Proceedings of the ASME 2018 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2018
August 26-29, 2018, Quebec City, Quebec, Canada

DRAFT: DETC2018-85318

OPTIMIZING FUNCTION-BASED FAULT PROPAGATION MODEL RESILIENCE USING EXPECTED COST SCORING

Daniel Hulse
Graduate Research Assistant
School of Mechanical, Industrial
and Manufacturing Engineering
Oregon State University
Corvallis, Oregon 97330

Christopher Hoyle *
Associate Professor
School of Mechanical, Industrial
and Manufacturing Engineering
Oregon State University
Corvallis, Oregon 97330

Kai Goebel
Tech Area Lead
Discovery and Systems Health
Intelligent Systems Division
NASA Ames Research Center
Moffett Field, California 94035

Irem Y. Tumer
Professor
School of Mechanical, Industrial
and Manufacturing Engineering
Oregon State University
Corvallis, Oregon 97330

ABSTRACT

Complex engineered systems are often associated with risk due to high failure consequences, high complexity, and large investments. As a result, it is desirable for complex engineered systems to be resilient such that they can avoid or quickly recover from faults. Ideally, this should be done at the early design stage where designers are most able to explore a large space of concepts. Previous work has shown that functional models can be used to predict fault behavior and motivate design work, however little has been done to formally optimize a design based on these predictions, partially because the effects of these models have not been quantified into an objective function to optimize. This work introduces a scoring function which integrates with a fault scenario-based simulation to allow for the optimization of functional model resilience. This scoring function enables the designer to consider trade offs between the design costs, operating costs, and expected fault response of a given design, and may be parameterized in terms of designer-specified design changes to suit optimization. This framework is adapted and applied to the optimization of controlling functions which recover flows in a monopropellant orbiter. In this case study an evolutionary algorithm is found to find the optimal logic for these functions, showing an improvement over a typical a-priori guess by exploring a large range of solutions, demonstrating the value of the approach.

1 Introduction

Complex engineered systems such as nuclear power plants, aerospace vehicles, and oil rigs are often associated with large investments and significant failure consequences. High-profile failures in these systems, such as the Chernobyl disaster, Challenger catastrophe, and Deepwater Horizon oil spill have caused deaths, environmental damage, and billions of dollars of economic loss. As a result, it is important to design the systems

*Address all correspondence to this author.

in such a way that minimizes risk and responds well to adverse circumstances such that performance, cost, and safety are maintained or recovered.

This goal presents a significant challenge, due to the inherent complexity of these systems. Indeed, complex engineered systems comprise many components, each with many possible interactions. Despite each component having relatively low failure probability, there is often only a poor understanding of compound failure risk. For example, in the aftermath of the Challenger catastrophe it was found that engineers' estimates of overall failure probability differed by orders of magnitude [1, see Appendix F]. While failures in complex systems are often attributed to poor management and operations, they can often be traced to design flaws.

To address this challenge, risk and failure approaches have been introduced which help designers reason about failures and their impact on the design [2], including failure modes and effects analysis [3], fault tree analysis [4], and model-based approaches [5] [6]. However, these approaches are generally not suitable for early design–the focus of this work–since they require detailed knowledge of the system. As a result, a variety of failure approaches have been formulated based on the functional model of the system, since these are most available at the early design stage.

1.1 Prior Work

Prior work introduced the function-failure design method (FFDM) to predict likely failure modes due to the loss of functions using past data to show which functions require more design attention [7]. This was extended in the risk in early design method (RED) using likelihood and consequence estimates to better inform designers [8] [9]. The function failure identification and propagation method (FFIP) in turn informed the analysis by constructing a behavioral model to take into account the function interactions, dynamics, and joint fault scenarios [6] [10] [11] [12]. Approaches have additionally been presented which associate the functional model as a fault tree [13] [14]. Other methods have been created to focus on the propagation of failures through a functional model [15]. Inherent Behavior of Functional Models (IBFM), which is used in this paper, provided a method to automate the creation of a state-based behavior model from the functional model itself [16].

Attempts have in turn been made to show how to generate, improve, or change the design based on these function-based failure frameworks. Initially in developing these frameworks, the resulting information was just used to show designers where attention should be paid in making design choices [7]. However, concepts have subsequently been presented to use graph grammars to change the structure of the model, and/or use a cost-risk analysis scoring function to compare between design alternatives [17]. Additionally, a concept has been presented for designing the operational decision-making in the model to determine when to, for example, route degraded flows to sacrificial subsystems [18]. While these approaches show many of the design changes that can be made in the functional model, and can be used to compare between design alternatives, no attempts have yet been made to use this knowledge to formally optimize a given design based on this knowledge.

1.2 Aims and Contributions

The ultimate aim of this research is to create a comprehensive optimization-based framework to approach resilient design in the early design stage. Resilience in general is accomplished through a variety of possible design changes and operational decision-making, including: redundancy and health management, changing the order of functions, using unused flows as inputs, combining functions (as identified in [17]), changing the operational decision-making of the functions in the model (as identified in [18]), selecting functions which behave differently, and generating totally different or novel functional model solution concepts. Towards that goal, this paper presents a scoring function which integrates with set of fault-event simulations to calculate the expected cost of a design considering every fault scenario, and applies it to a case study.

The next sections present background in function modeling and fault simulation, discuss context and definitions of resilience, introduce the scoring function, and demonstrate the approach by applying it to the optimization of a monopropellant orbiter.

2 Background

2.1 Functional Modeling

Functional modelling is a way of representing the concept of purpose in system which has been described as a language for conceptual design intention, a bridge between high-level decision-making and implementation [19], and a "blueprint" for the future system which is agnostic of any particular form [20]. While a variety of conventions have been presented, in general function modelling represents a system as a set of functions which act on flows to accomplish a given task [19]. This specific representation of functionality is one of many system representations of products, but is uniquely useful for its lack of ambiguity and ability to be reused and transformed to simulate behavior [21]. It has subsequently been standardized as a part of the systems design process [22] [23].

The representation used in this paper follows the convention described in [20]. Flows can represent any sort of material, energy, or signal which passes through the system, while functions represent any operations that happen to the flows on their way through the system which are necessary for accomplishing the overall purpose of the model, which are stated as verb-noun pairs. A standard way to develop a functional model is to first create what is called a black-box model of the system which states the overall function with all of the known flows going in and out of the black box. The designer then creates function chains by "following the flow," identifying and sequencing the

operations that must be done to the input flow to transform the input flow into the output flow. Finally, the function chains are aggregated and connected as needed to create the overall functional model.

2.1.1 IBFM simulation and model definition

This paper uses IBFM to simulate the fault behavior of a system given its functional model. IBFM was developed for this purpose, with a focus on simulating how the effects of a list of faults propagate through a functional model [16]. This tool acts by constructing a behavioral model of the system from the functional model by associating behaviors with functions and states with flows. The resulting behavioral model can be used to generate a list of failure scenarios based on each individual or combination of fault events which shows the end-state of the model given that event happens. IBFM can be run considering any number of joint-faults, and runs quickly because it executes the behavioral model as a state-machine, rather than a time-based dynamic model. Since the framework presented in this paper is designed to integrate with this tool and interfaces directly with several IBFM definitions, important related terminology is listed below:

Fault scenario: A particular instance of the IBFM model for one specific fault event. IBFM generates a list of fault scenarios depending on the number of joint faults considered in a run.

End-State: The result of a fault event being run to completion.

Mode: A state of a function which is associated with a particular behavior. Modes are either fault modes, implying an undesired behavior, degradation, or loss of function, or nominal modes, implying that the function is operating as desired. In this paper, we differentiate two types of fault modes: *initiating fault modes*, which are the initial faults which generate a scenario, and *conditional fault modes*, which happen as a result of other faults.

Condition: A rule that specifies a function's change in mode and resulting behavior as a function of incoming flow health state.

Behavior: A property of the function mode which determines how an input health state of flows determines the output health state of flows.

Health State: A property given to flow rate and effort variables that may take the value zero, low, nominal, high, or highest. It represents the degradation in the quality of the flow.

Rate Variable: A property given to flows to represent the rate aspects of a flow analogous to throughput or velocity.

Effort Variable: A property given to flows to represent the effort aspects of a flow analogous to force or pressure.

2.2 Resilience

A number of definitions of resilience have been introduced across a variety of fields, including ecology [24] [25], psychology [26], economics [27] [28], sociology [29], network science [30] [31], and management [32] [33]. Resilience is broadly defined as the ability of a system to prepare for, absorb, recover from, and adapt to failure events, and resilience strategies typically focus on the temporal adaptation to failures, as opposed to risk management, which is more focused on preventing the failure events [34]. However, definitions and metrics for measuring resilience vary across and within fields, with both qualitative and quantitative metrics [35]. Key dichotomies include:

Engineering resilience and ecological resilience: In engineering resilience, the performance and stability of the system state is recovered to the original system state while in ecological resilience the function of the system is recovered from a static failure state to a new dynamic state, potentially as a result of a change in components (e.g. similar species taking the role of a newly-extinct species) [25].

Deterministic and probabilistic measures: Probabilistic measures consider the uncertainty of disruptions or failure events while deterministic measures do not [35].

Dynamic and static measures: Dynamic measures take into account time-dependent behavior while static measures do not [35].

Of particular interest to this paper is how to use the concept of resilience to motivate design decision-making in engineering design. As with the broader fields of science, definitions and metrics of engineering resilience vary, however the time-based response to disruptive events is key [36]. While these metrics capture resilience as a metric, they are incomplete as design metrics because they do not incorporate the trade-offs between resilience and other cost and performance considerations. As a result, previous work has created cost functions based on decision theory which quantify the cost of reacting to failure events in different ways [37] [17], allowing for informed design decision-making and optimization considering resilience.

3 Method: Optimizing Resilience with a Scenario-based Scoring Function

This paper introduces a framework for optimizing the resilience of functional models in the early design stage by integrating a fault scenario-based simulation with a cost function. This framework is shown in Figure 1: the designer defines an initial functional model and creates a behavioral model by associating conditions, behaviors, and modes with the various functions as presented in [16] using IBFM. The designer then associates costs and probabilities with the various modes, functions, and flow states, and defines the changes to be explored to create a parameterized cost function. This cost function is then optimized iteratively with an algorithm fitting to the problem by running the fault simulation, scoring the design, and changing the parameters until an optimal design is found. This enables the designer to explore a large space of design alternatives in a systematic, automated way without a tedious investigation of every model

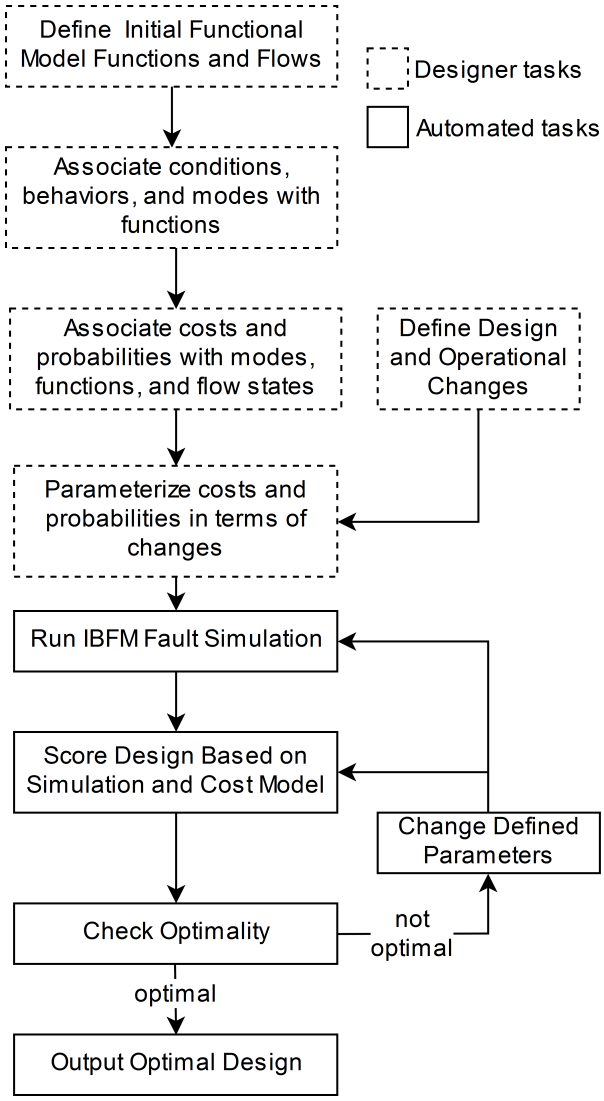

FIGURE 1. Framework enabled by integrating cost-based scoring and fault simulation. The designer sets up a parameterized design problem which is then solved by an optimization algorithm.

variant.

To enable this framework, this paper introduces a scoring function which incorporates the trade-offs between a system's design costs, operating costs, and failure behavior. Of particular interest is this function's approach to model failure behavior, which is built on IBFM's conception of a fault scenario–a set of faults which yield an end-state. The basic form of this function is a sum of the design costs C_D, operating costs C_O, and fault event costs C_E as shown in the following equation:

$$C = C_D + C_O + C_E \tag{1}$$

This function is quite similar in form to that devised in [17] in that it considers trade-offs between design and operation costs, and the response of the system to various failure events. However, the main difference is the incorporation of mitigating factors–while the scoring function devised in [17] considered the cost of mitigating factors which reduced the probability of end-states, the function introduced here is generally applicable to all sorts of mitigating actions and design changes, and shows more precisely how to translate the results of IBFM simulations into costs and probabilities, as will be shown in the following sections.

3.1 Design Cost

Design cost models may be different, depending on the design problem. This is because, in general, design costs come from a variety of sources, including research and development, required materials, manufacturing, and integration. This work considers that the design costs of a given functional model can be mapped back to the individual costs of the functions. As is the approach with risk and failure modes [7], these costs can in turn be estimated based on an organization's past costs for those functions. The resulting equation for design cost C_D is then:

$$C_D = \sum_{n \in N} C_n \tag{2}$$

where C_n is the cost of a given function n in the set of all function instances in the model N.

3.2 Operation Cost

As with design costs, operating costs must be estimated based on an expectation of the realized system which may result from past performance or a designer-created parametric model specific to the problem-situation. In general, while the operating costs are difficult to capture without an understanding of the individual problem, it should be noted that, in general, they relate to the individual flows going in and out of the black-box form of the model. Flows going into the model can result in costs (such as those having to do with raw materials and energy) and revenues (such as those that take in waste material), as do flows going out of the model, with costs potentially resulting from waste streams and revenues resulting from the useful goods created by the model. In general, the operational cost C_O then follows the form:

$$C_O = \sum_{l \in L} C_l - R_l \tag{3}$$

where C_l and R_l are the respective costs and revenues associated with the flow l in the set of inflows L entering and leaving the black box model. It should be noted that these C_l and R_l terms do not need to stand for explicit costs and revenue in dollars generated, but can also stand for the utility gained and lost by the organization. That is, if the organization has some purpose (e.g.

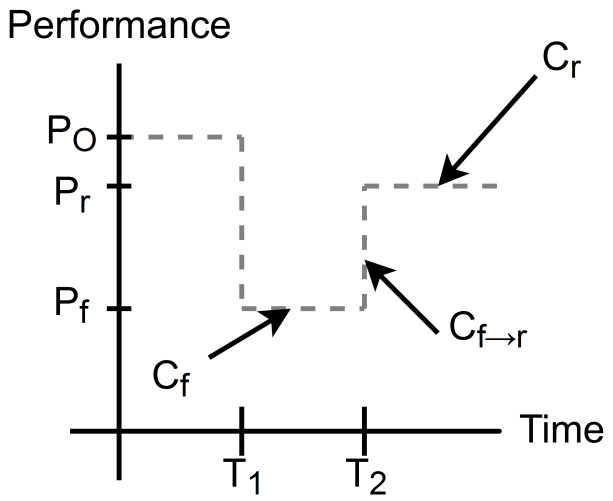

FIGURE 2. Costs associated with a failure event in a resilient system.

generate science, etc) that does not explicitly lead to more or less revenue, the utility gained or lost by the organization by pursuing this purpose can be quantified the same way as revenue and cost would.

3.3 Fault Scenario Cost

Key to representing resilience in a system is the time-based response to a large number of disruptions or threat vectors [38]. This paper's representation of these disruptions is based around IBFM's ability to simulate large numbers of fault events, which are created by initiating a set of faults which propagate through the model until an end-state is reached. These events have costs, which are determined from the system's response to the events, and probabilities, which are determined by the probabilities of the initiating faults. The cost incurred fault events is the expected cost of those fault events–the cost of each event is weighted by its probability. As a result, the fault event cost C_E follows the general form:

$$C_E = \sum_{e \in E} P_e * C_e \tag{4}$$

where $_e$ and C_e are the respective probabilities and costs of fault event e in the set of considered events E. The probability of a given scenario is simply the product of the probability of the specific combination of originating faults occurring multiplied by the probability that the rest of the system remains nominal as follows:

$$P_e = \prod_{m \in e} P_m * \prod_{n \notin e} (1 - \sum_{m \in n} P_m) \tag{5}$$

where P_e is the probability of a given fault event e, P_m is the probability of a given initiating fault mode m in scenario s occurring, and n is a function that does not have a fault in the scenario s.

To consider resilience in design, the system's response to a fault event is given three distinct costs which are associated with different stages in the system's failure and recovery, as shown in 2: the cost of the failure C_f, the cost of mitigating the failure $C_{f\rightarrow r}$, and the cost of deviations in the recovered performance C_r. It should be noted that this definition lines up with common representations of resilience, in which the system starts at a nominal stable condition, enters an unstable state due to a disruption, and then settles in a new recovered stable condition [39]. The resulting fault event cost follows the form:

$$C_e = C_f + C_{f\rightarrow r} + C_r \tag{6}$$

These cost definitions are additionally connected to IBFM's conception of a fault scenario in that each of the costs are derived from the end-states of IBFM running a simulation with specified faults in that:

C_f derives from the scenario end-state of the associated fault event e,
C_r derives from the scenario end-state of a new fault event simulation, with a certain set of modes repaired, and
$C_{f\rightarrow r}$ derives from the cost of repairing the modes present in the end-state of the associated fault event e not used as fault modes in the recovered fault event simulation.

Calculating these individual costs is discussed in the following sections.

3.3.1 Failure Cost

Failures result in costs because they degrade the important flows leading in and out of the system, resulting in higher costs, less revenue, or less utility. As was discussed in Section 3.2, these important flows must be identified by the designer with costs included. To determine the costs of specific failure events, specific costs must additionally be associated with the flow states present in the end-state of the fault event. These flow states are defined as the quality (zero, low, nominal, high, or highest) of a flow's rate and effort components. The general form of a specific matrix $\bar{c}_l$ for flow l is shown in Table 1. Note that these, as *specific* costs, are the cost per unit time until the failure is mitigated, as the total cost of a failure depends both on the severity of the failure on the flow state and the amount of time the flow is in the degraded state.

The cost of the failure part of the fault event can then be calculated using the state of the flows going in and out of the model and the time taken to mitigate the failure as follows:

$$C_f = t_f * \sum_{l \in L} \bar{c}_l[\vec{s}_{f,l}] \tag{8}$$

where C_f is the cost of the failure scenario end-state f that is the direct result of the simulation of fault event e, t_f is the time taken between the failure and the recovery, l is a given flow in the set

$$C = \sum_{n \in N} C_n + \sum_{l \in L} C_l - R_l + \sum_{e \in E} (\prod_{m \in e} P_m * \prod_{l \notin e} (1 - \sum_{m \in e} P_m) * (\max_{m \in f \cap \notin r} (t_m) * \sum_{l \in L} \bar{c}_l[\vec{s}_{f,l}] + \sum_{m \in f \cap \notin r} C_m + t_r * \sum_{l \in L} \bar{c}_l[\vec{s}_{r,l}])) \quad (7)$$

TABLE 1. Cost rate of an individual flow state for flow l based on combination of rate and effort health states.

	Zero	Low	Nominal	High	Highest
Zero	$\bar{c}_l[00]$	$\bar{c}_l[01]$	$\bar{c}_l[02]$	$\bar{c}_l[03]$	$\bar{c}_l[04]$
Low	$\bar{c}_l[10]$	$\bar{c}_l[11]$	$\bar{c}_l[12]$	$\bar{c}_l[13]$	$\bar{c}_l[14]$
Nominal	$\bar{c}_l[20]$	$\bar{c}_l[21]$	$\bar{c}_l[22]$	$\bar{c}_l[23]$	$\bar{c}_l[24]$
High	$\bar{c}_l[30]$	$\bar{c}_l[31]$	$\bar{c}_l[32]$	$\bar{c}_l[33]$	$\bar{c}_l[34]$
Highest	$\bar{c}_l[40]$	$\bar{c}_l[41]$	$\bar{c}_l[42]$	$\bar{c}_l[43]$	$\bar{c}_l[44]$

of ingoing and outgoing flows L, $\bar{c}_l$ is the specific cost matrix for that flow, and $\vec{s}_{f,l}$ is the end-state of that flow l in the given scenario f.

This time taken to recover the failure is the time necessary to repair the individual failure modes which are in the failure scenario but not in the recovered scenario. Considering that the repairs are done in parallel, this time can be calculated as the maximum of the times t_m needed to repair each failure mode m in the failure scenario end-state f but not in the recovered scenario r.

$$t_f = \max_{m \in f \cap \notin r} (t_m) \quad (9)$$

3.3.2 Mitigation Cost The cost of mitigation is considered a result of repairing the failure modes in the failed system scenario that are not present in the recovered system. This cost $C_{f \to r}$ is calculated as the sum of the cost C_m of recovering each mode m which is present in the failure scenario end-state f but not in the recovered scenario r, per the following equation:

$$C_{f \to r} = \sum_{m \in f \cap \notin r} C_m \quad (10)$$

3.3.3 Recovered System Cost Finally, the of the recovered system is the cost of the degraded flows still present in the end-state recovered system due to unrepaired or unrecoverable failure modes. This may be calculated similarly to the failure cost, by running a new fault scenario using the failure modes in the recovered state. This recovered cost C_r is a result of the time left in the recovered state (i.e., for the remaining life of the system) t_r and the specific costs $\bar{c}_l$ of the state $\vec{s}_{r,l}$ in the recovered end-state r of the flow l in the set of ingoing and outgoing flows L. This is shown in the equation:

$$C_{r,R} = t_r * \sum_{l \in L} \vec{c}_l[\bar{s}_{r,l}] \quad (11)$$

These costs are calculated over the rest of the life of the system. If the system is meant to operate for a long time, a discount factor should be applied based on the time value of money for the organization.

3.4 Summary

The previous sub-sections discussed how to calculate the costs of a system considering the design, operation costs, and fault scenario costs using the IBFM simulator, with special consideration of resilience–the ability to recover from failures.

Stated together, this cost function is shown in Equation 7, where:

C is the total cost
C_n is the cost associated with the design of a function
n is a function
N is the set of function instances in the model
C_l is the cost associated with an ingoing or outgoing flow
R_l is the revenue or utility associated with an ingoing or outgoing flow
l is a flow
L is the set of ingoing or outgoing flows
e is a fault event, a combination of fault modes
E is the set of considered fault events
m is a fault mode
P_m is the probability of a fault mode
f is the resulting fault scenario end-state of the fault event e
t_m is the time taken to repair a mode
r is the recovered scenario that is the result of repairing fault modes in f
$\bar{c}_l$ is the cost function of the flow based on its state
$\vec{s}_{f,l}$ is the state of the flow l in the scenario end-state f
C_m is the cost associated with repairing a fault mode
t_r is the time remaining between recovery and the end of life of the system
$\vec{s}_{r,l}$ is the state of the flow in the recovered scenario

4 Case Study: Optimization of Monopropellant System Controlling Functions

The following section shows an application of this framework to the Optimization of Controlling Functions in a Monopropellant orbiter, a system previously considered in [17]. Monopropellant propulsion systems are named as such because they do not require a separate oxidizer, and are commonly used in spacecraft. The propulsion system generates thrust when it receives

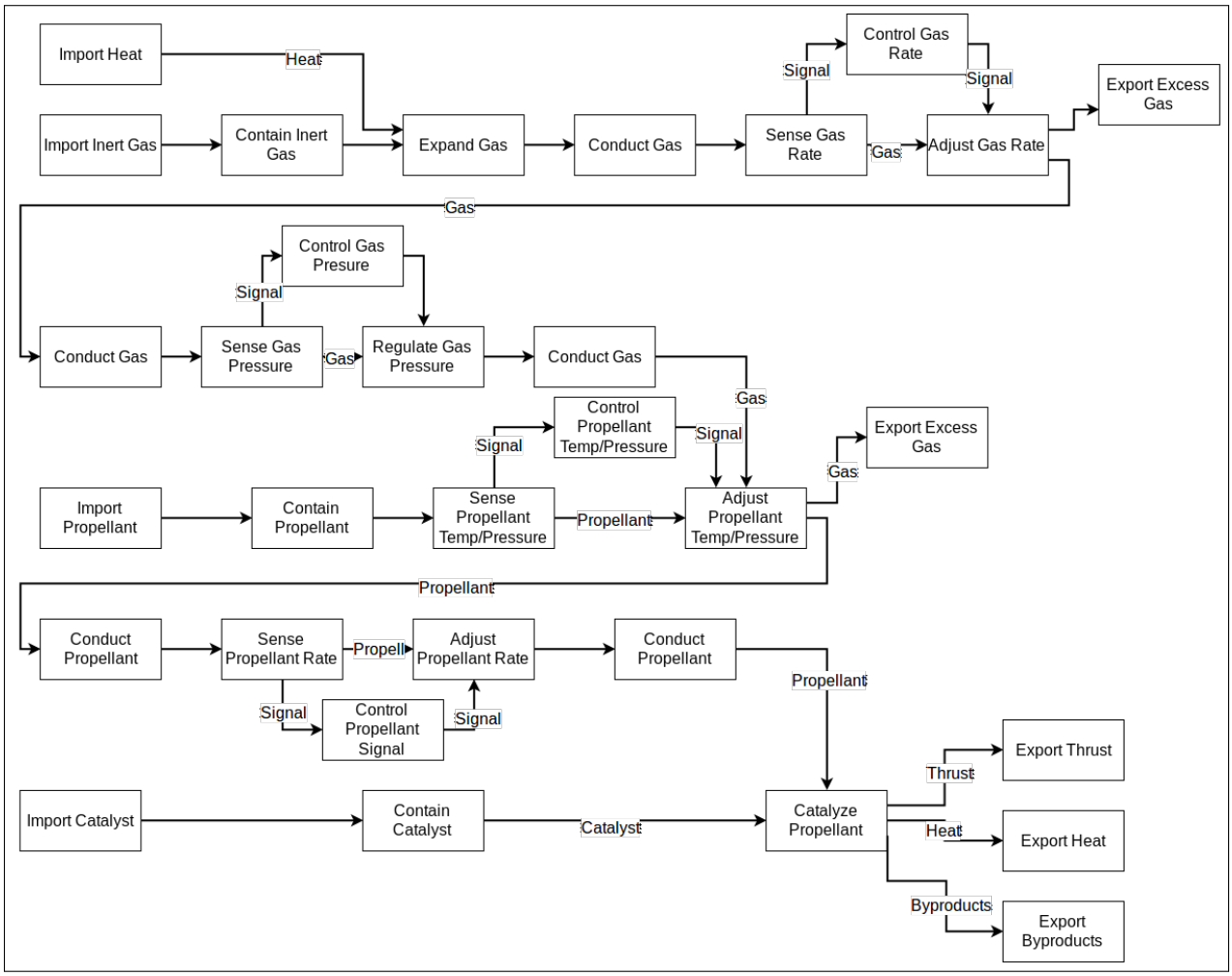

FIGURE 3. Functional Model of Monopropellant System with Controlling Functions

a command to do so. Then, heat is applied for a gas to expand, and the gas is regulated for appropriate temperature and pressure. The expanded gas then pushes and guides the monopropellant over the catalyst. When the monopropellant reaches the catalyst, thrust is generated. The functional model of this system is shown in Figure 4.

Controlling functions refer to the functions in the model which change the response of the system based on a signal indicating a change in flow. In this study they represent the design intent of the control systems of the regulating functions in case of a degradation or failure in the upstream flows. That is, they represent whether the system should be designed to recover a flow (which would compensate for the failure but increase initial design costs) or keep the flow state constant. In the model of the monopropellant propulsion system, these functions are *control gas rate*, *control gas pressure*, *control propellant temp/pressure*, and *control propellant rate*. In a realization of the system, these might be logic gates, control circuitry, or any system which takes actions based on an input. This is represented in IBFM as changes in conditions which cause the system to enter modes with different behavior.

```
mode 1 Operational EqualControl
mode 2 Operational IncreaseControl
mode 3 Operational DecreaseControl
condition 1 3 to 2 LowSignal
condition 1 2 to 3 HighSignal
condition 2 3 to 1 NominalSignal
```

FIGURE 4. Example controlling function conditions and modes.

To illustrate, in the function definition shown in Figure 4, the modes `EqualControl`, `IncreaseControl`, and `DecreaseControl` each refer to behaviors in which the con-

troller keeps the incoming flow state, increases the incoming flow state, and decreases the incoming flow state, respectively. Similarly, the conditions `LowSignal`, `HighSignal`, `NominalSignal` refer to a lower-than-nominal, higher-than-nominal, and nominal flow state, respectively. Finally, the conditional logic specifies which mode to enter based on the condition. For example, `1 3 to 2` means the function increases the flowstate by entering mode 2 (`IncreaseControl`) when it was previously in mode 1 (`EqualControl`) or 3 (`DecreaseControl`).

The trade-offs associated with this conditional logic is considered using the scoring function introduced in Section 3 by appropriately increasing the design costs from the related functions C_n when conditional logic is used for any mode except the `EqualControl` mode, and then increasing operational costs in each scenario in which that compensating mode is used.

4.1 Optimization Approach

The resulting optimization problem is an integer programming problem with a space of 3 choices for each of 3 conditions in 4 controllers considered, making $3^{3*4} = 531441$ total possible solutions. Due to the scope of these design changes and assumptions made in the case study, only a few components must be considered in the cost function (design, operation, and scenario costs), since they will be constant throughout the optimization. Additionally, because of the context of system, which operates in space, where there is no ability to repair or maintain the system, the repair costs and future state costs are not included in this function. Stated in negative null form, the resulting adaptation of the resilient scoring function is:

$$\begin{aligned} &\text{min.} && \sum_{n\in N} C_n(\vec{x}) + \sum_{e\in E} P_e * t_m * (\sum_{l\in L} \bar{c}_l[\vec{s}_{f(\vec{x}),l}]) \\ &\text{wrt.} && \vec{x} \\ &\text{where} && x \in [1,2,3] \end{aligned}$$

where the notation is consistent with that outlined in Section 3, except P_e is the probability of an event (which does not change with changes in condition), t_m is an (assumed constant) mitigation time, E is the set of single fault (and no-fault scenarios).

This problem is readily encoded as an integer vector, and can be solved using an evolutionary algorithm following the general optimization process shown in Figure 1. Although many integer programming approaches are possible, the evolutionary algorithm used in this paper works in the following way:

1. Initialization: A population of solutions is randomly generated and evaluated
2. Generation: A new population of the same size is created from the population using two types of transitions:
   (a) Randomizing one condition in the solution, and
   (b) Randomizing one controller in the solution.
3. Evaluation: The new population is evaluated and given a fitness based on the scoring function.

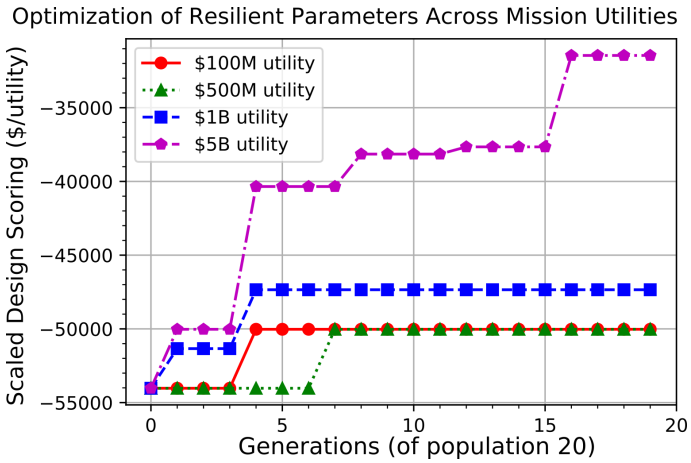

FIGURE 5. Cost optimization of the functional model using the evolutionary algorithm over differing mission utilities, showing the ability of our framework to increase mission value in a range of situations. Resilient features have more value when there is a higher mission utility.

4. Selection: Both populations are combined and the best half of solutions are selected for use in the next generation.
5. Iteration: Steps 1-4 are repeated over a number of generations or until a satisfactory best fitness is reached.

Since a good solution is initially known, the initial population is initially seeded with the solution of *EqualControl* for each state of each condition in each function to speed the solution process.

4.2 Results

To show the usefulness of this framework for exploring a wide range of design solutions, this framework is applied to the monopropellant system considering a range of different potential utilities for the mission of the orbiter. This is done by scaling the failure cost matrix by the utility of the mission, since the failure costs for this system are simply the utility lost by not being able to complete its mission. Applying this algorithm over 20 generations of population 20 gives the results shown in Figure 5. To compare the relative increase in value of using this design framework across mission utilities, the scoring function is scaled by the relative amount of utility of the mission.

As can be seen in Figure 5, depending on the amount of utility derived by the mission, the framework applied here with the algorithm is able to better increase the overall value of the design. For missions with a relatively small amount of utility at stake (100M or 500M), this increase is relatively small, because the design costs of implementing resilient features does not outweigh the resulting decrease in the cost of failure. On the other hand, for missions with a large amount of utility at stake (1B or 5B), there is more value gained by allowing the system to compensate for failures using the changes defined here. This shows how the framework presented here can be used both to explore a

large space of design solutions and to aid decision-making as to when to pursue resilient features.

5 Conclusions

This paper presents a framework for considering resiliency in early design using a scoring function to trade-off between the expected design, operational, and fault response costs. This framework is constructed to integrate with a comprehensive scenario-based fault simulation which determines the propagation of faults by mapping the costs of the failure event, its recovery, and the recovered states to flow states and failure modes in each scenario. Possible approaches towards optimizing this function are discussed, with a focus on the parameters available to optimize and related concerns. This framework is then demonstrated using the optimization of controlling functions in a Monopropellant system, using assumptions appropriate to the scope of the system design. Results of this optimization show an increase in overall mission scoring when pursuing resilient strategies which increases for missions with large utilities and resulting costs of failure. This case study demonstrates how this framework can be used to simultaneously explore resilient design features and negotiate trade-offs between the resilience and cost.

5.1 Future Work

A variety of possible directions present themselves for future work. In the near term, more case studies which use other types of design changes would be helpful towards understanding how this framework can be applied in the general case. Of particular interest is the ability to apply this framework towards making large structural changes in the functional model (e.g. adding wholly new function and flow chains). Additionally, the scoring function should be extended to accommodate the time-based system degradation, a key component of resilience [38], to enable consideration of health degradation (and increase in fault probabilities) over time. From a design perspective, this would allow the designer to trade off between maintenance, prevention, and fixing faults after they occur. Finally, further development will be spent towards automating the generation of this score function by integrating as many conceptions presented here (e.g. associating costs with flow state) with the IBFM codebase. The overall goal for future development is a comprehensive framework which may be used to optimize all available variables in a structured process that allows designers to automate the generation, evaluation, and optimization of large spaces of model variants to discover more resilient designs early in the design phase.

6 Acknowledgements

This research is in part supported by the NASA Ames Research Center (award number NASA NS295A) as an I/UCRC Center for e-design project (award NSF IIP-1362167). Any opinions or findings of this work are the responsibility of the authors and do not necessarily reflect the views of the sponsors or collaborators.

REFERENCES

[1] Rogers, E., 1986. Report to the President by the Presidential Commission on the Space Shuttle Challenger Accident. Tech. rep., National Aeronautics and Space Administration, Washington, DC.

[2] Seife, C., 2003. "Columbia disaster underscores the risky nature of risk analysis". *Science,* ***299***(5609), pp. 1001–1002.

[3] US Military Standard, 1980. Procedures for performing a failure mode, effect, and criticality analysis. Technical Standard MIL-STD-1629A, Department of Defense.

[4] Vesely, W. E., Goldberg, F. F., Roberts, N. H., and Haasl, D., 1981. Fault Tree Handbook. Handbook NUREG-0492, U.S. Nuclear Regulatory Commission, Jan.

[5] de Kleer, J., and Kurien, J., 2003. "Fundamentals of model-based diagnosis". *IFAC Proceedings Volumes,* ***36***(5), June, pp. 25–36.

[6] Kurtoglu, T., and Tumer, I. Y., 2008. "A graph-based fault identification and propagation framework for functional design of complex systems". *Journal of Mechanical Design,* ***130***(5), p. 051401.

[7] Stone, R. B., Tumer, I. Y., and Van Wie, M., 2004. "The Function-Failure Design Method". *Journal of Mechanical Design,* ***127***(3), July, pp. 397–407.

[8] Lough, K. G., Stone, R. B., and Tumer, I., 2006. "The risk in early design (RED) method: Likelihood and consequence formulations". In ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference.

[9] Lough, K. G., Stone, R., and Tumer, I. Y., 2009. "The risk in early design method". *Journal of Engineering Design,* ***20***(2), pp. 155–173.

[10] Sierla, S., Tumer, I., Papakonstantinou, N., Koskinen, K., and Jensen, D., 2012. "Early integration of safety to the mechatronic system design process by the functional failure identification and propagation framework". *Mechatronics,* ***22***(2), pp. 137–151.

[11] Coatanéa, E., Nonsiri, S., Ritola, T., Tumer, I. Y., and Jensen, D. C., 2011. "A framework for building dimensionless behavioral models to aid in function-based failure propagation analysis". *Journal of Mechanical Design,* ***133***(12), p. 121001.

[12] Papakonstantinou, N., Sierla, S., Jensen, D. C., and Tumer, I. Y., 2011. "Capturing interactions and emergent failure behavior in complex engineered systems at multiple scales". In ASME 2011 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. 1045–1054.

[13] Li, Z. S., and Mobin, M. S., 2015. "System reliability assessment incorporating interface and function failure". In Reliability and Maintainability Symposium (RAMS), 2015

Annual, IEEE, pp. 1–8.
[14] Oh, Y., Yoo, J., Cha, S., and Son, H. S., 2005. "Software safety analysis of function block diagrams using fault trees". *Reliability Engineering & System Safety,* ***88***(3), pp. 215–228.
[15] Krus, D., and Lough, K. G., 2009. "Function-based failure propagation for conceptual design". *AI EDAM,* ***23***(4), pp. 409–426.
[16] McIntire, M. G., Keshavarzi, E., Tumer, I. Y., and Hoyle, C., 2016. "Functional Models With Inherent Behavior: Towards a Framework for Safety Analysis Early in the Design of Complex Systems". In ASME 2016 International Mechanical Engineering Congress and Exposition, American Society of Mechanical Engineers, pp. V011T15A035–V011T15A035.
[17] Keshavarzi, E., McIntire, M., Goebel, K., Tumer, I. Y., and Hoyle, C., 2017. "Resilient System Design Using Cost-Risk Analysis With Functional Models". In ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, p. V02AT03A043.
[18] Short, A.-R., Lai, A. D., and Van Bossuyt, D. L., 2017. "Conceptual design of sacrificial sub-systems: failure flow decision functions". *Research in Engineering Design*, pp. 1–16.
[19] Erden, M. S., Komoto, H., van Beek, T. J., D'Amelio, V., Echavarria, E., and Tomiyama, T., 2008. "A review of function modeling: Approaches and applications". *Ai Edam,* ***22***(2), pp. 147–169.
[20] Stone, R. B., and Wood, K. L., 2000. "Development of a functional basis for design". *Journal of Mechanical design,* ***122***(4), pp. 359–370.
[21] Kruse, B., Gilz, T., Shea, K., and Eigner, M., 2014. "Systematic comparison of functional models in sysml for design library evaluation". *Procedia CIRP,* ***21***, pp. 34–39.
[22] Wood, K. L., Stone, R. B., Mcadams, D., Hirtz, J., and Szykman, S., 2002. A functional basis for engineering design: Reconciling and evolving previous efforts. Tech. rep., National Institute of Standards and Technology.
[23] Jänsch, J., and Birkhofer, H., 2006. "The development of the guideline vdi 2221-the change of direction". In DS 36: Proceedings DESIGN 2006, the 9th International Design Conference, Dubrovnik, Croatia.
[24] Holling, C. S., 1973. "Resilience and stability of ecological systems". *Annual review of ecology and systematics,* ***4***(1), pp. 1–23.
[25] Holling, C. S., 1996. "Engineering resilience versus ecological resilience". *Engineering within ecological constraints,* ***31***(1996), p. 32.
[26] Masten, A. S., 2001. "Ordinary magic: Resilience processes in development.". *American psychologist,* ***56***(3), p. 227.
[27] Briguglio, L., Cordina, G., Farrugia, N., and Vella, S., 2009. "Economic vulnerability and resilience: concepts and measurements". *Oxford development studies,* ***37***(3), pp. 229–247.
[28] Perrings, C., 2006. "Resilience and sustainable development". *Environment and Development Economics,* ***11***(4), pp. 417–427.
[29] Saint-Arnaud, S., and Bernard, P., 2003. "Convergence or resilience? a hierarchical cluster analysis of the welfare regimes in advanced countries". *Current sociology,* ***51***(5), pp. 499–527.
[30] Cohen, R., Erez, K., Ben-Avraham, D., and Havlin, S., 2000. "Resilience of the internet to random breakdowns". *Physical review letters,* ***85***(21), p. 4626.
[31] Ash, J., and Newth, D., 2007. "Optimizing complex networks for resilience against cascading failure". *Physica A: Statistical Mechanics and its Applications,* ***380***, pp. 673–683.
[32] Lengnick-Hall, C. A., Beck, T. E., and Lengnick-Hall, M. L., 2011. "Developing a capacity for organizational resilience through strategic human resource management". *Human Resource Management Review,* ***21***(3), pp. 243–255.
[33] Ponomarov, S. Y., and Holcomb, M. C., 2009. "Understanding the concept of supply chain resilience". *The international journal of logistics management,* ***20***(1), pp. 124–143.
[34] Linkov, I., Bridges, T., Creutzig, F., Decker, J., Fox-Lent, C., Kröger, W., Lambert, J. H., Levermann, A., Montreuil, B., Nathwani, J., et al., 2014. "Changing the resilience paradigm". *Nature Climate Change,* ***4***(6), p. 407.
[35] Hosseini, S., Barker, K., and Ramirez-Marquez, J. E., 2016. "A review of definitions and measures of system resilience". *Reliability Engineering & System Safety,* ***145***, pp. 47–61.
[36] Yodo, N., and Wang, P., 2016. "Engineering resilience quantification and system design implications: A literature survey". *Journal of Mechanical Design,* ***138***(11), p. 111408.
[37] Hu, C., and MacKenzie, C. A., 2017. "Optimizing resilience when designing engineered systems". In ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. V02AT03A047–V02AT03A047.
[38] Haimes, Y. Y., 2009. "On the definition of resilience in systems". *Risk Analysis,* ***29***(4), pp. 498–501.
[39] Henry, D., and Ramirez-Marquez, J. E., 2012. "Generic metrics and quantitative approaches for system resilience as a function of time". *Reliability Engineering & System Safety,* ***99***, pp. 114–122.