Proceedings of the ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference
IDETC/CIE 2017
August 6-9, 2017, Cleveland, Ohio, USA

DETC2017-67952

RESILIENT SYSTEM DESIGN USING COST-RISK ANALYSIS WITH FUNCTIONAL MODELS

Elham Keshavarzi
PhD Candidate
Mechanical Engineering
Oregon State University
Corvallis, OR, USA

Matthew McIntire
PhD Candidate
Mechanical Engineering
Oregon State University
Corvallis, OR, USA

Kai Goebel
Intelligent Systems Division
Coordinator
Ames Research Center
Moffett Field, CA, USA

Irem Y. Tumer
Professor
Mechanical Engineering
Oregon State University
Corvallis, OR, USA

Christopher Hoyle
Assistant Professor
Mechanical Engineering
Oregon State University
Corvallis, OR, USA

ABSTRACT

This paper presents a framework to compare the resiliency of different designs during the conceptual design, when information about implementation details is unavailable. We apply the Inherent Behavioral Functional Model (IBFM) tool to develop an initial functional model for a system and simulate the failure behavior. The simulated failure scenarios provide us the information on the unique failure propagation paths and the end state/final behavior of the system assigned to each failure. Each failure path is caused by injecting one or multiple simultaneous faults into the functional model. Within this framework, we generate a population of functional models from a baseline seed model, and evaluate its potential failure scenarios. We also develop a cost-risk model to compare resiliency of different designs, and produce a preference ranking. select the most resilient one, based upon the cost-risk objective. The risk is calculated based on the probability of having an undesired end state for each design, and a consequential cost is assigned to each failure to quantify the cost-risk for a given design. In this paper, we implement and demonstrate the proposed method on the design of a resilient mono-propellant system.

Keywords: resilient design, complex system, failure analysis, functional model

1. INTRODUCTION

Complex systems contain an ever increasing number of components and subsystems that interact with each other, often in unpredictable ways. Unintended interactions lead to unexpected behaviors and consequences, some of which have proven to be catastrophic. A key technical challenge in developing such complex systems is to ensure that the behavior of the system in undesired and uncertain situations is determined early in the design phase, prior to the manufacturing and operational life of the system.

To develop an engineered design, modeling and characterization of the complex systems, and simulation of the fault scenarios are necessary. Many complex engineering systems can be represented by their component model; however, their complexity is dependent on the quantity of different components as well as the formation of connections between those components, Also, when developing a new design, or in the early design phase, there is generally no component-level model available, and typically the set of components is not selected. Because of this, system properties are studied using functional models. Developing a functional model of a complex engineered system is an effective way to study the system and its behavior, and achieve a reliable fault analysis of the system.

An overall description of the design methodology is provided as follows. A functional model is the first phase of the overall design methodology. Based on the system requirements and user knowledge, functions and flows are defined. Each function can have different modes: nominal, degraded, and failed. Function, flow, and mode definitions are fed to the IBFM tool, and the simulation runs to produce all system fault scenarios. The unique fault scenarios provide the probability of undesired end states. Applying the cost-risk model the engineer evaluates the cost of a design versus the resiliency or the ability of the design to recover from a failure. If the design fulfills the requirements the process ends, otherwise, with modifications in the original functional model a new design is generated and the program runs until the algorithm achieves the lowest cost-highest resilient design. Figure 1 shows the flowchart for the presented method.

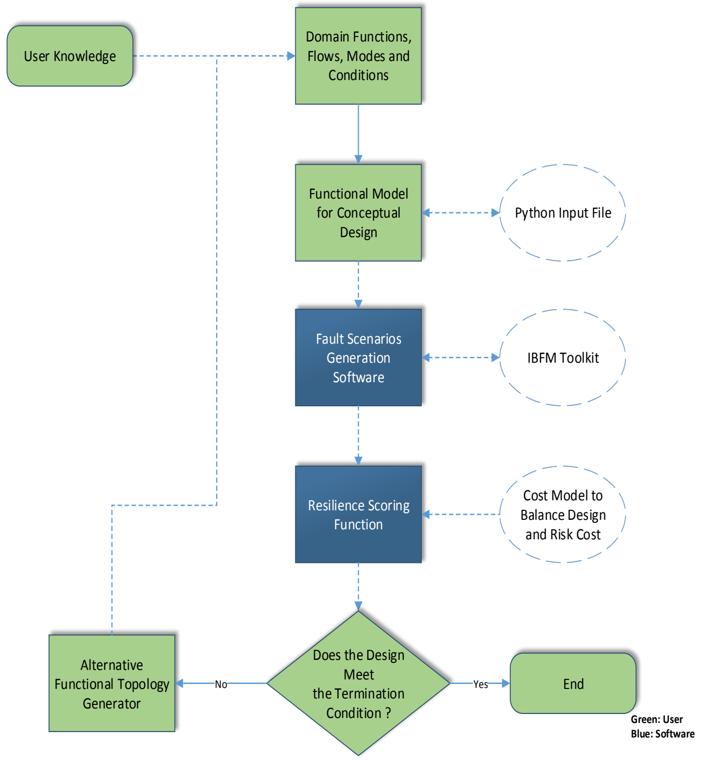
User Knowledge
Domain Functions, Flows, Modes and Conditions
Functional Model for Conceptual Design
Python Input File
Fault Scenarios Generation Software
IBFM Toolkit
Resilience Scoring Function
Cost Model to Balance Design and Risk Cost
Does the Design Meet the Termination Condition ?
No
Yes
Alternative Functional Topology Generator
End
Green: User
Blue: Software

Fig.1: Presented method flowchart

This paper provides an applicable framework to develop a resilient design. It illustrates how to develop a functional model, simulate the fault scenarios, and implement the cost-risk analysis to assess the resilient design.

The material of the paper is organized as follows: Section 2 proposes a clear and consistent definition of resilient design, and carefully disentangles it from reliability and robust design. Section 3 provides the function failure logic and behavior rule implementation. Section 4 illustrates system representation applying the functional models. Section 5 presents the scoring function. Section 6 presents generating different designs. Section 7 provides a case study of an electrical power system design, to illustrate the application of our method. Section 8 is the result summary and finally, section 9 provides conclusion and summary.

2. RESILIENT DESIGN

Resilience can be defined as a system ability to recover from a fault to maintain desired level of performance. The ability of an engineered system to recover from failure must be designed into the system [1].

Design strategies used for advancing *reliability* and *robustness* can be implemented for the purpose of advancing resilience in a system; however, there is meaningful difference between these concepts. Reliability describes the ability of a system or component to function under states conditions for a specified period of time. Reliability is defined as the probability of success or availability of a component/system [2]. Reliability concentrates more on “why and how” components or systems may fail or have failed. This information can be utilized to formulate detailed operational and maintenance manuals and procedures, as well as being applied to improve safety-critical designs. Fault Tree Analysis (FTA), Failure Modes and Effects Analysis (FMEA), and event tree analysis are known techniques to study reliability [3]. Practice has shown that the greatest criticism of these techniques is they require component-level detail throughout the system, at which point a designer may have limited ability to influence the design.

Robustness is the state where the system performance is minimally sensitive to factors causing variability either in the system or environment. [4, 5, 6]. However, the factors causing variability are not completely known and predictable when designing a system. Therefore, the ability to overcome such *uncertainties* should be embedded into the system. Uncertainty is the inability to specify something with precision [7]. Uncertainty influences designs, and system behaviors. Reducing uncertainty has been, and continues to be, a costly endeavor in time and resources [8].

Recovery from the effects of uncertain events is an alternative to reducing uncertainty. There are different ways to recover from the uncertain events: flexibility, Monitoring and Automated Contingency Management (ACM), and resiliency are known ways. Flexibility is the ability of a system to respond to changes in initial requirements and objectives, after it has been fielded, in a timely and cost effective way [9, 10, 11, 12]. An ACM system adapts autonomously and allows some degradation in the system performance when failure occurs with the goal of still achieving the mission [13]. *Resilient Design* enables engineering systems to recover from uncertain event occurrences [14]. A resilient system is one that maintains state awareness and an accepted level of operational normalcy in response to disturbances, including threats of an unexpected and malicious nature [15]. Yodo et al. provide a literature survey of existing studies in engineering resilience from a system design perspective, with the focus on engineering resilience metrics and quantification [16]. However, there is a lack of applicable methodology that shows how to design a resilient system from the early design phase.

To develop a resilient design, fault analysis is a necessary step. In most existing fault analysis methods, system designers need a precise model of system components to be able to analyze fault behavior in a complex system; however, in the early design stages, selection of specific components have not been made and such detailed models of system components are not available. Because of this lack of methods to manage risks on early stages of design, the idea of representing the complex system by only its intended functionality has been developed. The following section discusses failure analysis using functional models.

3. FAILURE ANALYSIS AND FUNCTIONAL MODELS

An engineered system is defined as an assemblage of sub-systems, hardware/software components, and people designed to perform a set of tasks to satisfy specified functional requirements and constraints [17]. The traditional approach for designing an engineered system is to establish a pre-defined set of requirements based on market studies and best estimate extrapolations of the current state, and then find the optimal design to satisfy the requirements [18]. However, these approaches are inadequate to respond to changes in initial requirements and uncertain events. This can lead to failure if the system is faced with significantly different conditions than the ones predicted. As a system becomes more complex, the uncertainty in the operating conditions increases. In such system, implementing a precise failure analysis in early design stage is vital to study system scenarios.

There are different types of failure analysis techniques for complex systems [19-27], including quantitative and probabilistic methods [28, 29], and reliability analysis techniques applied to design [30, 31], or knowledge-based approaches such as lessons learned databases and hazard analyses [32]. Studies have shown that the early design stages are the best times to catch potential failures and anomalies [33]. However, in the early design phase, decisions about the specific set of components is not made, and a component model is not available. The idea of using functional model, instead of component model, to design a complex system is of increasing interest. A functional model in systems engineering is a structured representation of the functions required to meet system requirements. The purpose of a functional model is to describe the system behavior and determine vulnerable parts of the design, resulting in potential system improvement.

Methods have been developed to combine functional modeling with failure analysis. Stone and Tumer developed the Function Failure Design Method (FFDM) which was the bridge between failure analysis and functional modeling. They applied functional models to represent the system design and identify possible failure modes for a given function [34, 35]. Grantham et al. developed the Risk in Early Design method to formulate functional-failure likelihood. Based on the consequence of failures, functions are classified as high-risk to low-risk [36, 37]. The idea of providing function-based failure analysis represented progress in the design of complex systems; however, these methods limit the designer of the system to only consider one single-fault impact analysis at a time.

To overcome this restriction and to enable the designer to investigate the effects of multiple function failures, the Function Failure Identification and Propagation (FFIP) method was developed [38-47]. The FFIP method identifies failure propagation paths by mapping component failure states to function health. This approach uses a separate behavioral model simulation to determine fault propagation and fault effect. Typically, the modeling language *Modelica* is applied to simulate failure scenarios [48, 49], however, system models cannot be automatically constructed from a description of the functional structure of a system and therefore it may not be useful in FFIP where multiple designs are investigated.

McIntire et al. have created The Inherent Behavior of Functional Models (IBFM), a new function failure reasoning method which generalizes failure behavior directly from functions. With this method, an engineer can create functional models to simulate the functional failure propagations a system may experience early in the design process without a separate behavioral model [50]. An open access IBFM tool in Python has been developed by our group at Oregon State University to simulate the behavior of the system as a hierarchical state machine. In the IBFM tool, one avoids equation-based physical behavioral modeling in favor of discrete event based modeling using a hierarchical state machine. Not only does this allow for incomplete knowledge of specific system implementation, but it allows for very fast simulation of a huge number of failure scenarios due to the lack of time-based differential equation solvers [50].

In the presented framework, we apply the IBFM tool to simulate the fault scenarios for different design topologies. The following section describes the functional models in detail.

4. SYSTEM REPRESENTATION BY FUNCTIONAL MODEL

The first step in our proposed method is to study the requirements and expectations from the system and define the functions and flow. We use a graph to represent the system functionality and its interaction with the environment. We implement the method in Python, to be used in conjunction with other function-failure analysis tools, such as representing the graphical model by applying NetworkX [51]. Models as Python graph objects help users quickly model multiple functional architectures of their proposed system.

Each graph edge (arrow) represents a flow of material, energy, or information within the system and each graph node (rectangle) represents a function that acts on the flows intersecting it. Figure 2 provides an example of a graph representation of a functional model consisting of a single internal function and three functions.

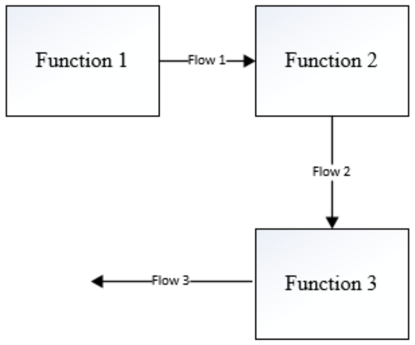

Fig.2: Graphical representation of a functional model

In this paper, a flow is defined similar to a *bond* used in the bond graph approach [50]. As in a bond graph, the flow has a direction, and uses two variables to define the flow: an *effort* variable, and a *rate* variable. For example, a liquid flow can be modeled as either a liquid pressure (effort) or a liquid volume flow rate (rate). The function at one end of the flow controls the effort state, while the function at the other end controls the flow rate. In the IBFM approach, effort and rate variables take qualitative values, such as Zero, Low, Nominal, and High.

Each function consists of a set of *modes*. Mode definitions show the different levels of functionality for a function, which are usually categorized as operational, degraded and failed modes. *Conditions* determine how the flows go from one function to another and basically provide the conditions to generate the failure paths. Conditions define the transition between modes, e.g. the transition from a nominal to degraded state, or from a degraded to failed state. All modes and conditions are qualitative, rather than quantitative in the IBFM approach. The conditions that regulate a function transitioning from one mode to another mode are also flow specific. For example, the operational mode of the function "Regulate Gas Pressure" has a "Gas High-Pressure" condition that leads to a failed mode. The "Gas High-Pressure" condition is only used by functions that have a "Gas" flow. Functions, flows, modes, behaviors and conditions are defined to construct a functional model to be fed to the IBFM tool to simulate all system fault scenarios.

The Inherent Behavior in Functional Models (IBFM) module in Python [50] is applied to simulate the fault scenarios. The IBFM tool is hosted on the Oregon State University Design Engineering Lab GitHub site at https://github.com/DesignEngrLab/IBFM. The repository contains the module `ibfm.py`, which includes classes defining each component that make up the program structure: Experiment, Model, Function, Flow, Mode, and Condition. It also contains the user guide and example scripts which run example models demonstrating the simulator [50]. The IBFM tool simulates the functional effects of single and multiple simultaneous faults. An exhaustive list of scenarios comprising fault paths is constructed by single fault, or two or more simultaneous faults.

The simulation of each fault scenario begins at that nominal model state, and then changes the mode of one or more functions to a degraded or failed mode. The end state of each scenario is recorded for further analysis. The number of unique paths that have a particular undesired end state is used in the cost-risk model described in the following session.

5. RESILIENT SCORING FUNCTION

We have developed a cost-risk model to evaluate the resiliency of different designs (Equation 1). The idea of this model is rooted in the risk-based utility theory [52]. This model studies the tradeoffs between the cost of designing a system that can recover from a failure, versus the cost of designing a system without a recovery while accepting the inherent risk of failure. The key element is that the "cost of risk" can be quantified, such that the trade-off between adding system cost versus accepting risk can be made. The proposed model is composed of three cost elements: the baseline cost of the design, the cost of mitigation, and the cost of risk, given by Eq. (1):

$$\text{Min } (C_D + C_O) + C_M + \sum_{i=1}^{N} C_{R,i} p_{R,i} \left(1 - p_{M,i}\right) \tag{1}$$

Where:

C_D = Baseline cost of the design

C_O = Operation cost

C_M = Cost of mitigation

C_R = Consequential cost of the risk

p_R = Probability of risk

p_M = Probability of mitigation

$1 - p_{M,i}$ = Probability of mitigation failure

N = Number of undesirable end states

C_D is the cost of design when there are no strategies to recover from failure embedded in the design: in other word the design is not resilient. C_M, represents the cost of changing the design to be able to recover from a particular failure; there are different ways to change a functional model to design a more resilient system and in the next session we describe four ways. Independent of the quality of performance, there is a certain cost to operate a system, defined as operation cost, C_O.

With respect to defining the "cost of risk", we define a risk as a triplet of its *impact* or consequence, its probability of fault *occurrence,* and its probably of being *mitigated*. In Eq. (1), C_R is the impact or consequential cost of the risk; in our approach, it is quantified as the cost of having a failure and not recovering from it (in units of dollars). Secondly, p_R is the probability of having a specific undesired end state or fault, and is quantified using the results of the IBFM fault simulation. It is quantified as the number of unique scenarios with a particular undesired end state (or fault) divided by total number of unique scenarios. Lastly, p_M is the probability a resilient design recovers from a failure due to a mitigation action (a mitigation action is assumed to have a cost of C_M). Probability of mitigation is also calculated from the IBFM simulation result, by adding the mitigation action as a function to the functional model and rerunning the simulation. N is the number of system end states that the system is designed to consider. Except for the nominal scenario when all functions perform nominally, other end states are fault scenarios and undesired. However, an undesired end state does not imply that a safety hazard is present, it may just reflect an end state that prevents the system from nominal operation or from completing a mission. For instance, when designing a car, having a flat tire is undesirable, but may not be classified as a safety hazard; however, an engine fire is both an undesirable state and a safety hazard.

Applying the cost-risk model, the designer specifies the tradeoff between the resiliency and cost of the design; the cost-risk model is treated as an objective function in an optimization framework. In the optimization formulation, the objective is to minimize total system cost (i.e., Eq. (1)), subject to any system-level constraints. Treating the search for the best system-level design as an optimization problem necessitates identifying a *termination condition* for the search. In application, the search would most likely be done by a heuristic or stochastic search method, given the discrete nature of the functional model representation of the system. Since these methods cannot be guaranteed to converge to the optimum in finite time, the search must be ended based on a heuristic. Termination Conditions for the proposed framework in Fig.1 can be defined as:

- An upper limit on the number of evaluations (designs) is reached.
- An upper limit on the number of evaluations of the fitness function (cost-risk model) is reached.
- The chance of achieving significant changes is very low.

If the design meets one or more of the termination conditions, it would be selected, otherwise a new design is generated and evaluated.

6. GENERATE NEW DESIGNS

As noted in Section 5, the cost-risk approach is implemented as a search problem, in which the objective is to find the lowest cost design. An issue to address is how to change of modify the design in a way to produce feasible designs in the search process. The challenge becomes one of identifying alternate functional models which are technically feasible, i.e., functional models which preserve the key system functions and the associated flows. Therefore, a method is needed to ensure that functional models evaluated in the search process are technically feasible. Graph grammars [50] could be implemented to address this issue and are compatible with the cost-risk approach. In this work, we define design strategies to generate new design based upon a few simple rules used by designers of aerospace systems. The design modification rules can be placed into four categories as follows:

1. *Redundancy and Health Management*

In this technique, redundancy or health management is added to the functional model. The redundancy could be the addition of a redundant function, or it could be added in the form of partial redundancy. In the case of partial redundancy, we may be able to fulfill the needed functionality using secondary functions from a specific component. For example, we may be able to use a pressure sensing function to also indirectly fulfill a flow rate sensing function, in the case that the flow rate sensing function is faulted. Adding redundancy or health management will affect C_M and p_M in Eq. 1; the tradeoff will in general be one in which mitigation cost is added to increase the probability of mitigation (and thus reduce the cost of risk).

2. *Change the order of the functions* (change to p_R)

Changing the order of functions is another way to generate a new design at the conceptual level of the design. This change affects the probability of the risk, p_R, by focusing on fault avoidance; i.e., arranging the functions is such a way that fault propagation path is changed and thus the probability of a risk is changed.

3. *Using unused flows as new inputs (using unused flows as inputs)* (changes C_D and p_R, or C_M and p_M)

In this method, any flows that transfer material or energy to the environment (i.e., are not used by the system directly) are utilized as additional inputs for other functions where applicable. This improves the fault avoidance aspect of the design. For example, one could use waste heat to supplement a heating function as a mitigation function (C_M and p_M), or one could lower the cost of design of a heating function (C_D) by coupling waste heat from a different function with the heating function.

4. *Combining functions (or splitting functions)*

Two or more functions can be combined (or split) to make a new design and potentially improve the fault avoidance of a system. This could affect both C_D and p_R. whether combining two or more functions into a single function (or splitting a single function into two or more functions) improves or worsens the cost-risk objective function must be determined by the results of an IBFM simulation. Combining functions may lead to elimination of a fault path (thus reducing p_M), or may make the new combined function more likely to fail (thus increasing p_M).

We can generate an infinite number of functional models from a single seed functional model. The following section presents a case study using a monopropellant propulsion system design to show how the methodology can be applied in a system design problem.

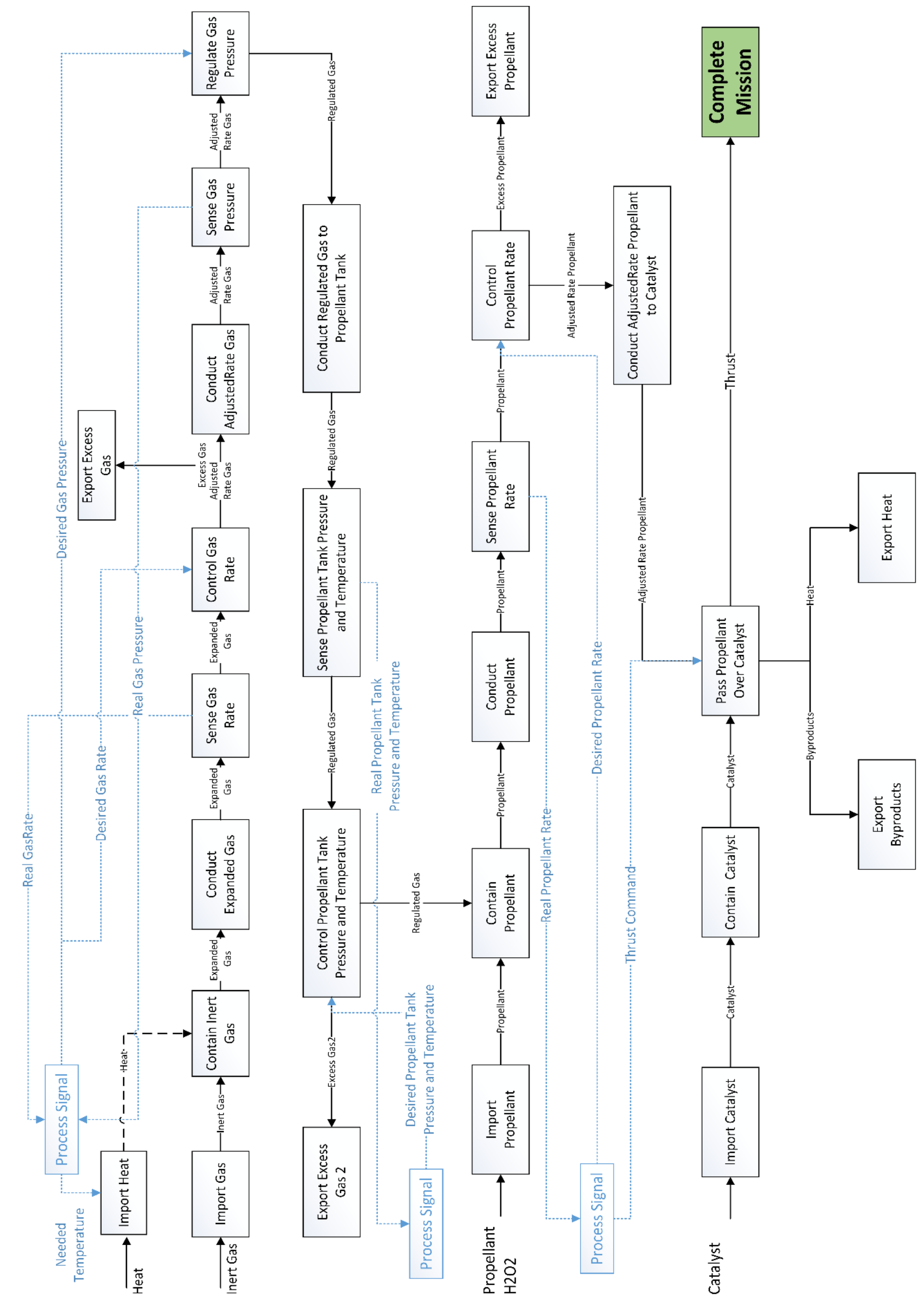

Fig.3 Functional model for monopropellant propulsion system

7. CASE STUDY: MONOPROPELLANT PROPULSION SYSTEM

A monopropellant propulsion system refers to a chemical propulsion fuel which does not require a separate oxidizer, and thus can be used in space. Monopropellant designs are typically used in the aerospace industry because it makes the engine lighter, less expensive, and more reliable. In this case study, the monopropellant is hydrogen peroxide (H_2O_2).

In designing a monopropellant system, outer space environmental conditions should be considered. With no gravity assistance, the system should be able to push the propellant towards the catalyst. The concept for this system design is to apply expanded gas to push the monopropellant over a catalyst and produce thrust. This system can be divided into three main subsystems; gas, propellant, and catalyst.

When there is a command for a change of the aircraft velocity, the inert gas is *heated* to expand. The expanded gas is fed through *regulation* and *control* functions to reach the right quantity, pressure, and temperature. The expanded gas places pressure on the propellant (hydrogen peroxide) and guides it to the catalyst. When the propellant passes the catalyst, combustion occurs and changes the velocity of the aircraft. Figure 3 presents the detailed functional model of the system.

When the thrust is commanded, if all functions are *nominal*, the system performs as expected; however, there are numerous scenarios in which something can go wrong with one or more functions and the result is not as expected. This would cause the engine to provide too much, too little or no thrust. In practice, it means the aerospace system passes the location, or does not reach it. In rare catastrophic failures, the system might catch on fire or explode (i.e. loss of system). To monitor system outputs, all sault scenarios are simulated by IBFM.

To apply IBFM tool, the first step is to define the functions, flows, modes and conditions in Python. The advantage of defining the functions and flows in Python is that the Python NetworkX tool can be used to represent the functional model graphically. The following session describes how to develop a functional model in IBFM tool and simulate all fault scenarios.

7.1. Functions

The monopropellant propulsion system shown in Fig.3 contains 29 functions and 129 modes. One example of how to define a function for mono propellant propulsion system is as follows:

```
function ImportHeat
      mode 1 Operational NominalHeatSource
      mode 2 Degraded LowHeatSource
      mode 3 Degraded HighHeatSource
      mode 4 Failed NoHeatSource
```

The first line contains the keyword function, followed by the desired name of the function. The indented lines contain all of the modes and conditions of the function. Each line describing a mode begins with the keyword mode, followed by a unique within-function alphanumeric identifier, followed by the function health associated with the mode, followed by the name of the mode. Available mode health states are Operational, Degraded, and Failed. A single mode in each function definition may be followed by the keyword default, which assigns that mode to be the initial mode of the function at the beginning of simulations.

7.2. Flows

Flows are defined in a single line. The definition always begins with the keyword flow, followed by the desired name of the flow, followed by the name of its parent flow type. The top level flows are Material, Energy, and Signal. All other flows derive from them.

```
flow Heat Energy
```

7.3. Modes

Mode definitions are more complicated, as all of the mode's behaviors must be explicitly described. Operational, degraded, and failed modes are required to be defined for each function. A simple mode definition example is the nominal gas source mode:

```
mode NominalGasSource
    InertGas output effort = Nominal
```

The first line consists of the appropriate keyword, in this case mode, followed by the desired name of the mode. Each indented line consists of a single assignment statement. The expression to the left of the assignment operator = is evaluated to determine the flow variable(s) being assigned to. The expression to the right of the assignment operator is evaluated to determine the state(s) to assign to the flow variable(s). Every flow in the statement must be referred to using three words: the flow type name, its direction, either input or output, and its variable, either effort or rate. More complex behaviors may be defined by using operators. A single unary operator is used in the definition of the "NoGasSource" mode:

```
mode NoGasSource
    InertGas output effort = Zero
```

This mode definition makes use of a constant state. Available states are Zero, Low, Nominal, High, and Highest. The definition of the drifting low pressure sensing mode uses two unary operators:

```
mode DriftingLowPressureSensing
    import NominalConductingRegulatedGas
    SignalDesiredPressure output effort =
RegulatedGas input effort --
```

The first one, the keyword import, copies all of the statements from the definition of the mode directly following the keyword. In this case, the two statements from the “NominalConductingRegulatedGas” mode are copied into “DriftingLowPressureSensing”. The second one, the decrement operator --, decreases the value of the state by one qualitative level.

7.4. Conditions

Condition definitions are similar to mode definitions. They name the condition being defined, and explicitly describe the behavior, but they only include a single behavior statement. Rather than being an assignment, the statement is a logical test. For example, the condition to test for a function being exposed to high temperature is:

```
condition HighTemperature
    Heat output effort > Nominal
```

Logical operators may be combined to form more complex tests. All binary operators are evaluated from left to right, so parentheses may be required.

Running the IBFM tool, all fault scenarios are produced and the unique scenarios terminating with particular undesired end state are tabulated. The final performance of the system is categorized into different main end states. The designers of the system decide about what end states the system failures could cause. In this case study, undesired end states are:

- Pass the Mission Location
- Do Not Reach to Mission Location
- No System Movement
- Loss of the System

7.5. Alternative system designs

The baseline design, shown in Fig.3, represents an early stage design of the functions that the mono propellant system is required to perform. In the baseline system, the normal operating condition assumes that all gas, propellant, and catalyst functions operate nominally. Any change or distribution will have some effect on the system; in other words, the initial design is not designed to be fault tolerant. However, the same set of functional design requirements can be met by designing different system topologies representing fault tolerant behavior by employing various levels of redundancy and reconfigurability. In addition, the sensor allocation related to the integrated health management functionality can be made several ways.

In this case study, we analyze the basic baseline design and compare it to six alternative conceptual system designs. These alternatives demonstrate different levels of risk mitigation in different functional areas of the system. The first modification of the baseline design includes a redundant gas rate sensor, if no signal is coming from the primary sensor, a secondary sensor can supply the required information. The second modification of the baseline design is an identical system with redundant gas pressure sensor. The third modification of the baseline design combines the adjusting pressure and rate into one function. The forth design uses output heat to expand inert gas, in the baseline the heat is exported from the system and disappear in the space. The fifth design incorporates the redundant the sensor in propellant tank. The sixth design applies the output thrust heat to preheat propellant, the propellant would play the role of cooling system. The following session presents the results.

8. RESULTS OF THE CASE STUDY

For the monopropellant propulsion system, six candidate designs are investigated. In each design, the functional model has been modified to be more resilient to a particular failure or end state. Table 1 shows all the designs generated by the functional model modifications.

Table 1: Generated designs for monopropellant propulsion system

Basic Design	The original unaltered system design
Design 1	Redundant gas rate sensor added
Design 2	Redundant gas pressure sensor added
Design 3	Combine adjusting gas pressure and rate into one function
Design 4	Use output thrust heat to expand inert gas
Design 5	Redundant propellant tank pressure sensor added
Design 6	Use output thrust heat to preheat propellant (propellant plays the role of cooling material)

All system scenarios have been simulated using the IBFM tool. The amount of CPU time required to simulate each scenario increases as the complexity and the number of faults injected to the model increases. For instance, the time of simulation for injecting two simultaneous faults is less than the time of simulation for having three simultaneous faults. In this study, we investigate injecting up to 3 faults in each one of the functional models. The classified simulated scenarios for the seed and the six design topologies is shown in Fig.4. In this histogram, the number of successful and failed scenarios simulated for each design is different.

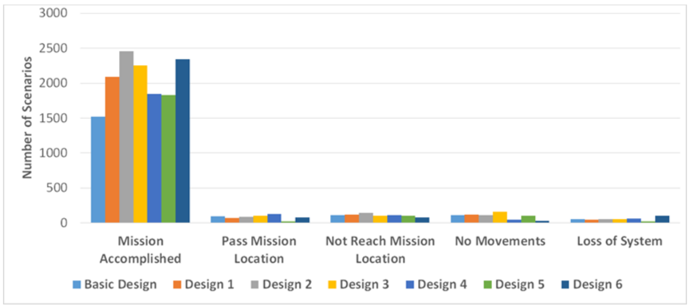

Fig.4: Simulation results for different monopropellant propulsion system designs

The probability of risk in each design is applied to the cost-risk model of Eq. 1. The mitigation cost C_M is the cost of adding to or changing the basic design to make it more resilient to a particular failure or undesired end state. The operation cost C_O for all candidate design is assumed to remain the same, based on the assumption that regardless how the system is performing, there is an on ground system to monitor, control and run the aircraft. For other design studies, the operating cost may vary by design concept. It is assumed that the aircraft completes three missions per year, and for each mission the operation cost is $500 million. The number of failed scenarios versus the total number of scenarios for each design defines the probability of risk, p_R, for each design. The probability of the mitigation failure, p_M , is the probability of the redundant part does not work when needed, these probabilities and The costs for having an undesired performance, C_R, are simulated in this study. The total cost reflects the tradeoff between making a resilient design and the costs, the lower total cost is the higher rank the design would be.

The results of cost-risk optimization for the monopropellant system are shown in Table 2. In the table, we tabulate all the elements of Eq. 1. Since there were only seven designs to consider, an optimization algorithm was not used; however, an evolutionary of other stochastic algorithm could be used to search larger design spaces. As shown the table, **Design 6** has the best tradeoff between resilience and cost; in this design the propellant acts as a cooling system and the preheating helps produce thrust. This design was selected because it had the lowest value of the cost-risk objective function. The next best design is **Design 3**, which combines the gas pressure and rate control into a single function. This may indicate that since both functions are critical to operation, and a failure of either function is highly detrimental to the system, it is better to address them with a single function with a single failure rate.

Table 2: Scoring function for monopropellant propulsion system (costs are simulated in Millions $)

Cost-Risk Model	*Basic Design*	*Design 1*	*Design 2*	*Design 3*	*Design 4*	*Design 5*	*Design 6*
Cost of Basic Design C_D	100	100	100	95	90	100	80
Mitigation Cost C_M	0	10	10	0	0	20	0
Operation Cost C_O	1500	1500	1500	1500	1500	1500	1500
Cost of Risk $\boldsymbol{C_R p_R (1 - p_M)}$	45.50	32.20	30.75	34.90	40.70	21.30	35.20
Total Cost	1645.50	1642.25	1640.75	1629.90	1630.70	1641.30	1615.20
Resilient Score	**Rank 7**	**Rank 6**	**Rank 4**	**Rank2**	**Rank 3**	**Rank 5**	**Rank 1**

9. CONCLUSION

This paper presented a framework for resilient design based upon the evaluation of different designs during the early design stage. We created a framework to present an initial functional model of the system and evaluate its potential failure scenarios. To produce a preference ranking of designs, we developed a cost risk-model to generate a resilient score for each design. The central idea is that each design is evaluated based upon the trade-off between the cost of risk and embedding resiliency into the design of the system. The design implications of using this framework for the development of resilient engineered systems were discussed, with the focus on the functional model development, failure simulation, and cost-risk analysis to rank different designs and choose the most resilient one. The method was applied in the case of mono propellant propulsion system design. The results show that the presented method is an applicable engineering resilience analysis and design framework that can be used for system design.

While we have presented the general framework, future work is needed to improve the current approach. A primary issue is that the current approach does not address uncertainty in costs or failure estimation. This could be addressed by quantifying such uncertainties with probability distribution functions and creating a utility function, with the cost-risk equation as the selection criterion. Another issue is that we have only demonstrated this approach on a small design space and on a single system design problem. Graph grammars could be implemented to help generate a large number of design alternatives. Finally, visualization of the functional models generated and the results would be necessary for development of a tool which implements the proposed method.

ACKNOWLEDGMENTS

This research was funded in part by the National Science Foundation (project number CMMI 1363509) and NASA Grant and Cooperative Agreement NNX15AQ90G. The opinions, findings, conclusions, and recommendations expressed are those of the authors and do not necessarily reflect the views of the sponsor.

REFERENCES

[1] Yodo, Nita, and Pingfeng Wang. "Resilience allocation for early stage design of complex engineered systems." *Journal of Mechanical Design* 138.9 (2016): 091402.

[2] Barlow, Richard E. *Engineering reliability*. Society for Industrial and Applied Mathematics, 1998.

[3] Stamatis, Dean H. *Failure mode and effect analysis: FMEA from theory to execution*. ASQ Quality Press, 2003.

[4] Taguchi, Genichi, Subir Chowdhury, and Shin Taguchi. Robust engineering. *McGraw-Hill Professional*, 2000.

[5] Murphy, Terrence E., et al. "Robust engineering design. NSF conference, DMII division, January 2004."

[6] Beyer, Hans-Georg, and Bernhard Sendhoff. "Robust optimization–a comprehensive survey." *Computer methods in applied mechanics and engineering* 196.33 (2007): 3190-3218.

[7] Antonsson, Erik K., and Kevin N. Otto. "Imprecision in engineering design." *Journal of Vibration and Acoustics* 117.B (1995): 25-32.

[8] Helton, Jon C. "Uncertainty and sensitivity analysis in the presence of stochastic and subjective uncertainty." *Journal of Statistical Computation and Simulation* 57.1-4 (1997): 3-76.

[9] Chen, Wei, and Kemper Lewis. "Robust design approach for achieving flexibility in multidisciplinary design." *AIAA journal* 37.8 (1999): 982-989.

[10] Saleh, Joseph H., Gregory Mark, and Nicole C. Jordan. "Flexibility: A Multi-Disciplinary Literature Review and a Research Agenda for Designing Flexible Engineering Systems." *Journal of Engineering Design* 20.3 (2009): 307–323.

[11] Mark, Gregory T. "Incorporating flexibility into system design: a novel framework and illustrated developments." *Massachusetts Institute of Technology*, 2005.

[12] Keshavarzi, Elham, Matthew McIntire, and Christopher Hoyle. "Dynamic Design Using the Kalman Filter for Flexible Systems With Epistemic Uncertainty." *ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers, 2015.

[13] Saxena, Abhinav, et al. "Automated Contingency Management for Propulsion Systems." *Control Conference (ECC), 2007 European*. IEEE, 2007.

[14] Hollnagel, Erik, David D. Woods, and Nancy Leveson. *Resilience engineering: Concepts and precepts*. Ashgate Publishing, Ltd., 2007.

[15] Li, Junxuan, and Zhimin Xi. "Engineering Recoverability: A New Indicator of Design for Engineering Resilience." *ASME 2014 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers, 2014.

[16] Yodo, Nita, and Pingfeng Wang. "Engineering Resilience Quantification and System Design Implications: A Literature Survey." *Journal of Mechanical Design* 138.11 (2016): 111408.

[17] Suh, Nam P. "Axiomatic design theory for systems." *Research in engineering design* 10.4 (1998):189-209.

[18] Siddiqi, Afreen, and Olivier L. de Weck. "Modeling Methods and Conceptual Design Principles for Reconfigurable Systems." *Journal of Mechanical Design* 130.10 (2008): 101102.

[19] Tamilselvan, Prasanna, and Pingfeng Wang. "Failure diagnosis using deep belief learning based health state classification." *Reliability Engineering & System Safety* 115 (2013): 124-135.

[20] Hawkins, P. G., and David J. Woollons. "Failure modes and effects analysis of complex engineering systems using functional models." *Artificial intelligence in engineering* 12.4 (1998): 375-397.

[21] Li, Wenyuan. *Risk assessment of power systems: models, methods, and applications*. John Wiley & Sons, 2014.

[22] Youn, Byeng Dong. *Advances in reliability-based design optimization and probability analysis*. Diss. The University of Iowa, 2001.

[23] Standard, Military. "Procedures for performing a failure mode, effects and criticality analysis." *MIL-STD-1629, November, AMSC Number N3074* (1980).

[24] Vesely, William E., et al. *Fault tree handbook*. No. NUREG-0492. Nuclear Regulatory Commission Washington DC, 1981.

[25] Zang, T. A., et al. "Needs and Opportunities for Risk-Based Multidisciplinary Design Technologies for Vehicles." *NASA TM, July* (2002).

[26] Backman, B. "Design Innovation and Risk Management: A Structural Designer's Voyage into Uncertainty." *ICASE Series on Risk-based Design* (2000).

[27] Liu, Hu-Chen, Long Liu, and Nan Liu. "Risk evaluation approaches in failure mode and effects analysis: A literature review." *Expert systems with applications* 40.2 (2013): 828-838.

[28] Smith, Natasha, and Sankaran Mahadevan. "Probabilistic methods for aerospace system conceptual design." *Journal of spacecraft and rockets* 40.3 (2003): 411-418.

[29] Greenfield, Michael A. "NASA's use of quantitative risk assessment for safety upgrades." *Space safety, rescue and quality 1999-2000* (2001): 153-159.

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

[31] Xi, Zhimin, and Ren-Jye Yang. "Reliability analysis with model uncertainty coupling with parameter and experiment uncertainties: a case study of 2014 verification and validation challenge problem." *Journal of Verification, Validation and Uncertainty Quantification* 1.1 (2016): 011005.

[32] Ericson, Clifton A. *Hazard analysis techniques for system safety*. John Wiley & Sons, 2015.

[33] Mahadevan, Sankaran, Natasha L. Smith, and Thomas A. Zang. "System risk assessment and allocation in conceptual design." (2003).

[34] Kurtoglu, Tolga, and Irem Y. Tumer. "Ffip: A framework for early assessment of functional failures in complex systems." *ICED, Cite des Sciences et de L'industrie, Paris, France* (2007).

[35] Stone, Robert B., Irem Y. Tumer, and Michael E. Stock. "Linking product functionality to historic failures to improve failure analysis in design." *Research in Engineering Design* 16.1-2 (2005): 96-108.

[36] Lough, Katie Grantham, Robert B. Stone, and Irem Tumer. "Implementation procedures for the risk in early design (red) method." *J Ind Syst Eng* 2.2 (2008): 126-143.

[37] Lough, Katie Grantham, Robert Stone, and Irem Y. Tumer. "The risk in early design method." *Journal of Engineering Design* 20.2 (2009): 155-173.

[38] Jensen, David C., Irem Y. Tumer, and Tolga Kurtoglu. "Modeling the propagation of failures in software driven hardware systems to enable risk-informed design." *ASME 2008 International Mechanical Engineering Congress and Exposition*. American Society of Mechanical Engineers, 2008.

[39] Jensen, D., Irem Y. Tumer, and Tolga Kurtoglu. "Design of an electrical power system using a functional failure and flow state logic reasoning methodology." *San Diego, CA* (2009).

[40] Jensen, David, Irem Y. Tumer, and Tolga Kurtoglu. "Flow State Logic (FSL) for analysis of failure propagation in early design." *ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers, 2009.

[41] Krus, Daniel, and Katie Grantham Lough. "Applying function-based failure propagation in conceptual design." *ASME 2007 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers, 2007.

[42] Kurtoglu, Tolga, and Irem Y. Tumer. "A graph-based fault identification and propagation framework for functional design of complex systems." *Journal of mechanical design* 130.5 (2008): 051401.

[43] Kurtoglu, Tolga, Irem Y. Tumer, and David C. Jensen. "A functional failure reasoning methodology for evaluation of conceptual system architectures." *Research in Engineering Design* 21.4 (2010): 209-234.

[44] Papakonstantinou, Nikolaos, et al. "Capturing Interactions and Emergent Failure Behavior in Complex Engineered Systems at Multiple Scales." *ASME 2011 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers, 2011.

[45] Sierla, Seppo, et al. "Early integration of safety to the mechatronic system design process by the functional failure identification and propagation framework." *Mechatronics* 22.2 (2012): 137-151.

[46] Tumer, Irem, and Carol Smidts. "Integrated design-stage failure analysis of software-driven hardware systems." *IEEE Transactions on Computers* 60.8 (2011): 1072-1084.

[47] Coatanéa, Eric, et al. "A framework for building dimensionless behavioral models to aid in function-based failure propagation analysis." *Journal of Mechanical Design* 133.12 (2011): 121001.

[48] de Kleer, Johan, et al. "Fault augmented modelica models." *The 24th International Workshop on Principles of Diagnosis*. 2013.

[49] Minhas, Raj, et al. "Using fault augmented modelica models for diagnostics." *Proceedings of the 10 th International Modelica Conference; March 10-12; 2014; Lund; Sweden*. No. 96. Linköping University Electronic Press, 2014.

[50] McIntire, Matthew G. *From Functional Modeling to Optimization: Risk and Safety in the Design Process for Large-Scale Systems*. Diss. 2016.

[51] Clauset, Aaron. "Five Lectures on Networks." (2014).

[52] Van Bossuyt, Douglas, et al. "Risk attitudes in risk-based design: Considering attitude using utility theory in risk-based design." *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 26.04 (2012): 393-406.