International Journal of Research and Engineering
ISSN: 2348-7860 (O) | 2348-7852 (P) | Vol. 5 No. 9 | September-October 2018 | PP. 535-546
Digital Object Identifier DOI® http://dx.doi.org/10.21276/ijre.2018.5.9.7

Failure Analysis in Conceptual Phase toward a Robust Design: Case Study in Monopropellant Propulsion System

Author(s): [1]*Elham Keshavarzi, [2]Kai Goebel, [3]Irem Y. Tumer, [4]Christopher Hoyle
Affiliation(s): [1,3,4] Dept. of Mechanical Engineering, Oregon State University
Corvallis, Oregon, USA
[2]Dept. of Intelligent Systems Division, NASA Ames Research Center
Moffett Field, California, USA
Corresponding Author: Keshavae@oregonstate.edu

ORIGINAL ARTICLE

Abstract - As a system becomes more complex, the uncertainty in the operating conditions increases. In such a system, implementing a precise failure analysis in early design stage is vital. However, there is a lack of applicable methodology that shows how to implement failure analysis in the early design phase to achieve a robust design. The main purpose of this paper is to present a framework to design a complex engineered system resistant against various factors that may cause failures, when design process is in the conceptual phase and information about detailed system and component is unavailable. Within this framework, we generate a population of feasible designs from a seed functional model, and simulate and classified failure scenarios. We also develop a design selection function to compare robust score for candidate designs, and produce a preference ranking. We implement the proposed method on the design of an aerospace monopropellant propulsion system.

Keywords*: *failure analysis; robust design; design complex systems; conceptual design; cost-risk analysis

I. INTRODUCTION

The number of parts in complex engineered systems is increasing rapidly. For instance, the components only in an integrated system is expected to be double every year as Gordon Moore predicted properly. Therefore, the interaction between various parts of the system, and between the system and external factors gets more complicated as technology and market demand is changing. The complicated interactions cause different type of uncertainties, and unexpected uncertainties can cause undesired behavior of the system or even catastrophic failures.

An important concept in designing complex engineered 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. It requires to conduct a failure analysis in the conceptual design phase when the component model of the system and design specifications have not been developed yet. Failure analysis in early design phase help the designers to find strategies to improve the design to enable the system cope with the uncertainties during the operational life, as well as reduce the design revisions in further design steps shapers [1]

To study the failure behavior of a system in the early design stage, modeling and simulation of the failure scenarios are the necessary steps. In many complex engineered systems, the characterization of the system is represented using a component model. However, when developing a new design, or in the early design phase, there is no component-level model available, and typically the set of components is not selected. Because of this, we came up with the idea of using functional model to study the system's failure behavior in early design phase, to achieve a robust design.

An overall description of our proposed design methodology is provided as follows. Developing a functional model is the first step. To develop a functional model for a complex engineered system, all the functions and flows and operational modes should be defined based on the system requirements and expert knowledge. Each function can have different operational modes: nominal, degraded, and failed. Next step is simulation of the system failure behavior. We have developed an open access tool in Python to simulate failure scenarios using functions, flows, and modes. The

unique failure scenarios provide information on the probability of having undesired end states. Applying the cost-risk model, the designer evaluates the cost of a design versus the robustness of the design against a failure. If the design fulfills the requirements, the process ends. Otherwise, a new design is generated by modifying the functional model. The program runs until the search algorithm achieves the design with the optimal robust score. Fig. 1 shows the flowchart for our proposed method. Green boxes represents the steps that are required to be done by designers and blue boxes are produced by software.

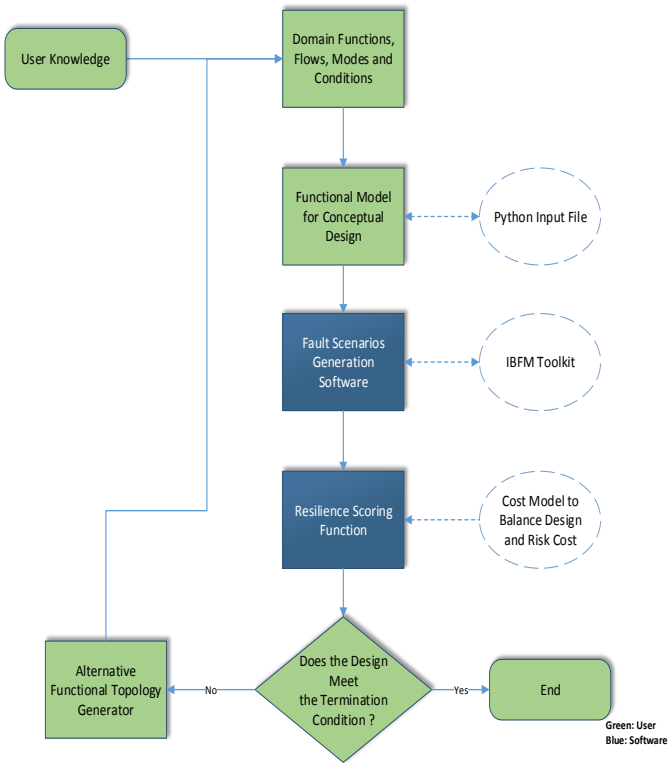

Fig. 1. Proposed framework for robust design in conceptual phase

This paper provides an applicable framework that shows the designers how to implement failure analysis and achieve a robust design in early design phase, when information about components is not available. We illustrate how to develop a functional model, simulate failure behavior of the system, and evaluate robustness of a design in early design phase. We implement the proposed approach in early design phase of an aerospace monopropellant propulsion system and achieve the design with optimal cost and resistance against failure. The material of this paper is organized as follows: Section 2 proposes a clear and consistent definition of robust design, and carefully disentangles it from reliability and resilient design. Section 3 describes different failure analysis methods and our logic to select functional model to study the failure behavior of the system. Section 4 illustrates how to represent a design applying a functional model. Section 5 proposes a cost-risk function to evaluate cost of a design and it's robustness against failure. Section 6 presents generating different designs and search algorithm. In section 7, we apply the proposed method to find the robust design for an aerospace monopropellant propulsion system. Section 8 is the result and finally, section 9 provides conclusion and summary.

II. ROBUST DESIGN

Failure analysis is the first step to design a complex engineered system and the strategies to make it a robust system must be designed into the system from the beginning, because as we go forward in the design process making changes is more complicated and expensive [1].

Design strategies used for advancing reliability are implemented for the purpose of advancing robustness in a system; however, there is meaningful difference between these concepts. Reliability is the ability of a system to perform nominally for a specific period of time. In fact, reliability is the probability of success or availability of a component/system for a specified period of time [2]. Reliability concentrates more on "why and how" components or systems may fail or have failed. 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 main issue of these techniques is the requirement of the component-level detail of the system from the beginning of the design process.

Robustness is where the system's performance is minimum sensitive to internal and external uncertainties that can cause failure [4-6]. The ability to overcome such uncertainties should be embedded into the system from the beginning. Uncertainty is the lack of ability to determine something precisely [7, 8]. Uncertainty management is an important concept in the design process of complex engineered systems. Reducing uncertainty when designing a system results in more efficient system with less cost.

Resilience can be defined as a system's ability to recover from a failure. Recovery from a failure is an alternative to reducing uncertainty. There are different ways to recover from uncertain events, including flexibility, Monitoring and Automated Contingency Management (ACM).

Flexibility is the ability of a system to respond to changes in initial requirements and objectives, after it starts operating, in a timely and cost-efficient way [9]. Keshavarzi et al, developed a strategy to apply flexibility in designing a complex engineered system to cope with epistemic uncertainty during the system operation lifetime [10, 11]. An ACM system adapts automatically and allows some degradation in the system performance when failure occurs with the goal of still achieving the mission [12-15]. For example, taking photos with less resolution is applying ACM when the mission dictates possessing an image. 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 the strategies to quantify those metrics [16]. To improve reliability, robustness or resiliency of a design, failure analysis is a necessary step. In most existing failure analysis methods, system designers need a precise model of system components to be able to study complex behavior of the system. However, in early phase of design process, specific set of components have not yet been selected and such detailed models of the complete system is not available. Because of this lack of methods to study failure behavior in the early design phase, the idea of representing the complex system by only its intended functionality is proposed as part of our methodology. The following section discusses failure analysis using functional models.

III. FAILURE ANALYSIS

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 a system, implementing a precise failure analysis in early design stage is vital [19]. There are different types of failure analysis techniques for complex systems [20-27], like probabilistic methods [28, 29], or reliability techniques [30, 31], or approaches based on observations analysis [32]. Studies have shown that the early design stage is the best time 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 to meet system requirements. The goal of developing a functional model is to describe the system behavior and determine vulnerable parts of the design, resulting in potential system improvement, particularly helpful in the early design stage when the detailed component model of the system is not available.

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 potential failure states for each function [34, 35]. Grantham et al., estimated the failure likelihood for each functional in the system. Lough et al., classified functions to high-risk to low-risk based on the consequence of failures [36, 37]. The idea of providing function-based failure analysis provided some improvements in the design of complex engineered systems; however, these methods limit the designers of the system to considering only one single-fault impact analysis at a time.

To overcome this restriction and to enable the designer to consider multiple function failures effect, the Function Failure Identification and Propagation (FFIP) method was presented [38-47]. The FFIP method identifies failure propagation paths by defining states of function health. This approach uses a separate behavioral model simulation to show failure propagation paths and failure effects. 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 IBFM tool (Inherent Behavior of Functional Models), to simulate failure scenarios applying functions, modes, flows between the functions and conditions for transition between the functions. With this tool, designers can create a functional model of the system in the early design stage and simulate the failure propagation paths for the system without developing a separate behavioral model [50].

In our presented framework, we apply the IBFM tool to simulate the failure scenarios for different design topologies. The following section describes the method to develop a functional models to represent a system.

IV. CONSEPTUAL PHASE

Fig. 1. The first step in the 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, because Python is free access and provides other tools like NetworkX which can be utilized to represent the functional model graphically. Graphs provide the designers the ability to represent functional models with complicated structure [51].

Fig. 2. 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. Fig. 2 provides an example of a graph

representation of a functional model consisting of a single internal function and three functions.

Fig. 3. In this paper, 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 our approach effort and rate variables take qualitative values, such as Zero, Low, Nominal, and High.

Fig. 4. 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. In the IBFM approach, all modes and conditions are qualitative, rather than quantitative. 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, and conditions are defined to construct a functional model to be fed to the IBFM tool to simulate all system failure scenarios.

Fig. 5. Our IBFM tool can be applied to simulate failure scenarios for a functional model. The IBFM tool is hosted on GitHub at https://github.com/DesignEngrLab/IBFM. The repository contains the module ibfm.py, and a user guide.

The simulation of each fault scenario begins at the nominal 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.

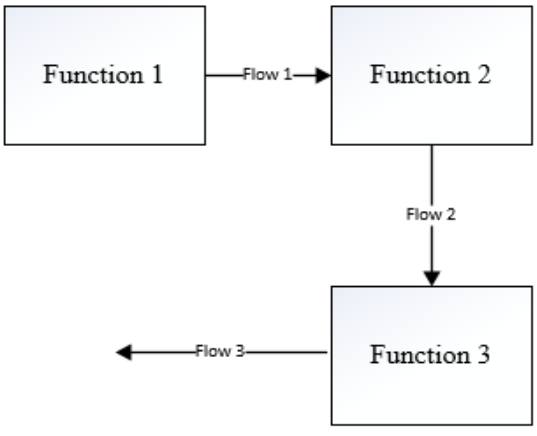

Fig. 2. Graphical representation of a functional model

V. ROBUST VALIDATION

In this part, a cost-risk model is proposed to evaluate the robustness of different designs for a complex engineered system. The idea of this cost-risk model is rooted in the risk-based utility theory [52]. This model studies the tradeoffs between cost of designing a system that is resistant against a failure, versus the cost of designing the system without robustness 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 Equation (1).

$$\text{Min } (C_D + C_O) + C_M + \sum_{i=1}^{N} C_{R,i} P_{R,i} (1 - P_{M,i}) \quad (1)$$

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 make the design robust. C_M, represents the cost of changing the design to make it robust to a particular failure. In the next section, we describe four strategies to change a functional model to represent different designs for a system. 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 failure *occurrence*, and it's probability of being *mitigated*. In Equation (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 (in units of dollars). Secondly, P_R is the probability of having a specific undesired end state or failure behavior and is quantified using the results of failure 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 that a design resist against 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 failure simulation result by adding the mitigation action to the functional model and rerunning the simulation. N is the number of system undesired end states or failure behavior that the system is designed to resist. Except for the nominal scenario when all functions perform nominally, other end states are failure scenarios and undesired. However, an undesired end state does not imply that a failure 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 or failure of the system; however, an engine fire is both an undesirable state and a safety hazard.

Applying the cost-risk model, the designer can determine the tradeoff between robustness and cost of a 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 cost (i.e., Equation (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 [53]. Termination Conditions for the proposed framework can be defined as:

- An upper limit on the number of evaluations (designs) is reached.
- An upper limit on the time 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 gets selected, otherwise a new design is generated and evaluated.

VI. GENERATE ALTERNATIVE DESIGN

As noted in previous part, 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/modify a functional model to produce other feasible designs in the search process. The challenge is 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. We propose four 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:

A. 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. 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 Equation (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).

B. Conceptual Order

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 failure avoidance; i.e., arranging the functions is such a way that failure propagation path is changed and thus the probability of a risk is changed.

C. Utilizing Wasted Flows

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 failure 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.

D. Combining/Splitting Functions

Two or more functions can be combined (or split) to make a new design and potentially improve the failure 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 failure simulation. Combining functions may lead to elimination of a failure path (thus reducing P_M), or may make the new combined function more likely to fail (thus increasing P_M).

We can generate a large space of alternative models by using the functions, however a small portion of the space can be technically feasible. For instance, by having every possible order of functions from initial mono propellant propulsion system model, we can generate a large space of designs, however having thrust function before heating gas is meaningless. Therefore, we narrow down the design space to small number of models that could be considered for the system. In the following section, we apply the proposed approach in early phase of designing a monopropellant propulsion system. The goal is to design the system to be

robust to events that can cause failure while managing the cost.

VII. 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 they make the engine lighter, less expensive, and more reliable. In this case study, the monopropellant is hydrogen peroxide (H_2O_2).

It is important when designing a monopropellant propulsion system to consider environmental condition in space. 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 spacecraft 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 spacecraft. Fig. 3, presents the general idea for a monopropellant propulsion system design.

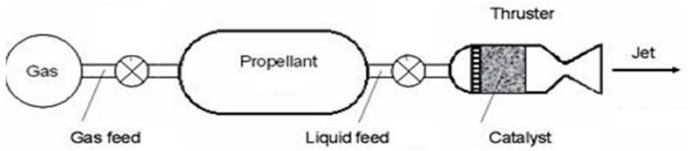

Fig. 3. General design for a monopropellant propulsion system

A. *Failure Behavior*

Assume a spacecraft with monopropellant propulsion engine has a mission to travel to a planet in outer space and from external orbit to the middle orbit and then to the lower orbit and take some images and get back to earth. Fig. 4, shows different scenarios. When the thrust is commanded, if all functions are *nominal*, the system performs as expected. In Fig. 4, the green dot shows the spacecraft accomplished the mission. However, there are scenarios in which something can go wrong with one or more functions and the result is not as expected. Blue, yellow and red dot in Fig.4, represents the scenarios that spacecraft ended up with an undesired situation. This would cause the engine to provide too much (yellow dot), too little (blue dot) or no thrust (red dot). In practice, it means the aerospace system passes the commanded orbit, or does not reach it. In rare catastrophic failures, the system might explode (i.e. loss of system). Therefore, the final behavior of failure analysis for a monopropellant propulsion system is classified to 5 groups:

- Mission Accomplished (Desired)
- Too Much Thrust (Undesired)
- Too Little Thrust (Undesired)
- No Thrust (Undesired)
- Loss of the System (Undesired)

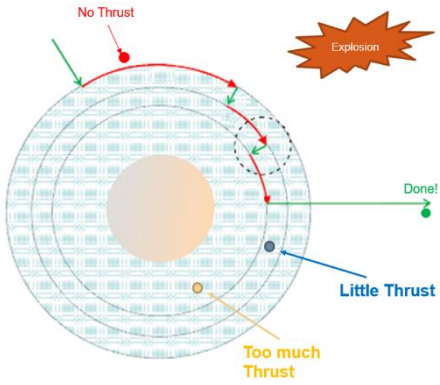

Fig. 4. End states of failure analysis for a monopropellant propulsion system

The first step in our proposed approach is developing a functional model. This includes defining functions, and flows, failure modes and conditions from transitioning from one state to another state.

Fig. 5, shows the graphical representation of our developed model for the monopropellant propulsion system. The boxes illustrate the functions required to produce thrust to change the velocity of the spacecraft when commanded. The blue boxed are the functions implemented by control software. The flows are represented by arrows; black arrows represent material flow between two function, the material can be solid, liquid or gas. The dot blue arrows illustrate flowing information or signal between two functions. Dashed black arrow is energy. The definition of the system health states (including failure and non-failure modes) and conditions are not visualized in Fig. 5, to avoid too much information in one graph. The following session describes how to define the design in Python and simulate failure behavior.

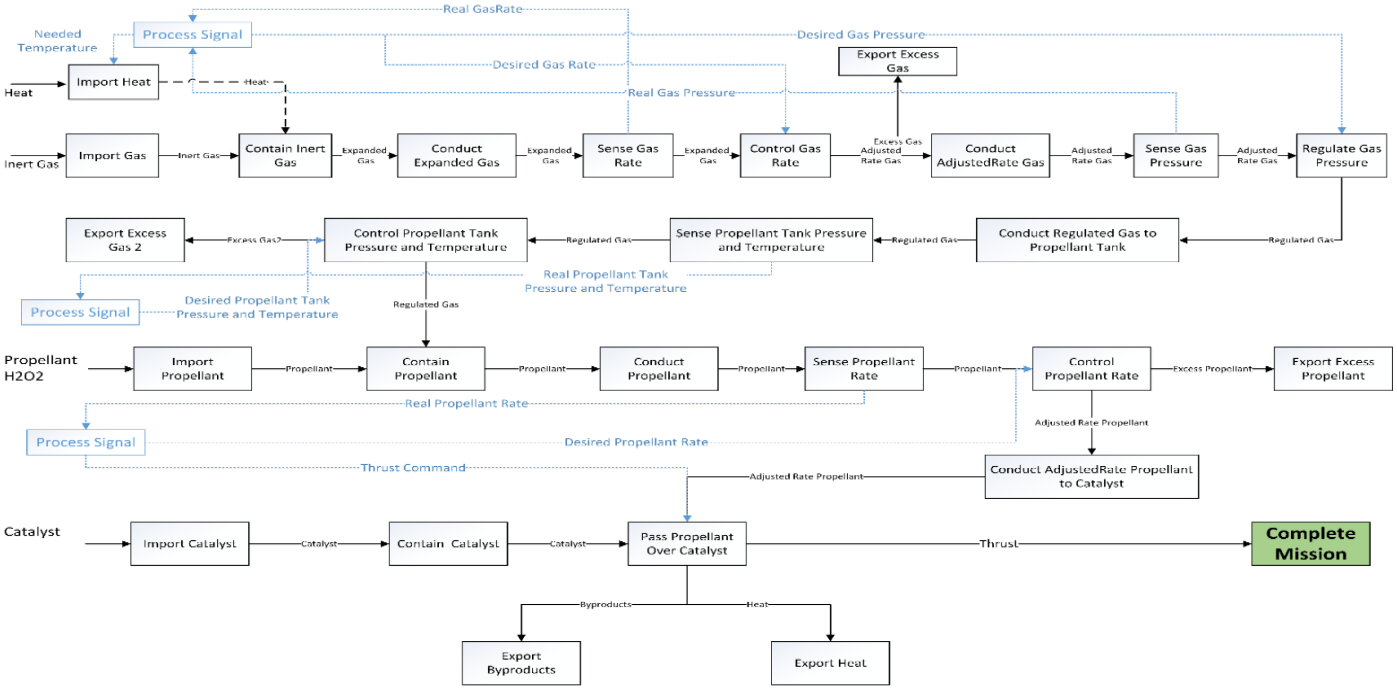

Fig. 5. Developed functional model for monopropellant system

A. *Functions Definition*

The monopropellant propulsion system model in Fig. 5, contains 29 functions and 129 modes. One example of our function and modes definition for the monopropellant propulsion system in Python is as follow:

```
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 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 modes. *Failed modes* are the ways, in which the system might fail. A single mode in each function definition is followed by the keyword Nominal, which by default assigns that mode to be the initial mode of the function at the beginning of simulation.

B. *Flows Definition*

Flows are defined in a single line. Our strategy to define a flow is to mention the keyword *flow*, then the type of flow, followed by the name of the category of the flow which can be Material, Energy, or Signal. All flows are derived from one of mentioned three categories of flows.

```
flow Heat Energy
```

C. *Modes Definition*

Modes definition is 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 being assigned to. The expression on the right of the assignment operator determines the state of the flow variable. 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.

D. Conditions Definition

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.

We simulate all failure scenarios using our developed IBFM tool. We tabulate the unique scenarios terminating with particular undesired end states. The final behavior of the system as shown in Fig. 4, is categorized into different main end states. The designers of the system decide 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 when Needed
- Loss of the System

E. Alternative System Designs

The baseline design, shown in Fig. 3, represents a functional model developed for a monopropellant system in the early design phase. In this design, the desired final behavior is the mission accomplishment, in which all gas, propellant, and catalyst functions operate nominally. Any improvements in the functional model influence the final behavior of the system. In other words, the baseline model is not designed to be robust. Therefore, different system topologies with focusing on improvement the robustness of the system can be investigated. Improvement strategies include applying different levels of redundancy, re-configurability or integrated health management sensors in the system design. For each design, the failure scenarios can be classified into five end states. Fig. 6, represents the *Pie Chart* for a candidate design for monopropellant propulsion system.

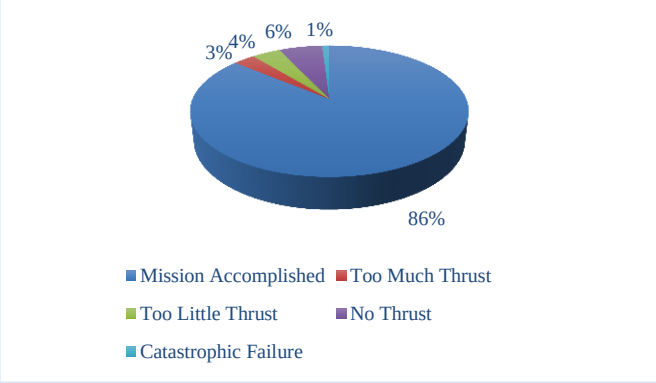

Fig. 6. Classification of failure scenarios for basic monopropellant design

In this case study, we simulate the failure behavior of the initial design and compare it to six alternative functional models representing different designs for the monopropellant propulsion system. In each alternative design, changes are applied in the functional model to improve the robustness of the design against some particular failures.

The first modification of the baseline design includes a redundant gas rate sensing function. In this design, if no signal is coming from the primary sensing function, a secondary sensing function can supply the required information. The second modification of the initial design is an identical system with redundant gas pressure sensing. The third modification of the initial design combines the adjusting pressure and rate into one function. The fourth design uses output heat to expand inert gas (by contrast, in the baseline, the heat is exported from the system and disappears into space). The fifth design incorporates the redundant sensing for propellant. The sixth design applies the output thrust heat to preheat the propellant. In this way, the propellant plays the role of a cooling system. The following section presents the result of failure simulation for all six alternative designs and evaluate them based on our proposed robust score function.

VIII. RESULT

Seven candidate designs for the monopropellant propulsion system are examined. In each design, the functional model has been modified to be more robust to a particular failure or undesired end state. Table 1 shows all the designs generated by the functional model modifications.

TABLE 1. Generated designs for monopropellant system

Basic Design	The original unaltered system model
Design 1	Redundant gas rate sensing added
Design 2	Redundant gas pressure sensing 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 pressure sensing
Design 6	Use output thrust heat to expand inert gas and preheat propellant

Failure behavior has been simulated for all six alternative designed in Table 1. 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 injecting three simultaneous faults. In this study, we investigate injecting up to 3 faults (all possible three failures) in each one of the functional models. Fig. 7, displays the classified simulated scenarios for the seed and the six design topologies is shown in Fig. 7. In this histogram, the number of successful and failed scenarios simulated for each design is demonstrated.

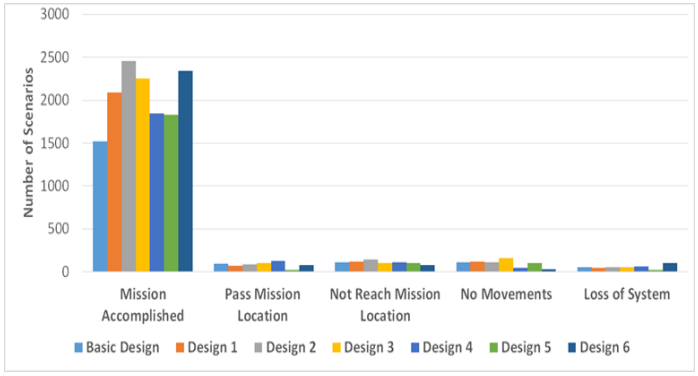

Fig. 7. Failure simulation results for candidate monopropellant designs

The probability of risk in each design is applied to the cost-risk model of Equation (1). The mitigation cost C_M is the cost of changing the basic design to make it more robust to a particular failure or undesired end state.

The operation cost C_O for all candidate designs is assumed to remain the same because on-ground systems cost is the main operation cost for a monopropellant propulsion system. It's assumed that regardless of how the system is performing, there is an on-ground system to monitor, control and run the spacecraft. 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.

TABLE 2. Robust evaluation for monopropellant system candidate designs (\$million)

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 $C_R P_R(1-P_M)$	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
Robust Score	Rank 7	Rank 6	Rank 4	Rank 2	Rank 3	Rank 5	Rank 1

The probability of the mitigation, P_M, is the probability that the mitigated part does not work when needed. These probabilities are quantified based on the IBFM simulation and cost numbers are estimated based on the space system cost models and data provided by Miller et al. [54]. The total cost reflects the tradeoff between designing a robust system and the cost of doing so. The lower the total cost, the higher the rank of the design would be.

Table 2 represents the results of robust evaluation for developed designs for monopropellant propulsion system. It tabulates all the elements of Equation (1). Since there were only seven designs to consider, an exhaustive optimization algorithm was used; however, an evolutionary or other stochastic algorithm could be used to search larger design spaces.

As shown the table, Design 6 has the optimal trade-off between robustness and cost. In this design the propellant acts as a cooling system. The heat produced by propulsion is used to preheat the propellant and expand the inert gas. Utilizing this source of energy helps the successful combustion process. 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. 543

IX. CONCLUSION

This paper proposed a framework to design a complex engineered system in early design phase with the ability to resist against failure. We discussed the lack of current methods to address failure analysis and robustness of a system in the early design phase. Current design methods require a detailed component model of a system to be able to simulate and study the failure behavior of the system. However, such methods are not applicable in early design phase when the set of components is not selected yet and design is in the conceptual step. Ideally, the strategies to make a robust design should be implemented in early design phase, because as we go forward in the design process, making changes is more expensive and challenging. In our proposed method, we showed how to represent a complex engineered system with a functional model in early design phase. We illustrated how to define functions, modes, flows and conditions to simulate the failure behavior of the system and how to generate different designs for a system using a functional model. Finally, our developed robust score function provides the ability to evaluate the trade-off between cost and robustness of a particular design and search for the optimal one among candidate designs.

We applied the proposed method in early phase of designing an aerospace monopropellant propulsion system. The results show that the presented method is a practical design framework to conduct failure analysis and develop robust complex engineered systems in the early design phase, when the complete knowledge of the system components and specifics is not available. A future research is needed to investigate the proposed approach in a design problem where the design space is large and there is a large number of feasible candidate designs. Graph grammars could be implemented to help generate a large number of design alternatives.

ACKNOWLEDGMENT

This research was funded by NASA Grant and Cooperative Agreement NNX15AQ90G. We really appreciate their support.

REFERENCES

[1] Ullman, David G. "The mechanical design process." (2003).

[2] O'Connor, Patrick, and Andre Kleyner. "Practical reliability engineering." *John Wiley & Sons*, 2012.

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

[4] Phadke, Madhan Shridhar. "Quality engineering using robust design." *Prentice Hall PTR*, 1995.

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

[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] Hund, Edelgard, D. Luc Massart, and Johanna Smeyers-Verbeke. "Operational definitions of uncertainty." *Trac trends in analytical chemistry* 20.8 (2001): 394-406. 544

[8] Milliken, Frances J. "Three types of perceived uncertainty about the environment: State, effect, and response uncertainty." *Academy of Management review* 12.1 (1987): 133-143.

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

[10] Keshavarzi, Elham, Matthew McIntire, and Christopher Hoyle. "A dynamic design approach using the Kalman filter for uncertainty management." AI EDAM 31.2 (2017): 161-172.

[11] 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.

[12] Ge, Jianhua, M. J. Roemer, and George Vachtsevanos. "An automated contingency management simulation environment for integrated health management and control." *Aerospace Conference, 2004. Proceedings. 2004 IEEE*. Vol. 6. IEEE, 2004.

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

[14] Fiksel, Joseph. "Designing resilient, sustainable systems." *Environmental science & technology* 37.23 (2003): 5330-5339.

[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] Blanchard, Benjamin S., Wolter J. Fabrycky, and Walter J. Fabrycky. "Systems engineering and analysis." Vol. 4. *Englewood Cliffs, NJ: Prentice Hall*, 1990.

[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] Walsh, Hannah S., Andy Dong, and Irem Y. Tumer. "The structure of vulnerable nodes in behavioral network models of complex engineered systems." ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. American Society of Mechanical Engineers, 2017.

[20] Stone, Robert B., Irem Y. Tumer, and Michael Van Wie. "The function-failure design method." Journal of Mechanical Design 127.3 (2005): 397-407.

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

[22] 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.

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

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

[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] Quoilin, Sylvain, et al. "ThermoCycle: A Modelica library for the simulation of thermodynamic systems." *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.

[53] Hulse, Daniel, et al. "Towards a Distributed Multiagent Learning-Based Design Optimization Method." ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. American Society of Mechanical Engineers, 2017.

[54] Miller, David W., Col John Keesee, and Mr Cyrus Jilla. "Space systems cost modeling." (2003).