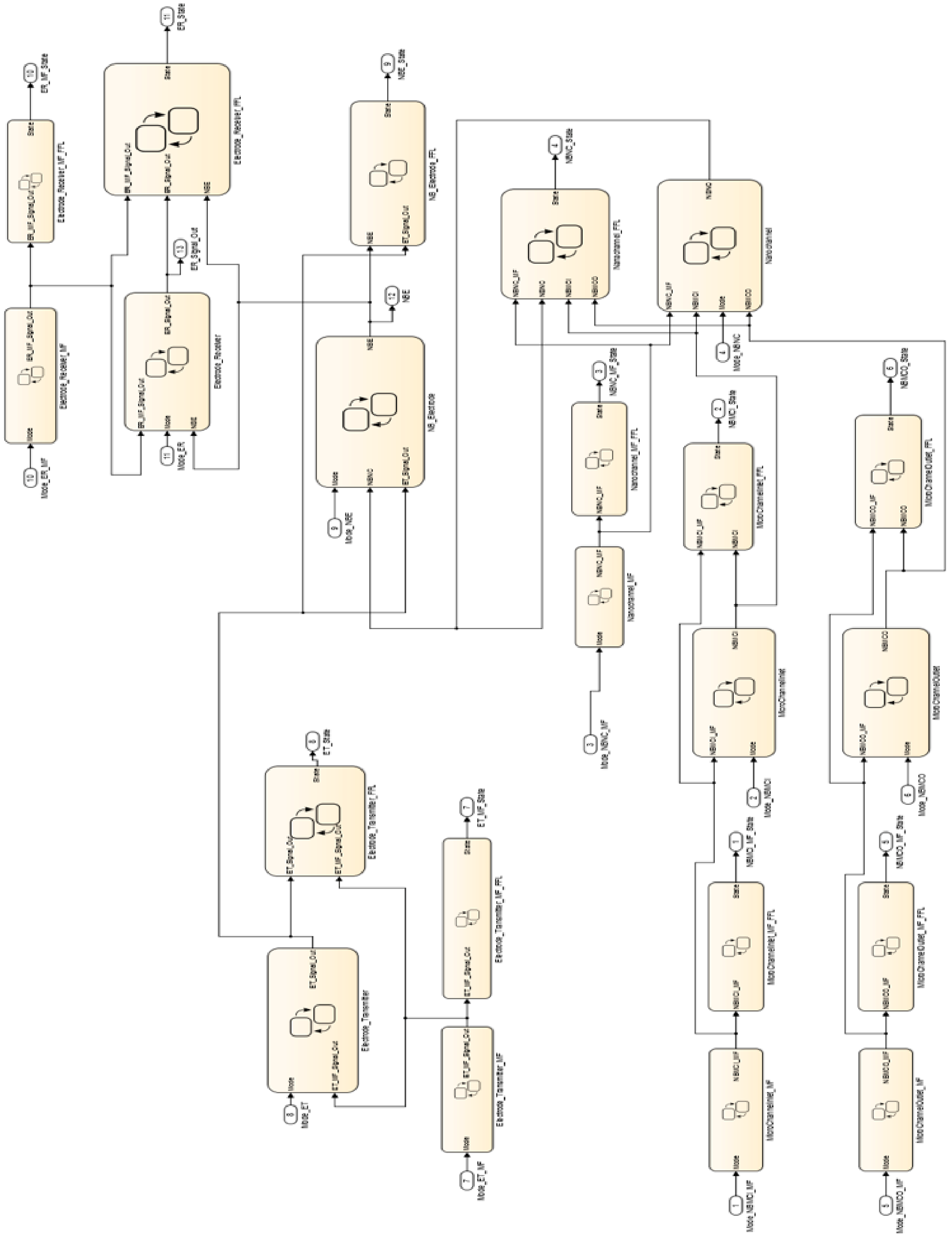


Figure A1. Overview of Simulink Model used for MEMS DNA Sequencer

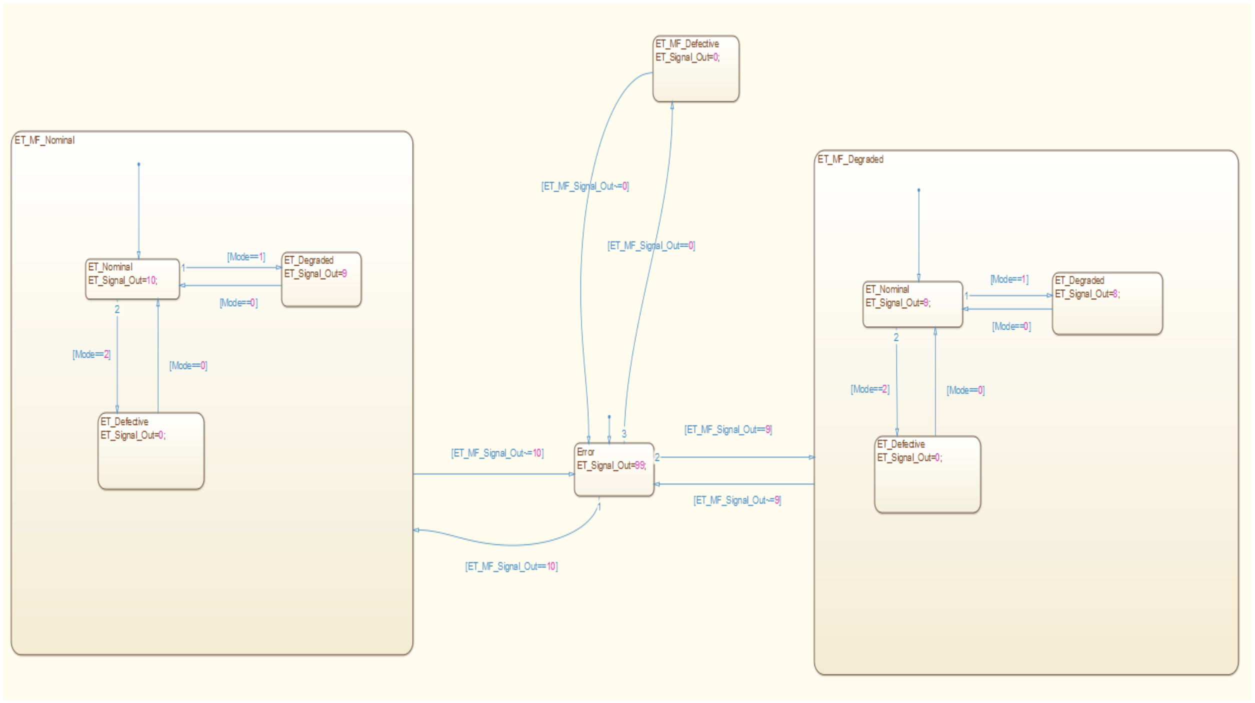


Figure A2. Example of a simple Performance State-Machine (ET Component)

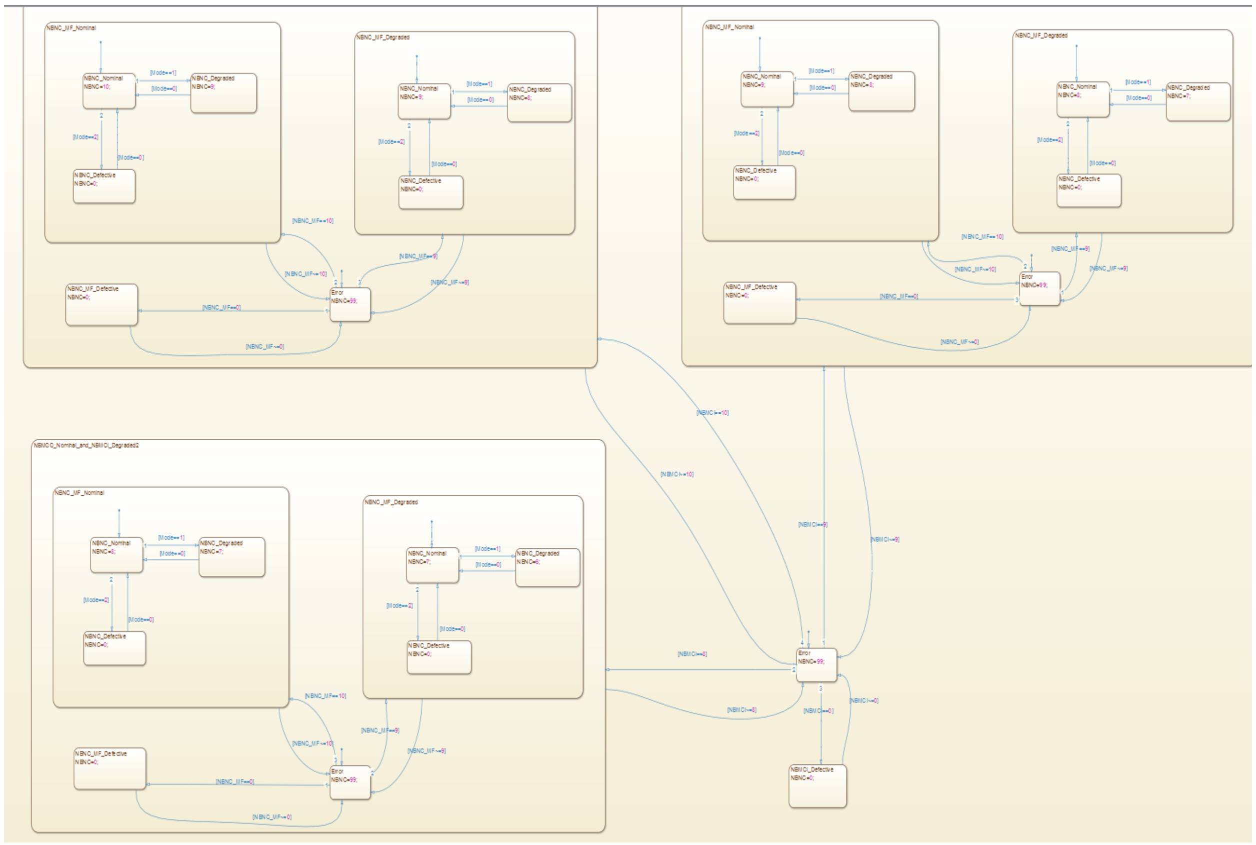


Figure A3. Example of part of a more complicated State-Machine (NBNC Component)

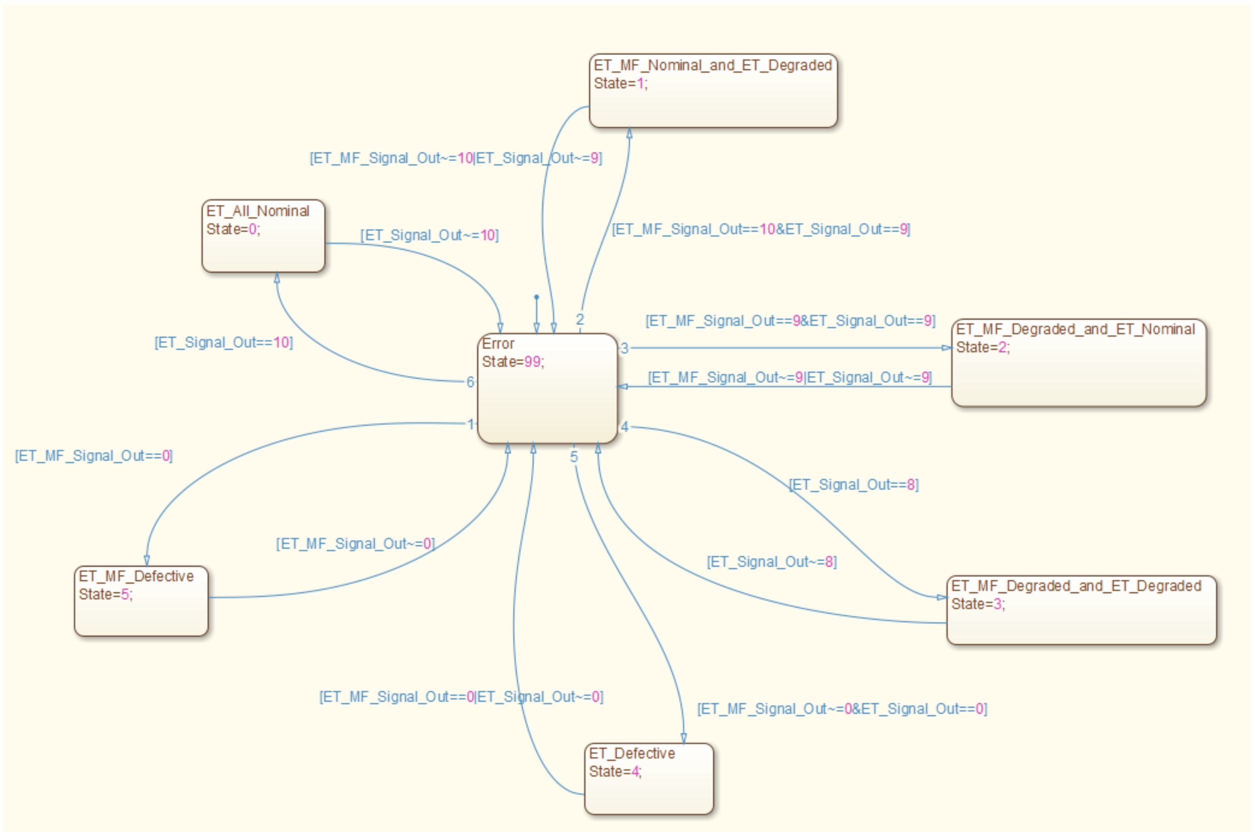

ET_MF_Nominal_and_ET_Degraded
State=1;
[ET_MF_Signal_Out~=10|ET_Signal_Out~=9]
ET_All_Nominal
State=0;
[ET_Signal_Out~=10]
[ET_MF_Signal_Out==10&ET_Signal_Out==9]
[ET_Signal_Out==10]
Error
State=99;
[ET_MF_Signal_Out==9&ET_Signal_Out==9]
ET_MF_Degraded_and_ET_Nominal
State=2;
[ET_MF_Signal_Out~=9|ET_Signal_Out~=9]
[ET_MF_Signal_Out==0]
[ET_Signal_Out==8]
[ET_MF_Signal_Out~=0]
[ET_Signal_Out~=8]
ET_MF_Defective
State=5;
ET_MF_Degraded_and_ET_Degraded
State=3;
[ET_MF_Signal_Out==0|ET_Signal_Out~=0]
[ET_MF_Signal_Out~=0&ET_Signal_Out==0]
ET_Defective
State=4;
1
2
3
4
5
6

Figure A4. Example Failure-Logic Diagram (ET Component)

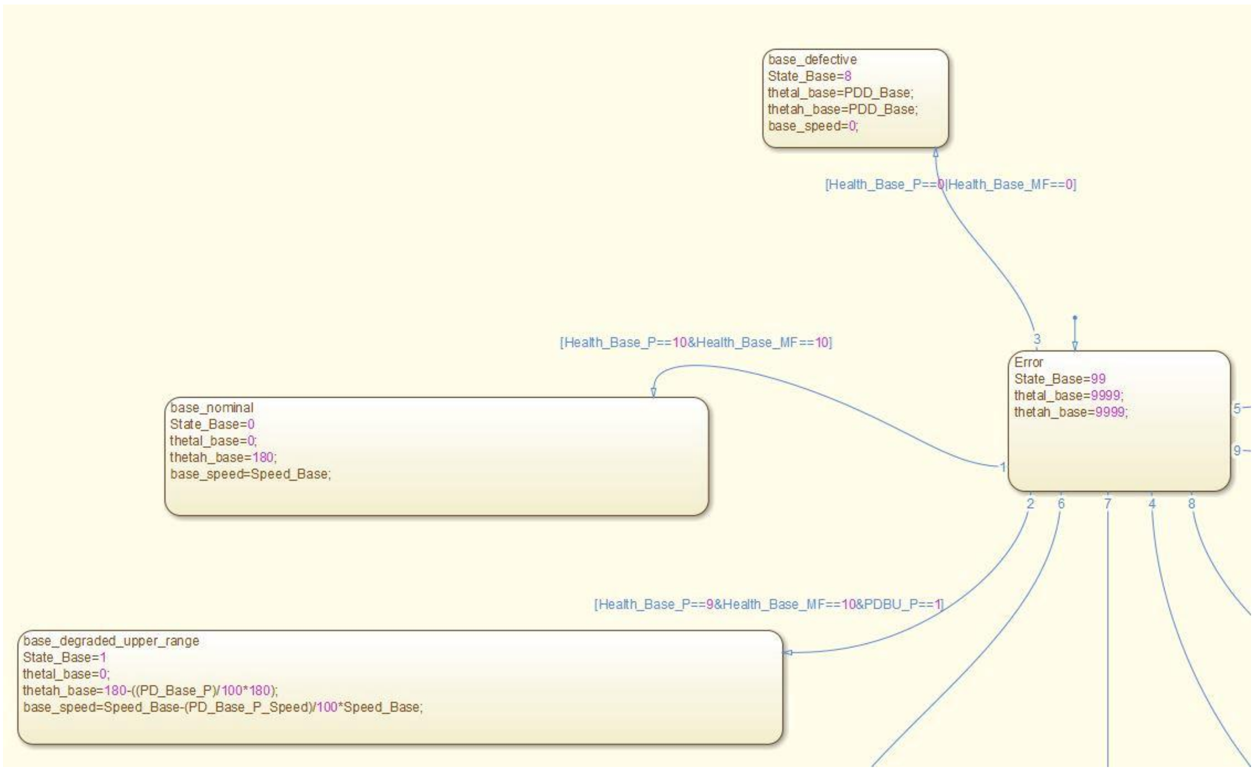


Figure A5. Example 1 of State-based equations (Base Joint)

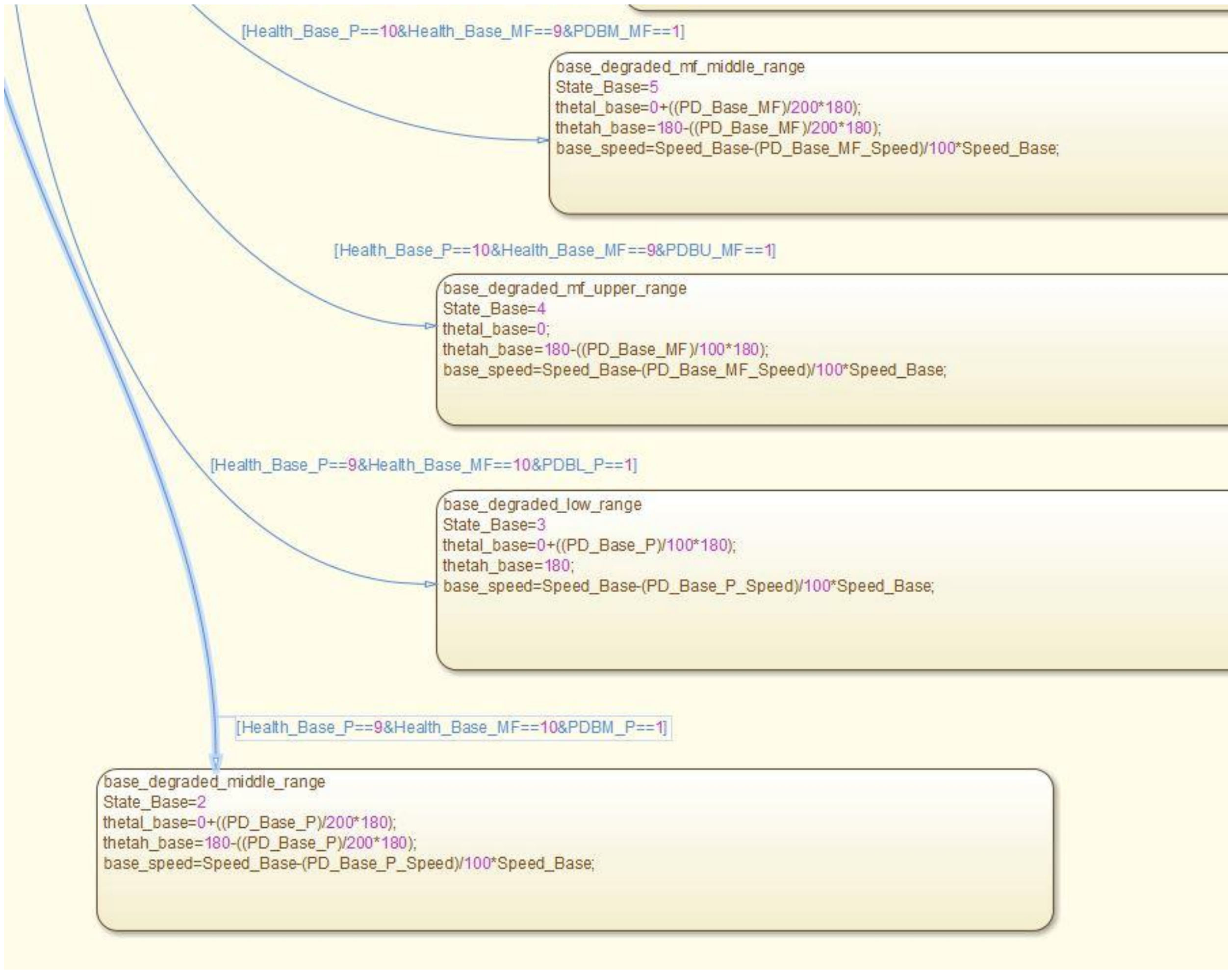

```
[Health_Base_P==10&Health_Base_MF==9&PDBM_MF==1]

base_degraded_mf_middle_range
State_Base=5
thetal_base=0+((PD_Base_MF)/200*180);
thetah_base=180-((PD_Base_MF)/200*180);
base_speed=Speed_Base-(PD_Base_MF_Speed)/100*Speed_Base;

[Health_Base_P==10&Health_Base_MF==9&PDBU_MF==1]

base_degraded_mf_upper_range
State_Base=4
thetal_base=0;
thetah_base=180-((PD_Base_MF)/100*180);
base_speed=Speed_Base-(PD_Base_MF_Speed)/100*Speed_Base;

[Health_Base_P==9&Health_Base_MF==10&PDBL_P==1]

base_degraded_low_range
State_Base=3
thetal_base=0+((PD_Base_P)/100*180);
thetah_base=180;
base_speed=Speed_Base-(PD_Base_P_Speed)/100*Speed_Base;

[Health_Base_P==9&Health_Base_MF==10&PDBM_P==1]

base_degraded_middle_range
State_Base=2
thetal_base=0+((PD_Base_P)/200*180);
thetah_base=180-((PD_Base_P)/200*180);
base_speed=Speed_Base-(PD_Base_P_Speed)/100*Speed_Base;
```

Figure A6. Example 2 of State-based equations (Base Joint)

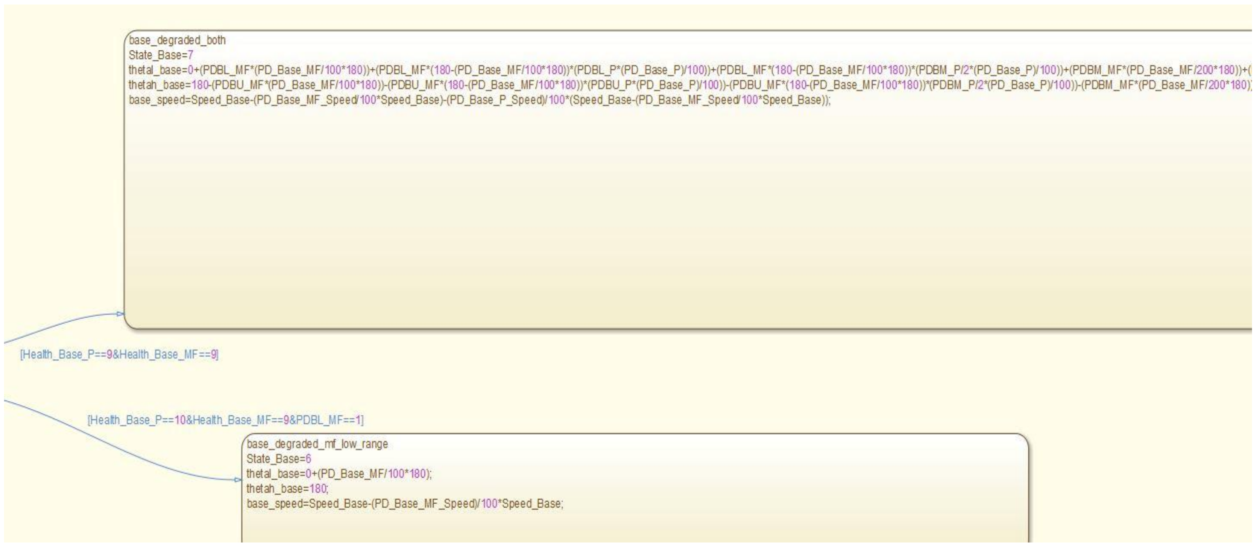

```
base_degraded_both
State_Base=7
thetal_base=0+(PDBL_MF*(PD_Base_MF/100*180))+(PDBL_MF*(180-(PD_Base_MF/100*180))*(PDBL_P*(PD_Base_P)/100))+(PDBL_MF*(180-(PD_Base_MF/100*180))*(PDBM_P/2*(PD_Base_P)/100))+(PDBM_MF*(PD_Base_MF/200*180))+(
thetah_base=180-(PDBU_MF*(PD_Base_MF/100*180))-(PDBU_MF*(180-(PD_Base_MF/100*180))*(PDBU_P*(PD_Base_P)/100))-(PDBU_MF*(180-(PD_Base_MF/100*180))*(PDBM_P/2*(PD_Base_P)/100))-(PDBM_MF*(PD_Base_MF/200*180);
base_speed=Speed_Base-(PD_Base_MF_Speed/100*Speed_Base)-(PD_Base_P_Speed)/100*(Speed_Base-(PD_Base_MF_Speed/100*Speed_Base));

[Health_Base_P==9&Health_Base_MF==9]

[Health_Base_P==10&Health_Base_MF==9&PDBL_MF==1]

base_degraded_mf_low_range
State_Base=6
thetal_base=0+(PD_Base_MF/100*180);
thetah_base=180;
base_speed=Speed_Base-(PD_Base_MF_Speed)/100*Speed_Base;
```

Figure A7. Example 3 of State-based equations (Base Joint)