UTILIZING FAILURE INFORMATION FOR MISSION ANALYSIS FOR COMPLEX SYSTEMS

DeStefano, Charlie; Jensen, David
University of Arkansas-Fayetteville, United States of America

Abstract

This paper presents a new failure analysis method, Failure Identification for Mission Analysis (FIMA), which performs an overall and mission-specific failure analysis for complex systems. The FIMA method identifies all possible functions of a complex system and then analyzes how various failures may have differing effects on these functions. Then, the FIMA method uniquely uses this information to analyze specific mission plans, which are made up of individual mission tasks that a system must complete. The FIMA method uses multiple unique 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 system's remaining tasks and functionality. This method aims to utilize failure information to enhance the adaptability of complex systems in order to reduce the effects of failures and extend lifespans.

Keywords: Failure Analysis, Mission Analysis, Design theory

Contact:
Charlie DeStefano
University of Arkansas-Fayetteville
Mechanical Engineering
United States of America
cdestefa@uark.edu

Please cite this paper as:
Surnames, Initials: *Title of paper*. In: Proceedings of the 20th International Conference on Engineering Design (ICED15), Vol. nn: Title of Volume, Milan, Italy, 27.-30.07.2015

1 INTRODUCTION

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, which can cause great difficulty for failure analysis due to the fact that the different functions of a complex system may all experience significantly different effects from the same failure. Thus, the various use-cases of a complex system must be taken into account in order to provide accurate failure information. Unfortunately, the inclusion of these possible complex use-cases is where current methods fall short. Most current failure analysis methods, many of which are discussed in the following section, typically only identify potential component failures and how they might affect component and system-level behaviors, in a general sense. These methods do not consider a failure's effect on specific missions, however. A mission is defined here as the system's high-level use-case 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. Not taking into account such mission details greatly limits current analysis methods 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. Thus, the system's expected use after a failure occurs must be considered in order to accurately identify the effects and severity of the failure, and therefore, this paper presents a new method to do just that: the Failure Identification for Mission Analysis (FIMA) method.

Along with identifying which functions or mission tasks are still achievable after a failure occurs, it is also important to know if there are any functional or control redundancies that could help restore any 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 (Umeda et al., 1995). Understanding a system's functional and control redundancies is the first step towards improving a system's robustness. Robustness is a system's ability to adapt to failures and extend its own lifespan in order to get the most use before any external involvement is required, such as the repair or replacement of faulty components. 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, the FIMA method presented in this paper has been designed to identify when, where, and which redundancies are needed based on current mission plans.

The FIMA method described in this paper focuses on a quasi-quantitative analysis approach using performance state-machines. This method is a continuation of research done by the authors on using performance state-machines for failure analysis (DeStefano, 2014 & DeStefano and Jensen, 2014). The quasi-quantitative approach in this paper focuses on abstract failure modes, such as 'Nominal,' 'Degraded,' and 'Defective,' but allows for different degrees and types of degradation by incorporating physics-based equations to more accurately define a system's behavior during nominal and faulty conditions, such as a channel being '10% too big/small'. These new behaviors are then used to assess and optimize the system's performance for specific mission plans. This paper will first explain the theory and methodology for the FIMA method and then apply it to an example case study.

2 BACKGROUND

Two of the main failure analysis methods currently used in industry are Failure Modes and Effects Analysis (FMEA) (Nannikar et al., 2012 & Tague, 2004) and Failure Modes, Effects and Criticality Analysis (FMECA) (Reliability Analysis Center, 1993). 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. Other existing methods that do explore model-based failure analysis include the Function-Failure Design Method (FFDM) (Stone et al., 2005 & Stone et al., 2006), Function-Failure Identification and Propagation (FFIP) (Kurtoglu and Tumer, 2008 & Jensen et al., 2012), and Risk in Early Design (RED) (Grantham-Lough et al., 2008 & Grantham-Lough et al., 2009). 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. 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 failures.

3 THEORY & METHODOLOGY

The first step of the FIMA method is to create an abstract model of functional relationships and dependencies between the system's components. These functional relationships 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 component's state-machine allows for that component to switch between any of its potential performance states, such as "Nominal Performance" or "Degraded Performance," and then provide a unique output value based on which state the component is currently located. Within each potential performance state, governing equations are used to describe how the different states, or types of failures, will influence the system's behavior. For example 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 an airplane's landing gear failing open or closed. This ability 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 an airplane's 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.

Once all system components have been created and defined within the Simulink model, the user can begin simulating specific failure combinations for general system analysis, as well as specific mission analysis using a customized MATLAB code. 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 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 FIMA method's optimization technique 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 CASE STUDY: 3-LINKAGE ROBOTIC ARM

4.1. Case Study: Overview

A 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, will be the focus of this 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 in the [X,Y,Z] coordinate plane.

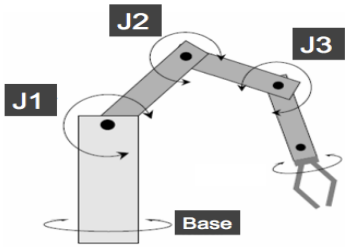

Figure 4.1 – 3-Linkage Robotic Arm Assembly

Unfortunately, when everything is working nominally there is no simple way of determining how robust 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, thus, making it seem like the better mission, 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 Case Study: Methodology

The first step was to create the Simulink state-machines 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 4.3. 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. 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 be changed to include obstacles or exclude certain types of arm movements by adding limitations within the MATLAB code. Lastly, all arm coordinates were calculated using the following forward kinematic equations within the MATLAB code:

$$r = L1*\cos(Theta1)+L2*\cos(Theta1+Theta2)+L3*\cos(Theta1+Theta2+Theta3) \quad (1)$$

$$Z = L1*\sin(Theta1)+L2*\sin(Theta1+Theta2)+L3*\sin(Theta1+Theta2+Theta3) \quad (2)$$

$$X = LR*\sin(Theta0) \quad (3)$$

$$Y = LR*\cos(Theta0) \quad (4)$$

where, 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.

Next, the MATLAB user is prompted to input 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 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.

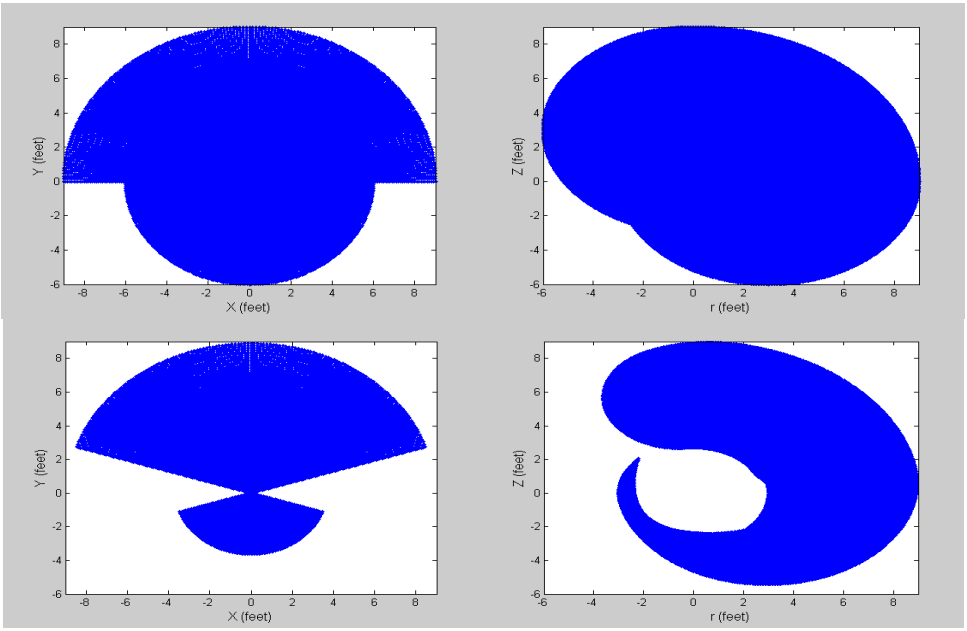

Figure 4.2 – Possible Movement Coverage for 3-Linkage Robot Arm

(Top: Nominal, Bottom: Degraded)

(Left: X-Y axis, Right: r-Z axis)

Next, the user will input the various mission 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.

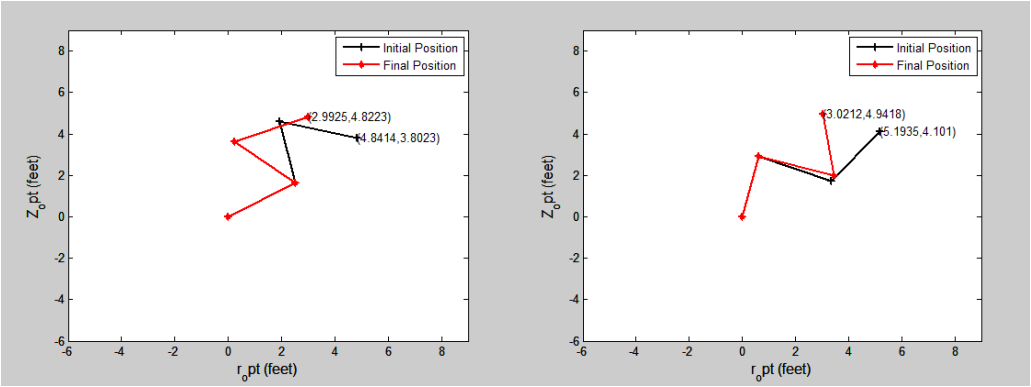

Figure 4.3 – Nominal (Left) vs. Optimized for Degradation (Right) arm positions on the r-Z axis

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.

4.3 Case Study: Results and Discussion

4.3.1 Use-Case 1: Mission Comparisons

The first use-case 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, where 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. 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.

4.3.2 Use-Case 2: Mission Adjustments

The second use-case 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 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.

5 CONCLUSIONS

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 Failure Identification for Mission Analysis (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.

6 FUTURE RECOMMENDATIONS

While, currently, 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 MATLAB code could be used for real-time optimization of a real-world physical system. 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, for the manufacturing robot in the case study, instead of the user inputting a "Percent Degraded" value prior to a mission, an actual robot would run a quick 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.

Future work will also include path-planning optimization. For this paper's 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 a failure affects the arm's joint angle combinations at various mission points, but also how a failure affects the arm's ability to avoid obstacles as it moves 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 extend in and out, and so depending on the required mission plan, the arm's path between points will need to be optimized along with the joint angles.

Lastly, future work will include validation of the models through experimentation on a physical testbed. For the case study examined in this paper, 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.

REFERENCES

DeStefano, C. (2014). Utilizing Failure Information for Mission Assessment and Optimization for Complex Systems. MSc Thesis. University of Arkansas-Fayetteville.

DeStefano, C. and Jensen, D. (2014). 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,

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

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

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

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

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

Reliability Analysis Center. (1993). Failure Mode, Effects and Criticality Analysis (FMECA).

Stone, R., Tumer, I., and Van Wie, M. (2005). The Function-Failure Design Method. In ASME Journal of Mechanical Design. Vol. 127. Pg. 397-407.

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

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

Umeda, Y., Tomiyama, T., and Yoshikawa, H. (1995). A Design Methodology for a Self-Maintenance Machine. University of Tokyo, Japan.