AN ABSTRACT OF THE DISSERTATION OF

Daniel Hulse for the degree of Doctor of Philosophy in Mechanical Engineering presented on December 7, 2020.

Title: A Computational Framework for Resilience-Informed Design

Abstract approved: _____

Irem Y. Tumer Christopher Hoyle

It is desirable for complex engineered systems to perform missions efficiently and economically, even when these missions' complex, variable, long-term operational profiles make it likely for hazards to arise. It is thus important to design these systems to be resilient so that they will actively prevent and recover from hazards when they occur. To most effectively design a system to be resilient, the resilience of each design alternative should be quantified and valued so that it can be incorporated in the decision-making process. However, considering resilience in early design is challenging because resilience is a dynamic and stochastic property characterizing how the system performs over time in a set of unlikely-but-salient hazardous scenarios. Quantifying these properties thus requires a model to simulate the system's dynamic behavior and performance over the set of hazardous scenarios. Thus, to be able to incorporate resilience in the design process, there is a need to develop a framework which implements and integrates these models with design exploration and decision-making. This dissertation fulfills this need by defining resilience to enable fault simulations to be incorporated in decision-making, devising and implementing a modelling framework for early assessment of system resilience attributes, and exploring optimization architectures to efficiently structure the design exploration of resilience variables. Additionally, this dissertation provides a validity testing framework to determine when the resilient design process has been effective given the uncertainties present in the design problem. When each of these parts are used together, they

comprise an overall framework that can be used to consider and incorporate system resilience in the early design process.

A Computational Framework for Resilience-Informed Design

by

Daniel Hulse

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented December 7, 2020
Commencement June 2021

Doctor of Philosophy dissertation of Daniel Hulse presented on December 7, 2020.

APPROVED:

Co-Major Professor, representing Mechanical Engineering

Co-Major Professor, representing Mechanical Engineering

Head of the School of Mechanical, Industrial, and Manufacturing Engineering

---

Dean of the Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my dissertation to any reader upon request.

Daniel Hulse, Author

ACKNOWLEDGEMENTS

Many people helped support this work at different points during the conceptualization, development, and writing process, and they deserve acknowledgement. Below I attempt to credit the people who helped support this work in ways big and small.

First I would like to express gratitude to my advisors. Thank you to Dr. Christopher Hoyle for being consistently available and willing to discuss the research approach, agenda, details, and plan. His timely feedback on the work when needed has helped the project move forward. Thanks also to Dr. Irem Tumer for her sharp guidance and help with both planning out this dissertation and my Ph.D work in general. She has been fantastic at helping me keep the high-level plan and goals in mind as I have progressed through the research.

Second, I would like to recognize the help of my project collaborators. Thank you to Kai Goebel for you detailed, critical feedback and for help with understanding how this work fits into the broader PHM field. It has helped each paper achieve a crucial level of quality prior to submission. Additional thanks to Chetan Kulkarni, for engaging with this work and giving helpful comments.

Third, I would like to especially acknowledge the help of my lab-mates who contributed to this work. Hannah Walsh deserves both thanks and credit for her expertise with network models and her ability and willingness to discuss research in a way that has been generative. Thanks also to the other NASA smart-stereo project members, Sequoia Andrade and Eleni Spirakis, for helping me test, workshop, and demonstrate the fmdtools toolkit, which has been helpful for understanding some of the usage considerations to be addressed in future work. I would additionally like to thank and credit Arpan Biswas for his code contributions and help understanding bi-level optimization architectures and conceptualization of the optimization approaches and comparisons used in this work.

Fourth, I would like to extend thanks to my NASA collaborators, including Guillaume Brat and Misty Davies, for supporting this work. This support has helped me workshop ideas and develop the resilience modelling framework–the key contribution which enables the rest of this work–into a

rich environment for expressing fault behaviors, and will help it continue to become an increasingly useful and relevant tool going forward.

Finally, I would like to thank my current and former lab-mates, including Lukman Irshad, Hongyang Zhang, Nico Soria, Trung Pham for their willingness to talk about research and interest in the project at various times throughout the program. Hongyang Zhang especially deserves credit for his contributions to the the pandemic model in the fmdtools examples repository.

CONTRIBUTION OF AUTHORS

This work was largely adapted from previously published and submitted manuscripts which were written from the start of Spring 2018 to present collaboratively with a number of different co-authors who contributed in both directly to the work (i.e. through coding and writing) and in an advisory capacity (i.e., through conceptualization, feedback, revision, etc.). This section provides a short research narrative which traces these contributions to their corresponding sections of the dissertation and notes the direct contribution of authors, if present (unless otherwise noted, coauthors can be assumed to have served in an advisory capacity).

- Daniel Hulse, Christopher Hoyle, Kai Goebel, and Irem Y Tumer. Quantifying the resilience-informed scenario cost sum: A value-driven design approach for functional hazard assessment. *Journal of Mechanical Design*, 141(2), 2019

  Work on this dissertation started with this journal article, which conceptualized the original idea of a combined design-modelling-optimization framework for resilience using expected cost scoring based on fault simulations. This work was adapted for use in:

  - Section 5.4, the monopropellant system design problem used to exemplify resilience optimization and concept selection;
  - Section 5.2, an exploration of considerations specific to formulating and solving resilience optimization problems using function-based fault models;
  - Section 3.3, the formulation of the expected-cost based resilience function for IBFM models;
  - Section 3.5, a simple design problem used to exemplify the framework (the validation section was added afterward);
  - Section 2.1, the background section about resilience definitions, and
  - Section 2.2.3, background section about function-based fault modelling approaches;

– Sections 3.7 and 5.6, the conclusions for the resilience metric and optimization chapters.

Much of this work was previously presented in the conference publication:

– Daniel Hulse, Christopher Hoyle, Kai Goebel, and Irem Y Tumer. Optimizing function-based fault propagation model resilience using expected cost scoring. In *ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages V02AT03A052–V02AT03A052. American Society of Mechanical Engineers, 2018

- Daniel Hulse, Christopher Hoyle, Irem Y Tumer, and Kai Goebel. How uncertain is too uncertain? Validity tests for early resilient and risk-based design processes. *Journal of Mechanical Design*, pages 1–23, 2020

  To consider the validity of the overall proposed design framework, the work in this journal article developed a testing approach to consider how uncertainty can affect the design process. Chapter 6 is heavily based on this work, with a few revisions, as was the background in Section 2.1.2 regarding the consideration of uncertainty in the design process. This journal article itself built on previous work that was presented in the conference papers:

  – Daniel Hulse, Christopher Hoyle, Kai Goebel, and Irem Tumer. Using value assessment to drive phm system development in early design. In *Proceedings of the Annual Conference of the PHM Society*, volume 11, 2019

    This conference paper helped conceptualize the design of resilience as a decision-making process that could be used for early Prognostics and Health Management system development and began to conceptualize the effect of discrete uncertainty on the design process. Some text and figures were adapted for use in Section 3.1 and this paper featured an early version of the aircraft PHM design problem in Section 6.3

  – Daniel Hulse, Christopher Hoyle, Irem Y Tumer, and Kai Goebel. Decomposing incentives for early resilient design: Method and validation. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*,

volume 59193, page V02BT03A015. American Society of Mechanical Engineers, 2019

This conference paper produced the original conception of the continuous validity-testing approach in Section 6.2.2.2 as well as the continuous-uncertainty EPS redundancy example in Section 6.4.

- Daniel Hulse, Christopher Hoyle, Irem Tumer, Chetan Kulkarni, and Kai Goebel. Temporal fault injection considerations in resilience quantification. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers, 2020. IDETC2020-19287

  During the development of the fmdtools modelling framework, this conference paper developed a corresponding sampling approach for resilience quantification for use with dynamic fmdtools models. This paper additionally explored the effect of different sampling approaches in resilience quantification, and discussed the the considerations for using these approaches in the design process. In this dissertation, this work was adapted to:

  - Section 3.4, the description of the resilience sampling approach for use in dynamic models,
  - Section 3.6, the pump example, which is used both to demonstrate resilience quantification and study the effect of different sampling approaches on accuracy, and
  - Section 2.2.4, which has relevant background on representing fault scenarios in risk quantification.
  - The second paragraph of the conclusions in Section 3.7 regarding the temporal fault sampling approach.

- Daniel Hulse, Hannah Walsh, Andy Dong, Christopher Hoyle, Irem Tumer, Chetan Kulkarni, and Kai Goebel. fmdtools: A fault propagation toolkit for resilience assessment in early design. 2020

  Work on the fmdtools modelling framework was ongoing began in the Summer of 2018, after it was determined that the IBFM framework used in the initial paper limited the expression

of dynamic fault behaviors and made it difficult to parameterize models for optimization. Development of this tool went through a number of prototyping phases before taking the form used in this work around the Fall of 2019. This codebase was then developed into the overall modelling, simulation, and visualization environment described in this paper, which was submitted in Spring 2020 and is currently in review at the International Journal of Prognostics and Health Management (IJPHM). Hannah Walsh contributed directly both to the development of this tool (developing network analysis codes that could be used in fmdtools models) and by writing the corresponding sections in the paper (Sections 4.2.2.1 and 4.3.1). This manuscript has been adapted to:

  - Chapter 4, except for Section 4.4, which describes the fault modelling toolkit, and
  - Section 2.2, the background on previously-developed fault simulators

- Daniel Hulse, Sequoia Andrade, Eleni Spirakis, Hannah Walsh, and Misty Davies. Smartstereo: Preliminary model description. Technical Report 20205007481, NASA Ames Research Center, September 2020. https://ntrs.nasa.gov/citations/20205007481

  This Technical Memo describes an fmdtools model that was collaboratively developed with Sequoia Andrade, Eleni Spirakis and Hannah Walsh at NASA Ames Research Center in the Summer of 2020. This work was included here because it exemplified the ability of the fmdtools modelling framework to simulate complex dynamic interactions and to be extended to express and represent specialized behaviors. This work was summarized for use in Section 4.4 with this end in mind.

- Daniel Hulse, Arpan Biswas, Christopher Hoyle, Irem Tumer, Chetan Kulkarni, and Kai Goebel. Exploring architectures for integrated resilience optimization

  Once the simulation framework was in place, the ultimate goal of formalizing the problem and exploring potential solution architectures was pursued, resulting in the manuscript above, which was submitted in the Fall of 2020 and is currently in review at the AIAA Journal of Aircraft Information Systems (JAIS). Arpan Biswas contributed both to the conceptualization

and implementation of the optimization optimization architectures and to the resulting writing used in this work (see Sections 2.3, 5.2, and 5.5). This manuscript has been adapted to the following sections:

- the background about optimization architectures and resilience optimization approaches in Section 2.3 and Section 2.1.1
- Section 5.2, the description of the integrated resilience optimization problem and elaboration of optimization architectures for this problem
- Section 5.5, the example integrated optimization on a drone problem and the comparison of optimization architectures.
- Section 5.6, the conclusions about the use of the optimization architectures presented in the examples.

These works were compiled, placed in the dissertation structure, revised to fit this overall work, and supplemented with an overall narrative (i.e., introduction, conclusions, missing fragments in each chapter) over the course of 2020.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF TABLES

LIST OF TABLES (Continued)

Chapter 1: Introduction

For a system to be viable, it needs to respond well to disruptions. When a system operates in an uncertain, uncharacterized, variable, or dynamic environment with hazards to safety and economy, it is important for the system to be able to adapt and recover the behaviors needed to achieve its overall function. This property is especially relevant to complex engineered systems, which must be operate sustainably over a desired lifecycle to achieve a mission–even in the possibility of internal or externally-caused hazards and faults. The following examples of the Galileo spacecraft and Mars Surveyor 98' program illustrate how a system's response to hazards can determine whether a mission ultimately succeeds or fails.

The NASA Galileo spacecraft was launched in 1989 to study Jupiter and its moons [1]. However, in April 1991 when the main antennae was scheduled to deploy, operators determined that the ribs of the antennae were stuck and the antennae could not be deployed [129]. Nevertheless, the spacecraft still had functioning low-gain antennas which could be used to send small amounts of data. The operators were able to reconfigure these antennas to send data at a higher throughput by arraying the antennaes, improving modulation efficiency, using more efficient channel coding, changing the telemetry to a telemetry-based packeting scheme, using compression algorithms to reduce file size, and giving the spacecraft more time on the Deep Space Network antennae network used to send files [129]. As a result, it completed its mission in 2003, with a majority of its original science goals complete and three mission extensions which enabled researchers to achieve more scientific goals than originally envisioned, despite this and other faults [1]. This has been noted as a case where the system was resilient–the reconfigurability in the system enabled the operators to continue the mission of the system despite faults [58].

The Mars Surveyor 98' program, on the other hand, was launched in 1999 to study the Martian climate, polar ice cap and soil [65]. However, upon reaching the Martian atmosphere, contact was

lost with the Mars Polar lander, Deep Space 2, and Mars Climate Orbiter (which was meant to act as a relay to the Mars Polar Lander) [65]. While it was determined that a unit conversion led to a failure in the Mars Climate Orbiter–causing it to descend too low [31]–there was not enough evidence to conclude the definitive cause of the lander and probe failures, though premature engine shutdown due to improper sensor readings from the lander legs is considered the likeliest cause [5, 30]. In this case, the system was not resilient to faults: it failed immediately and catastrophically. This was not only because of the loss of communication and short course of events, but also because of a lack of reconfigurability and fragility in the system design (sensitivity of the descent process to sensor readings).

These examples illustrate how resilience is a desirable property to incorporate in the design of a system: when a system is resilient to faults, it succeeds, when it is not resilient to faults, it fails. However, while resilience can define the success or failure of complex engineered systems, it can also be difficult to define. Early theories of resilience emerged from the study of natural systems [230] (and later complexity science [237]), where it was observed that ecosystems can maintain their overall "function"–or high-level behavior–even when individual species are removed [98]. Furthermore, natural systems often can also self-recover after disruptions without outside intervention. This conception of responding to hazards is different from the risk mitigation approaches often pursued in engineering, where the goal is merely to prevent known failures [100], often at the expense of performance and economy. Thus, truly seeking resilience in an engineered system means considering not just the risk typically considered in traditional reliability engineering, but also the ways flexibility can be used to achieve system recovery post-fault [105, 54]. Finally, recent technological developments in Prognostics and Health Management have led engineers not only to be able to design the system to recover from faults, but to proactively predict and mitigate faults before they occur [145]. To consider these new technologies, considering resilience in engineering must additionally take into account the active fault prevention strategies they enable.

Resilience in engineering has, as a result, come to include not just post-fault reconfiguration and recovery, but pre-fault prediction and prevention [304, 255]. To enable this broad consideration of

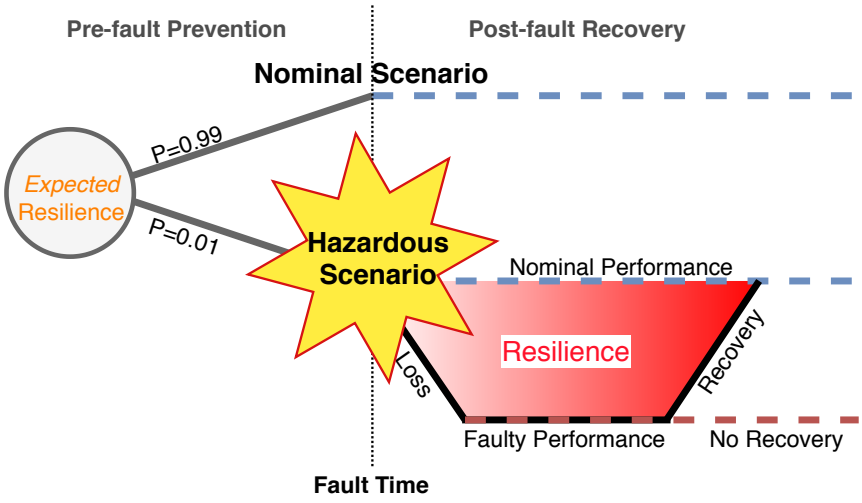


Figure 1.1: Definition of resilience used in this work: the expected lost performance over a set of scenarios, which captures the effect of both pre-fault prevention and post-fault recovery strategies.

Table 1.1: Design and operational approaches that can be used to achieve system resilience.

	Prevention		**Recovery**	
Approach	Reliability	Prognostics & Health Management		Safety
Control Actions	Maintenance Schedule	Predictive Maintenance	Diagnostics & Reconfiguration	Emergency Procedures
Design Features	Reliability, Fault-free Design	Sensors & Hardware	Sensors & Flexibility	Redundancies Fail-safes

resilience, this work adopts the definition of *expected* resilience shown in Figure 1.1. In this definition, resilience is the dynamic deviation of performance which occurs as a result of entering a hazardous scenario. This captures both the robustness of the system to faults and the ability of the system to recover post-fault. *Expected* resilience is then the statistical expectation of this lost performance over the set of hazardous scenarios considered over the lifecycle of the system. This definition enables one to additionally consider the inherent reliability of the system (that is, the susceptibility of the system to hazardous scenarios) and the ability of the system to proactively mitigate faults before they occur through maintenance.

To ensure that complex engineered systems are resilient, resilience needs to be considered and incorporated in the decision-making process, both for the design of the system and the resulting op-

erations, as shown in Table 1.1. As shown, ensuring resilience can involve both traditional reliability and safety techniques for fault prevention and recovery and Prognostics and Health Management (PHM) approaches which use specialized sensors and diagnosis/prognosis hardware and software to prevent the fault beforehand or mitigate the fault afterward. Because these systems rely on designed features of the system, they require more attention in the design process to function well. Ideally, this should be present at each stage of the design process–starting at the earliest stages–to take the most advantage of design effort (see: [273]). However, incorporating resilience in design is challenging because one must model the behaviors in the system post-fault and trade this resilience against other design considerations (e.g. performance) over a wide and uncertain space of potential design solutions. Over a large set of design solutions and considered failure scenario, performing this sort of analysis by hand is intractable, especially when one needs to iteratively change design solutions. Thus, to be able to consider different resilient design solutions (especially for PHM systems), there is a need to for there to be a unified computational framework that enables one to tractably represent and explore different design alternatives, simulate faults and their resulting behaviors, and use these fault responses and resulting resilience metrics to select the optimal design.

1.1 Research Objectives and Contributions

The intent for this dissertation was to understand how to use computation to aid the consideration of resilience in early stages of design. Broadly, this meant pursuing the following aims:

- studying the decision-theoretic aspects of resilience to understand how resilience can be traded with other desirable design attributes and how uncertainties in this understanding can affect the preferability of designs and validity of the resulting design process;
- determining how the defining attributes of resilience–the dynamic effects of faults–can best be modelled in the early design phase to best represent the real physical behaviors while enabling fast iterative evaluations in the design process; and
- investigating the use of mathematical optimization to automatically explore and evaluate large

spaces of designs by developing optimization architectures that best structure the problem to achieve desired solution quality while reducing iteration time.

Thus, the overall goal of this work is a unified computational framework to enable the consideration of system resilience in the early design phase. To achieve this, this framework pursues the four main research objectives described in the next subsections.

1.1.1 Research Objective 1: A definition of resilience that enables the trading of resilience with other desirable attributes in design.

Research Objective 1 studies how to make decisions that consider a variety of resilient design alternatives. The challenge of making decisions based on resilience is considering the effects of design features which may mitigate a large number of faults while adding new hazards, changing performance attributes, and/or increasing overall system complexity. To approach this challenge, this work develops a decision-theoretic definition of resilience and investigate its use in the design process. Previous work has shown that expected cost has a number of advantages compared to the traditionally-used Risk Priority Number, including consistent risk prioritization and the ability to trade it one-to-one with design and operational costs [147].

As described previously, using the *expected resilience* enables one to consider both prevention and recovery approaches to achieve resilience. However, even with this definition it may still be difficult to justify design features based on resilience alone when they also effect the implementation and operations of the system. In design, it is desirable to choose concepts based on their cost, value [50], or utility [90] in a holistic model that takes into account all desirable and undesirable attributes of the concept [259]. Thus, to account for resilience in design decision-making, this work develops an expected cost of resilience model that enables one to trade system resilience against design and operational costs in early design. Specifically, the contributions of this work are:

- A generic decision-theoretic definition of expected resilience for use in design. Presented in Section 3.2, this work identifies and quantifies the sources of cost that result from different

fault scenarios: disruptions, loss of performance, and repairs.

- Adaptation of this definition to static fault models developed in the IBFM fault modelling toolkit presented in previous work [185]. Presented in Section 3.3, this work creates an expected resilience cost model specifically for this previous fault modelling methodology to enable the consideration of resilience in this modelling paradigm.

- Adaptation of this definition to the dynamic fault simulation modelling methodology developed for Research Objective 2 (Chapter 4). Presented in Section 3.4, this work creates an expected resilience cost model for the fmdtools simulation toolkit and explores how to sample fault modes in time for accurate and efficient quantification of expected resilience measures.

Using this approach, designers will be able to make choices systematically: balancing the value of resilient features against the cost of operation and implementation to make the best overall decision.

1.1.2 Research Objective 2: A modelling approach and toolkit to simulate the resilience of a system and enable design.

Research Objective 2 develops an approach to model resilience in the early design process using dynamic fault simulation. The challenge to modelling resilience in early design is balancing simulation cost (both in terms of setting up and running the model), level of abstraction (the ability to inform high-level choices without getting into details) and accuracy. Current methodologies for modelling faults in systems are not suitable for representing expected resilience in design because they foccus on the immediate effects of hazards (e.g., [185, 39]) or because they are are based on late-design representations of the system (e.g., [104, 136]) that are difficult to leverage in early design [80]. Thus, this work develops a method to simulate a system's fault response to enable resilience quantification and early conceptual design, providing these two contributions:

- A modelling methodology and toolkit for simulating the resilience behaviors of a system over a set of fault scenarios to quantify the expected resilience of a system. This implementation,

presented in Section 4.2, further enables early design by allowing the system to be modelled at different levels of fidelity through the design process, visualization of resilience metrics on model structures, and parameterization/optimization;

- Verification and study of dynamic resilience scenario sampling approaches, which is performed in Section 3.6 in the context of expected cost modelling.

With these contributions, designers will be able to simulate the dynamic fault responses of a system over time to understand and quantify resilience early in the design process. Because the resulting toolkit is developed specifically for the early design process, designers will be able to use it build resilience into a system without being slowed down by inefficient model specification or computationally expensive simulations.

1.1.3 Research Objective 3: Architectures to efficiently structure the optimization of resilience.

Research Objective 3 studies how mathematical optimization can be used to explore the solution space to achieve optimal system resilience in the early design phase. The optimization of resilience requires one to consider both design features and the operations of the system over the set of possible contingencies. To approach these challenges, this work will investigate multidisciplinary design optimization architectures to efficiently and effectively structure the optimization problem. Multidisciplinary design optimization–a way of structuring an optimization problem into individual sub-problems to be solved at different levels–has a number of benefits, including the ability to choose the best algorithm for each sub-problem, the ability to exploit parallelism in model execution, the ability to federate models, and the ability to reduce the dimensionality to solve the problem more efficiently [97]. To adapt these architectures to the resilient design problem, this work provides the following contributions:

- A formalization of the integrated resilience optimization problem, in which the design features,

operational policy, and resilience policies are optimized in a unified framework, in Section 5.2;

- Exploration (Section 5.2) and comparison (Section 5.5.2.2) of optimization architectures to solve this problem in the most efficient and effective way, including bi-level, sequential, and all-in-one formulations as well as problem decomposition approaches which can be used to manage and reduce the complexity of the optimization problem.

Using these approaches, designers will be able to automatically explore the space design solutions to find the most resilient system without artificially limiting the design space to a particular set of variables. As a result, designers will be able to measure the full value of resilience-affecting early design stage choices (e.g., redundancy) with an overall health management approach by determining how each impacts the resulting operations (e.g., performance effects of weight) and recovery actions (e.g., reconfiguration and contingency management).

1.1.4 Research Objective 4: A validity testing framework for the resilient design process.

While the framework resulting from Research Objectives 1-3 can be adapted to many different design contexts, it may not be clear that a given instantiation of the resulting design process is valid given the underlying uncertainties in the model and parameter values [236, 19]. This is especially true in the early design process, when the space of uncertainty is very high and there is a wide space of potential solutions. To make sense of the validity of resilience-based design processes in these contexts, this work provides a testing framework (in Chapter 6) based on uncertainty propagation that helps the designer determine when a resilient design and analysis process was valid given the choice-affecting uncertainties present in design assumptions. Using this validity testing framework, designers will be able to understand when a given resilient design procedure provides meaningful results given the uncertainties in input parameters so that they can know when to proceed and when to seek more information.

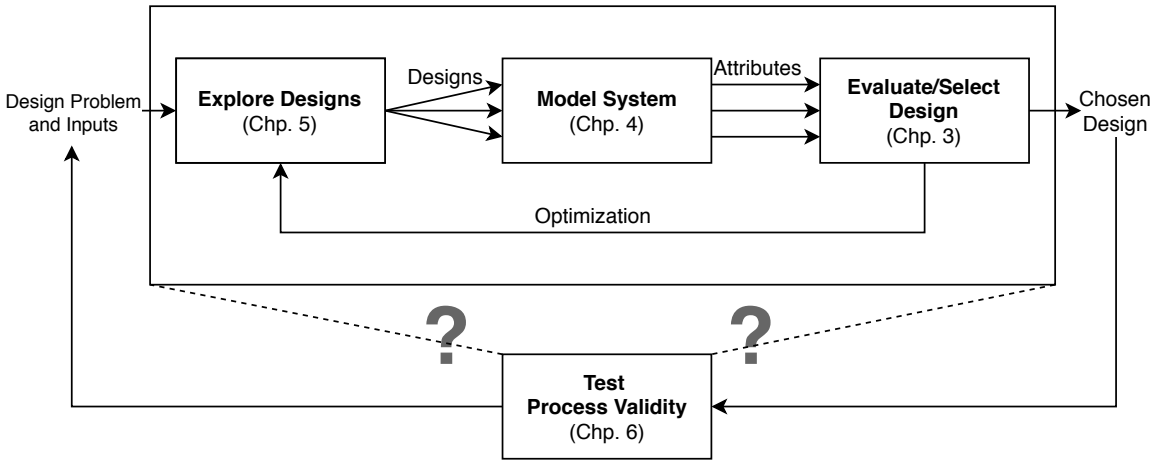


Figure 1.2: Design and design validation process enabled by this work

1.1.5 Overall Framework

This computational framework gives one the ability to consider and incorporate resilience in system design process, enabling one to carry out the procedure shown in Figure 1.2. In this process, one first defines the design problem using a high-level model characterizing the system faults, behavioral propagation, and the costs of hazardous or faulty states. Then, the design model is varied either through parameterization, developing model variants, or creating different models (a process that can occur iteratively as the designs become more detailed). Then the chosen design is selected based on the expected cost of resilience determined via fault simulation and modelled design and operational costs (a process that can also occur iteratively in an optimization loop). If one then has questions about the validity of this process, the tests can then be performed by quantifying uncertainty in the problem inputs and checking how it affects the results.

1.2 Motivating Example

To motivate the need for a computational framework for resilience-informed design, consider the design of a multirotor drone used for short aerial surveillance missions. This design problem has

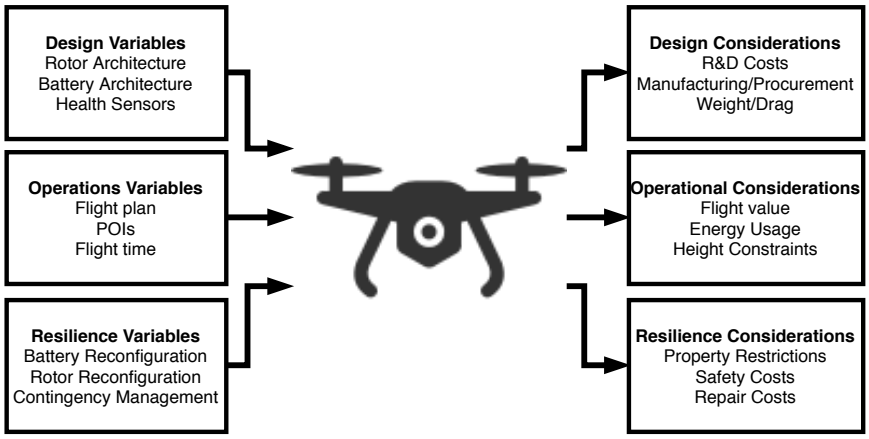


Figure 1.3: Resilience-related design variables and considerations in a multirotor drone design problem

a number of interconnected design, operational, and resilience-related considerations, as shown in Figure 1.3. For example, a multirotor drone could be used to surveil crowds or sensitive areas, but its ability to do so safely would rely on the resilience of the system afforded by its design (e.g., rotor architecture), operational profile (e.g., height and flight time over the area), and contingency management (i.e., how it would change the mission if a fault occurred). Some of these variables may have counter-intuitive effects. For example, an overly heavy redundancy architecture may make it less effective at pursuing its mission and may even make it less resilient by introducing faults and lowering the available flight time. Effectively designing a system like this is difficult because one must consider the complex trade-offs resulting from the interactions between design, operational, and resilience variables.

To effectively make sense of these interactions to inform design, it is first important to be able to understand the value of the different considerations so that resilience considerations can be balanced with design and operational considerations. In the drone design problem, one must balance the safety considerations of the drone losing stability in certain fault scenarios against the design and operational cost of using heavy redundancy architectures that reduce mission length. While creating models of design and operational costs for problems like this has been covered in previous work, similar cost metrics of resilience need to be quantified so that it can be taken into account in the

same decision. To solve this problem, Chapter 3 presents expected cost of resilience metrics which quantify the value of fault simulation results so it can be traded against design and operational costs in design.

While creating a cost model of resilience enables one to trade the resilience and design and operational costs of the drone, to inform these metrics one must first quantify the resilience of the design under consideration. To do this, one must consider how the drone operates under a number of different fault scenarios (e.g., if the battery has a short or a rotor jams). While one could perform this process manually (e.g., by creating FMEAs for each design variant), doing so would be tedious and difficult to perform in a consistent way for each design. To perform this process in a consistent way that expresses resilience (the dynamic behavior of the system post-fault) through the design process, one needs a dynamic simulation that enables one create a low-detail simulation in early design and increase fidelity as the design process progresses. To solve this problem, Chapter 4 presents a modelling toolkit and framework to model the system and simulate the effects of faults in it that supports modelling the system at different levels of fidelity.

Given that one has a model of the drone and a metric to trade resilience with design and operational considerations, one can then use them to compare different variants (e.g., different rotor architectures). While this can be done in a manual process (e.g., comparing each architecture), considering a wide space of variables requires a more systematic and exhaustive computational process, especially when the variables are coupled (e.g., each rotor architecture leads to different potential recovery actions). To perform this design exploration, Chapter 5 presents architectures to effectively structure the optimization of resilience.

Finally, given the uncertainties in modelling assumptions about failure rates and safety effects, it may be unclear that the design decisions made were valid. While one may have fairly good information about drone component reliability from previous operational data, there may be less data about the effectiveness of recovery systems under hazards, or the risk in the operational environment. This sort of data would be difficult to gather–especially if the system design is novel–and would result in a considerable amount of uncertainty regarding whether or not to choose one of these systems. As a

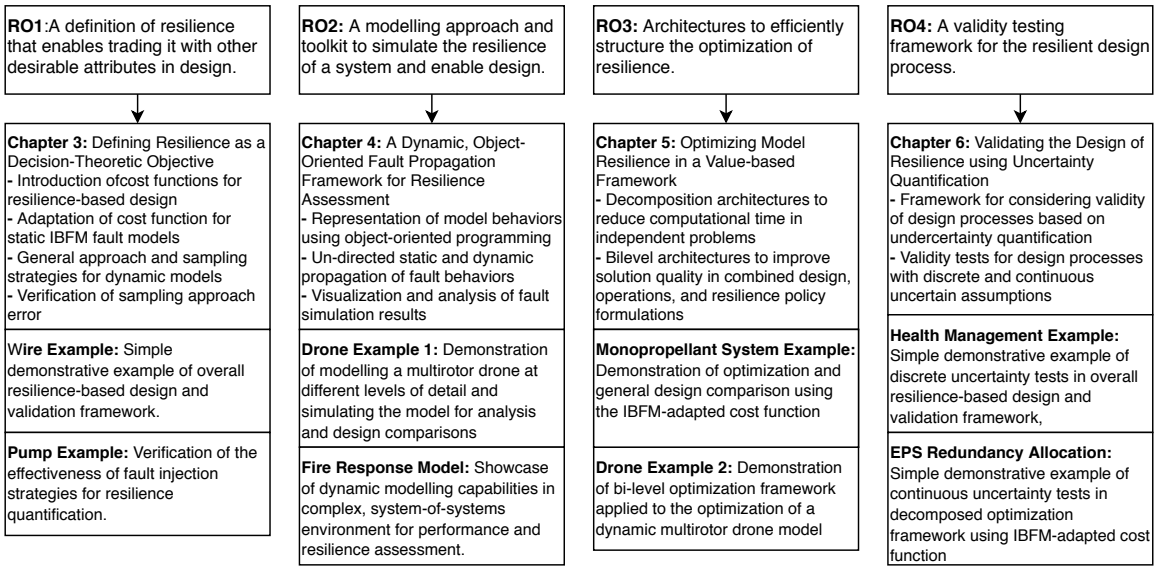


Figure 1.4: Embodiment of research objectives in paper.

result, it may not be clear whether a decision made using this data is valid, given the uncertainty. To consider this problem, Chapter 6 provides a framework to consider the effects of these uncertainties on the validity of the decision being made.

This example demonstrates how the methods and overall framework presented in this work inform the design of a real system. Further examples are provided throughout the work (see Table 1.2) which apply this work to system design problems. Specifically, more detailed drone design examples are provided in Section 4.3 and 5.5, which show how models can be used to represent a drone and how optimization and value modelling can be used to find an optimal design, respectively.

1.3 Organization

Before proceeding, this section describes the structure of the dissertation. Chapter 2 provides context into the methods provided, including how resilience has been incorporated in design previously and related considerations (Section 2.1), previously-used models for risk and resilience assessment (Section 2.2), and optimization architectures used in complex systems design problems (Section 2.3).

The framework and contributions of this work are then shown in 1.4. As shown, each research objective discussed in Section 1.1 is presented in Chapters 3-6. These objectives are then demonstrated with examples, which are provided at the end of each chapter. To summarize:

- Chapter 3 provides the definition of resilience as an expected cost used to make decisions in the overall design framework, with adaptations to the IBFM fault modelling framework provided in previous work [185], a general approach for quantifying these costs in the modelling framework in Chapter 4, and a comparison of the error of different sampling methods which can be used in this general approach.

- Chapter 4 presents a modelling toolkit and approach to represent faulty behaviors in engineered systems that enables a high-level representation and analysis of faults in the system in early design. This python toolkit enables the dynamic propagation of faults through an object-oriented model as well as the visualization and analysis of the resulting fault responses.

- Chapter 5 studies the optimization of resilience in the value-based frameworks and models presented in Chapters 3 and 4. In Section 5.2 it formalizes the joint optimization of design, operational, and resilience costs and presents architectures to efficiently and effectively structure the optimization of resilience, including a decomposition architecture to reduce computational time in uncoupled, independent risk-based design, and a bilevel architecture to improve solution quality in coupled design, operations, and resilience policy optimization. It then compares these architectures in terms of computational performance and optimization effectiveness (Section 5.5.2.2) and discusses their use in design (Section 5.5.3).

- Chapter 6 provides a framework for testing the validity of resilience-based design processes (such as those in Chapter 5 and those used in the examples throughout) based on quantifying the effects of the uncertainty underlying the assumptions used to make decisions. Sections 6.3 and 6.4 then show examples of where a design process fails and passes these tests, respectively, to show how this framework works in practice.

Finally, Chapter 7 discusses the use of the methods in the case study, reflects on the usefulness and

Table 1.2: How examples in this work demonstrate the presented methods.

	Wire	Pump	Drone 1	Fire Response	Monopropellant Sys.	Drone 2	PHM Sys.	EPS Red.
Chapter/Section	3.5	3.6	4.3	4.4	5.4	5.5	6.3	6.4
Resilience Quantification Methods								
Cost Modelling	X	X	X		X	X	X	X
IBFM Cost Adaptation (RISCS)					X			X
Resilience model cost sampling		X	X			X		
Modelling Methods								
Fmdtools fault propagation/visualization		X	X	X		X		
Design/Optimization Methods								
Design comparison	X	X	X	X	X		X	
Optimization					X	X	X	X
Decomposed optimization approach								X
Optimization architectures						X		
Validation Methods								
Discrete-uncertainty validity testing	X						X	
Continuous-uncertainty validity testing								X

limitations of the work, and identifies possible research directions for future work.

Examples are provided throughout chapters 3-6 to show the application of each part of this framework. While these examples demonstrate the contributions of their respective sections, because each successive chapter builds on the previous chapters, each example demonstrates multiple contributions and methods presented throughout. Table 1.2 summarizes how different methods presented in this work are demonstrated. As shown, while each example has as specific purpose described in Figure 1.4 (and no single example represents every provided method), as a whole the examples cover the set of methods provided in this work adequately to not only exemplify specific contributions, but the framework as a whole.

Chapter 2: Background

This chapter presents the background necessary to understand the research context, contribution, and theoretical basis of the computational design framework presented in this disseration. First, Section 2.1 presents previous in resilience-based design, including the definitions of resilience in the literature, previous resilience-based design and optimization frameworks, and work considering uncertainty in design. Then, Section 2.2 presents previous modelling approaches used for risk, reliability, and resilience-based design, including previous resilience modelling toolkits, related probabilistic risk assessment tools, previous function-based fault modelling methodologies which influenced the overall system representation used in the modelling framework in Chapter 4, and the types of fault models are used in risk assessment. Finally, Section 2.3 presents existing (bi-level, two-stage, multidisciplinary design optimization) approaches for structuring complex optimization problems like the ones persued in Chapter 5.

2.1 Resilience in Design

Broadly, resilience is the ability of a system to prevent and mitigate failure events [53], though quantitative and qualitative definitions are used across the fields of ecology, economics and management, and engineering [106]. A number of definitions of resilience have been introduced across a variety of fields, first emerging from ecology [99, 101, 230], where it was observed that ecological systems have the tendency to remain overall "function," even when specific species are removed [99]. This concept has further been adapted to the social sciences [173, 228, 228], network science [49, 14], and management [165, 234] (particularly the management and design of supply chains [42]).

As opposed to risk management, which focuses on failure prevention, strategies to achieve resilience in a system often focus on recovery from failures instead [169]. However, definitions and

metrics for measuring resilience vary across and within fields, with both qualitative and quantitative metrics [106]. Within the literature, distinctions have been made between these definitions:

- **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) [101].

- **Deterministic and probabilistic measures:** To summarize Hosseini et al. [106], probabilistic measures, such as passive and proactive survival rates [306] and expected displacement [71] incorporate uncertainty in system behavior while deterministic measures such as the value of resilience [95] and maximum disturbance [211] do not.

- **Dynamic and static measures:** Dynamic measures, such as the resilience triangle [36], recovery time, dynamic economic resilience [245], and similar definitions incorporate the time-based behavior of the system while static measures, such as static economic resilience and network structure resilience (e.g. [146]) do not.

Within the engineering discipline, resilience assessment has been considered a special case of risk assessment, where resilience is the expected time-based response of the system to faults: a combination of reliability and recoverability [304], or, as has been presented more recently, reliability, recovery, robustness, and reconfigurability [255]. Design for resilience often focuses on recoverability, defined as a product of the diagnosis capability, resource availability, and repair capability of the system [167], since prevention is already considered in traditional reliability approaches. Recent developments in resilience quantification have defined resilience as a statistical *expectation* over a set of simulations, such as the resilience index and fraction of simulations with resilient, failed, or recovered operations [56, 180], the loss of performance due to the fault behavior [201, 212], or, most commonly, the loss of value [195, 67, 174, 115].

This work takes the broad view that resilience is a comprehensive understanding of system

risk that necessarily incorporates the dynamic response of the system to faults but also includes its inherent reliability and active fault prevention. In prior work, risk and resilience have been considered competing or complementary approaches to understanding the susceptibility of a system to hazards [17], with risk generally quantified as failure probability and resilience quantified through a number of metrics (e.g., recovery time) [180]. This work unifies both by adopting a decision-theoretic definition that unifies both: risk is the expected cost of a set of hazards and resilience specifies that this model is dynamic such that consequences (and their mitigation strategies) unfold over time.

The next subsections discuss previous resilience-based design approaches and approaches to consider uncertainty in design used as the basis of the cost-based resilience definition and validity testing approach presented in this work.

2.1.1 Previous Resilience-based Design Approaches

Incorporating resilience in the design of a system is beneficial to ensure it operates as safely, sustainably, and economically as possible when there are hazardous events [169, 106, 53]. Optimizing resilience can involve the physical design of the system (e.g., redundancy and sensors) and the contingency management of the system (how it responds to hazardous events). Prior work [182, 240, 20, 202, 303] has shown both of these resilience optimization strategies in the design of aerospace systems. For example, the resilience optimization of a health management system in a power balance model of the Space Shuttle Main Engine involved the allocation of sensors needed to detect a hazard [182]–an instance of optimizing a *design* for resilience. Other work has focused on optimizing the resilience of in-flight aircraft fault accommodation (e.g., trim, throttle, and switch adjustments for various hazards) [240]–one of many instances (see: [20, 202, 303]) of optimizing the *contingency management* for resilience. These approaches complement each other, since contingency management must have some inherent flexibility afforded to it from the design features to operate effectively and resilient design features must likewise be leveraged effectively by contingency manage-

ment to effectively mitigate hazards. Designing resilience into a system thus can and should involve optimizing both its high-level design and operational features and its contingency management.

As discussed in Section 3.2, there are a number of competing definitions for resilience in the literature, which in turn effect how that resilience is incorporated in design. To enable the trading of resilience with performance and other considerations, many design frameworks have been put forward which quantify the value of resilience in the overall cost function used to assess the merit of designs [174, 306, 115] (though other approaches exist, see: [265, 260, 177]). Value-of-resilience design frameworks are advantageous because they, as value-driven design frameworks, provide an objective which can be used by an optimization method to find the most resilient set of parameters for a system and can incorporate the designer's valuation of uncertainty [300]. Thus, many optimization frameworks and applications use an overall life-cycle cost function as the overall objective, which incorporates both the cost of resilience and the cost of performance and design considerations [293, 108, 307, 138, 176].

There have been some previous efforts in resilience-based design to optimize the contingency management policy of a system together with its physical design and operations. Mehr et al. [187] proposed the unified optimization of a mission's prognostic and health management, asset design, and asset management as three disciplines in a multi-objective multi-disciplinary optimization of mission price, vehicle weight, and launch availability. Since then, major works in resilience-based design have theorized the general early-stage resilience allocation problem as a sequential problem: first allocating resilience targets to subsystems and then optimizing the reliability and health management of those systems to achieve the targetted resilience [306, 305]. Additionally, in a prognostics and health management application, Niu and Jiang used a sequential approach, first optimizing the "local" usage profile of a braking system to reduce maintenance and then optimizing the "global" maintenance policy for the brake to minimize the maintenance cost-per-braking-distance, finding that optimizing both resulted in less overall cost than only optimizing one or the other [206].

Finally, there have also been examples of bilevel and two-stage (a related architecture, see Section 2.3) architectures for use in resilience optimization. Two-stage approaches are often used in

the resilience-based design of infrastructure networks [67, 195, 184], though they have recently been introduced in a general systems design context [300]. Similar to the bilevel approach, the two-stage approach optimizes the system to prepare for adverse events in the first stage (upper level) and optimizes the system to mitigate and recover from adverse events in the second stage (lower level). The difference between the two-stage approach and the integrated resilience optimization formulation pursued in this work is that a two-stage formulation assumes that one can optimize the second-stage variables after the hazard has occurred, rather than relying on a predetermined policy (as is done in this work). More recently, Zhang et al. have used a bilevel optimization approach to determine the optimal construction and capacities of service centers that maximizes resilience in the case of service center disruptions [308]. One goal of this work is to compare the merit of bilevel approaches like this (e.g.,[308, 229]) with the sequential approaches previously (i.e., [306, 305]) presented for the general design of resilience.

2.1.2 Design Decision-making and Consideration of Uncertainty

The framework presented in this dissertation uses the principles of decision theory to systematically consider the choice of design approaches to achieve resilience. Decision-based design views engineering as a decision-making process and uses the principles of decision theory to structure and inform this process [90]. These frameworks often rely on the axiomatic definition of utility presented by Von Neumann and Morgenstern in *Theory of Games and Economic Behavior* [285]–seeking to maximize the statistical expectation of the utility of a design [278, 90, 91]. This utility is often calculated directly as a profit value (as in [81]), but may also be a function of a profit value, when different profit levels result in different marginal utilities [285]. One of the major advantages of value-based design frameworks is that they produce an objective which can be used to optimize a design, rather than relying on fixed requirements, thus avoiding cost growth due to missed requirements [50].

Decision-based design approaches risk in design as a lottery. To illustrate, if a design x could lead to only two mutually-exclusive outcomes, 1 and 2, with utilities $u_1(x)$ and $u_2(x)$, and outcome

1 has probability $p(x)$, the expected utility $U(x)$ of that design is:

$$U(x) = p(x) * u_1(x) + (1 - p(x)) * u_2(x) \tag{2.1}$$

For example, a design where $p(x) = 0.5$, $u_1 = 200$, and $u_2 = 0$ is equivalent to a design where $p(x) = 1.0$, $u_1 = 100$, and $u_2 = 0$, since both have the same expected utility of $U = 100$. Alternatively, a design where $p(x) = 1.0$, $u_1 = 100$, and $u_2 = 0$ would be preferred to a design where $p(x) = 0.9$, $u_1 = 100$, and $u_2 = 10$, since it has a higher total utility ($100 > 91$).

In the decision-theoretic understanding of the design process, the designer progresses through a successive set of choices to either analyze designs or select the best design based on current information [248]. The value of analyses (e.g., of resilience) is to reduce the preference uncertainty between different design options associated with low information. A variety of approaches have been presented to value this uncertainty to aid decision-making. Bradley and Agogino [34] first introduced Expected Value of Perfect Information (EVPI) (see Eq. 6.5) to the design field as a metric to decide whether or not to reduce uncertainty or make a given component choice decision in the Intelligent Real Time Design methodology (IRTD), which was later incorporated in the Decision-Based Conceptual Design framework [298]. In this context, a low EVPI would mean there is little value to be gained by waiting to make a particular decision because knowing more information would not lead one to choose a better design, while a high EVPI would mean the design should be characterized better before making that decision. Takai and Ishii [272] subsequently used EVPI to determine how to allocate resources to reduce uncertainty in concept selection, and Hsiao et al. [109] extended the EVPI using a risk attitude transformation to account for the designer's preference for more or less uncertainty. The main problem with using EVPI to value analyses is that it assumes the designer eliminates uncertainty entirely, which is the upper bound on the value of seeking more information. Nevertheless, it is an appropriate method of determining the value of analyses in early design, since the metrics that correct for this issue, such as the process performance indicator [193, 192, 191], expected value of sample information [87], and expected value of imperfect information [277, 276] assume a certain level of uncertainty reduction from the information which may

not be known. In these situations, where the largest contributor to uncertainty is reducible epistemic uncertainty and one wishes to keep analysis simple, EVPI is considered most appropriate [62] since the efficiency of information is high.

2.2 Modelling Approaches

Table 2.1: Comparison of Model-based Fault Simulation Toolkits for Design

Toolkit	**Behavioral/Causality Representation**	**Model Input/Platform**	**Availability**	**Use in design**
HiP-Hops [220]	Dynamic simulation with failure logic	Simulink, SimulationX, AADL, etc.	Commercial	Functional Hazard Assessment, Design Optimization [221]
Rodon [37]	Behavioral constraint network with failure logic	Modelica-like Rodelica model	Commercial	Model-based engineering process [172]
Modelica fault libraries [282, 196, 82]	Undirected behavioral/failure logic	Modelica	Open Source	Design exploration [163]
OpenErrorPro [198]	Probabilistic markov chain	Simulink/Stateflow, UML/SysML, and AADL	Open Source	Model-based reliable system design [199]
SHyFTOO [44]	Dynamic simulation with probabilistic hybrid fault tree	Simulink and toolkit-defined model	Open Source	Model-based design [45]
OpenCossan [227]	Probabalistic semi-markov transitions, external simulation	MATLAB toolkit-defined model	Open Source	Resilient, reliable, robust design under uncertainty [226]
IBFM [185]	Bond graph with failure logic	Text-based model, Python	Open Source	Functional decomposition and design optimization [115]
fmdtools	Dynamic un-directed behavioral simulation with failure logic	Python toolkit-defined model	Open Source	Early resilience-based design, analysis, visualization and optimization

Fault propagation is widely used in a number of domains to assess the safety and resilience of a system of interest. As shown in Figure 4.1 in Chapter 4, while most current tools focus on modelling the system in the later design stages and in the verification and validation process, when there are detailed models of the system, fmdtools–the simulation framework presented in this work–is meant to support early design processes. To do this, it provides analyses that support each phase of the function-behavior-structure design process commonly used in engineering design![107]. In this process, one creates a functional decomposition of the tasks the system is to perform, finds solution principles to achieve those tasks, and then synthesizes those principles into a realized design concept [217]. These design processes are supported by static failure-logic functional hazard assessment

and network models, dynamic behavioral models, and hierarchical fault models, respectively.

Additionally, a number of other modelling formalisms have been developed to assess the risk of hazards in engineered systems, including fault trees, bayesian networks, and stochastic petri nets (a type of discrete event simulation) [39]. While the simpler methods (fault trees, bayesian networks) enable stronger proofs of system dependability [39], they do not express the system resilience–the behavior over time resulting from a fault. In these situations, a discrete event simulation or continuous dynamic model is used. Discrete event simulation (e.g. [181]) has been used to assess and design resilience into systems, including maintenance [295] and recovery aspects [194]. Similarly, Monte Carlo techniques are often used with continuous simulation models to quantify risk of hazardous states [110]. While these approaches describe the underlying general approaches to simulating faulty behaviors in a system, the following sections describe specific toolkits that use these approaches to quantify resilience (2.2.1), related safety and reliability-oriented fault modelling toolkits (2.2.2), work regarding function-based fault models used as inspiration for the fault modelling class structures used here (2.2.3), and types of fault models used in probabilistic risk assessment (2.2.4).

2.2.1 Resilience Modelling Toolkits

As shown in Table 2.1, there have been some prior efforts to develop generally-applicable toolkits for the design of resilience in a model-based engineering process. One major difference between toolkits is the way they represent causality in the system to determine the effects and propagation of faults, which has a number of potential aspects, including the types of failure paths able to be represented (through the system behavior, failure logic, state transitions or a hybrid) [104], and the nature of causality (probabilistic or deterministic). To integrate with design activities, currently-available frameworks are built around either a standardized modelling language (e.g. AADL), or an existing systems modelling tool such as Simulink or Modelica. While this enables one to use a single unified model for multiple analyses through the process, it can also be limiting if certain

aspects of fault propagation are difficult to represent in the given formalism. Finally, while each of these toolkits are claimed to support different design processes, the design being performed is later stage embodiment design–not the early conceptual design shown in Figures 4.1 and 3.1. The main exception to this is Hip-Hops (Hierarchically Performed Hazard Origin and Propagation Studies) and IBFM (Inherent Behaviors of Functional Models), which both target early design processes. As a result, the implementation of early functional risk assessment tools has been cited as a necessity to bridge the gap between model-based hazard assessment methods in literature and their adoption in industry [80].

Development of the fmdtools toolkit presented in Chapter 4 was initiated to address limitations with the previously-developed IBFM simulator [185]. While IBFM was developed for the early, high-level functional design of engineered systems and was able to determine the end-states of large sets of system faults, the behavioral representation was restricted to a few given states (Zero to Highest), a set of given possible operations on input and output flows, event-based dynamic propagation, and a given syntax for specifying failure logic. This limited its ability to express all possible behaviors that could occur in a system as a result of a given fault. Additionally, the text-based model representation made it difficult to parameterize models and simulate them iteratively in an optimization algorithm. In Section 4.2.1, these limitations are addressed by defining the model as a set of interacting Python classes with possible faults and (nominal or faulty) behaviors to simulate over a set of discrete time-steps. With this model representation, one can easily express and optimize the dynamic behavior of the system without being limited by expressiveness of the underlying modelling tool. The fmdtools toolkit has additionally since expanded in scope to include not just fault propagation tools but the necessary analysis and visualization tools needed to interpret simulation results. In doing so, it comprises an early resilience-based design environment that enables quick, iterative analysis over a design model.

2.2.2 Related Fault Modelling Tools

In addition to the fault propagation toolkits mentioned above, there are additionally a number of toolkits used for fault propagation in safety assessment and for resilience assessment in specific applications (e.g., circuits, infrastructure) that merit a specialized simulation framework.

Safety assessment toolkits are used to verify that a given design meets desired safety and reliability criteria [104, 136]. Typically, these tools take a design model specified in a formal language (e.g., Simulink [134], Lustre [135], Modelica [282], AADL [268], UML [52]) and apply a model checker to the system description to find cases where the system does not work as intended. While this process can be performed in a nominal system model, model-based safety assessment tools extend this process to assess safety by including failure modes and faulty behaviors in the system description [134, 135]. However, because the intended purpose of these tools is to verify safety requirements post-design (see the right side of the V-model in Figure 4.1), they typically rely on detailed, fully-specified models of the design that are not available early in the design process [80]. Furthermore, these tools are not intended to assess resilience–the dynamic effects of failures due to faults, but instead assess the safety or reliability of the system (i.e., an overall fault tree or failure probability). As a result, they are not applicable to the design of resilience in the early design process when the system has not been fully specified and the designer is interested in assessing the system's dynamic response to faults.

Fault injection is used widely in software and hardware engineering to assess a computer system's ability to manage the different types of faults. Existing simulators vary by domain (e.g., distributed systems [179], servers [149], stand-alone systems) and type of faults (hardware-originated [76, 246] or software components/algorithms [78, 13]), though generic simulators also exist [296]. One of the advantages of software (as opposed to hardware/systems) engineering is that there is little comparative cost to implementing software prototypes, and as a result, these tools do not operate on models of the system as is needed in engineering design but real prototypes.

There are additionally a number of application-specific resilience assessment toolkits that interface with specific system simulators to model the resilience-related attributes for that domain. For

example, integrated circuits have well-defined properties that have led to a number of specialized fault simulation algorithms, which have since been implemented in software [183, 205]. Infrastructure often has specific hazards to assess, which has resulted in tools to consider natural disasters for cities [72, 186] and cyber-physical threats in smart grids [287]. Finally, assessing the hazards of autonomous vehicles in a real system is both costly and hazardous [74], which has led to the development of a number of simulators that enable one to try different policies to approaching hazards which the vehicle will encounter [132, 133]. While these toolkits can assess resilience in specific design contexts with high fidelity, they are not applicable to a generic systems design context. The fmdtools framework presented in Chapter 4, on the other hand, is meant to for early design stage resilience modelling, where it is more advantageous to use a general-purpose framework to prototype a large space of potential systems at a high level.

2.2.3 Function-based Fault Models

As discussed in the previous sub-sections, most existing fault/failure modelling tools and approaches have been developed around late-stage design models. Thus, previous work involving the use of fault models in early design have developed function-based fault models which use the functional model of the system to represent the system interactions. Functional modelling is a way of representing the concept of purpose in a system which has been described as a language for conceptual design intention, a bridge between high-level decision-making and implementation [64], and a "blueprint" for the future system which is agnostic of any particular form [270]. While a variety of modelling conventions have been presented in general, function modelling represents a system as a set of functions which act on flows of energy, material, and signal to accomplish a given task [64]. This specific representation of functionality is one of many potential system representations, but is uniquely useful for its lack of ambiguity and ability to be reused and transformed to simulate behavior [153]. It is also a major part of engineering design curriculum [280, 217, 281] and has been standardized as a part of the systems design process [297, 128].

Functional decomposition is meant to abstract the system in such a way that allows the designer to consider a variety of creative solutions that provide the necessary functions required by the system [217]. In general, functions are operations stated as noun-verb pairs which are performed temporally on the material, energy, and signal flows that pass through the system. Decomposition follows the following process:

- Developing an overall black-box model of the main function of the product
- Determining the primary set of functions required to achieve the overall function, and connecting the flows of energy, material, and signal proceeding from one task to another
- Adding the supporting functions needed to provide the flows needed by the main set of functions

Often, functional decomposition follows a hierarchical process and representation as shown in Figure 2.1 for an evtol aircraft. That is, this decomposition continues throughout the design process from a high-level representation of the tasks performed by the system determined at the outset to lower level decompositions determined based on the concept and configuration chosen for the system. For example, in this system, the high-level task "Manipulate Environment" is performed by (in the horizontal flight mode shown) achieving forward velocity and using that velocity to achieve lift and control the direction of movement and (in vertical flight mode) by varying the torque and thrust generated by several vertical propellers.

In the functional decomposition process illustrated in Figure 2.1, fault models can be developed which determine the effect of functional failures on the overall system. These models constitute a model-based Functional Hazard Assessment and can be used to then compare different functional architectures and individual function requirements using assumptions (to be given as requirements) about fault rates. A variety of approaches can be used to develop function-based fault models, including using graph-based models [158], hierarchical dynamic system models [220], dimensionless behavioral models [48], and state-based models [130]. In each approach, behaviors are associated with each function and a list of scenarios is simulated in which faults are injected into model in

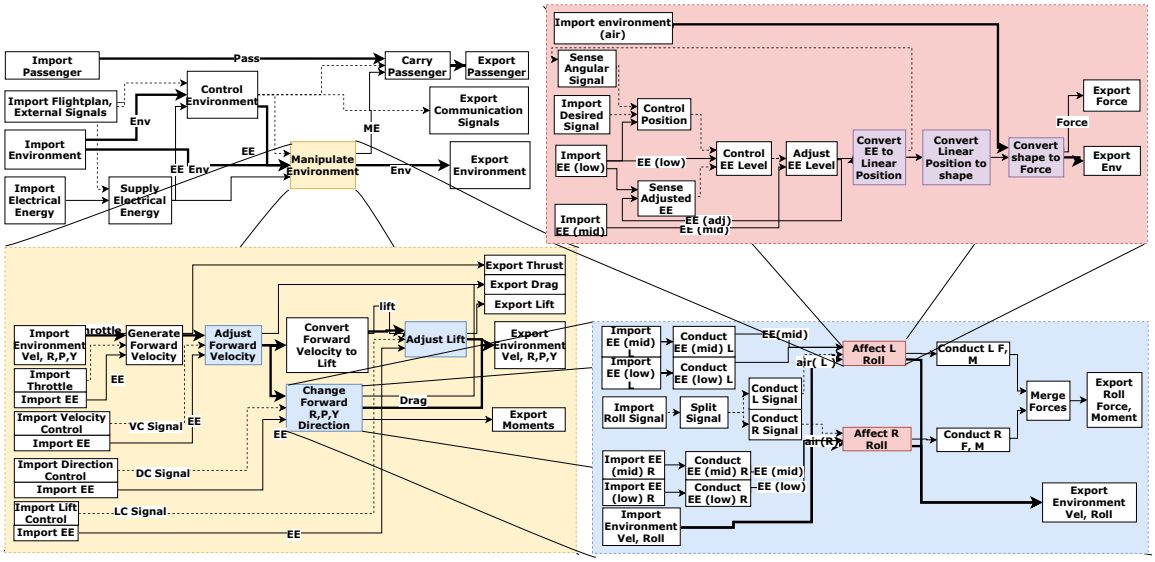


Figure 2.1: Decomposing the functions of an EVTOL Aircraft from the high level tasks which must be performed to the sub-functions needed to perform those tasks. Decomposition occurs after one step of design is completed to motivate the next step, starting with the task clarification model (upper-left) which motivates the conceptual design (lower-left) and then the embodiment design (lower and upper right). As such, the final level of decomposition (seen in the upper-right corner) maps directly to the components of the system.

each function (or a set of functions) and propagated through the model to observe the consequences. Depending on the depth of insight desired from a function-based fault model, behaviors may be coded as states in the function (nominal, degraded, or failed) or from the dynamic equations related to each function.

To incorporate functional hazard information in design, prior work introduced formal methodologies to enable designers' use and understanding of the information. 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 [269]. This was extended in the risk in early design method (RED) using likelihood and consequence estimates to better inform designers [171] [170], which has since been shown to better allow students to assess risk [124]. To analyze social and organizational hazards in engineering systems, the Functional Resonance Analysis Method (FRAM) was developed to analyze hazards within socio-technical systems based on high-level functional re-

lationships [102], such as air traffic management [59]. This socio-technical interaction has also been analyzed using a hierarchical functional decomposition of a system to identify hazards within process plants [239] and food processing [238]. These tools are based on building tables or databases of previous faults which occurred within functions, as well as their effects concurrently with a functional model.

Other approaches use function-based computational models to encode risks. Hierarchically Performed Hazard Origin and Propagation Studies (HiP-HOPS) was developed as a tool to assess risk throughout a hierarchical system model by integrating functional and classical techniques into a single consistent model [220]. Rather than modeling failures within engineered systems as a result of component failures, the Systems-Theoretic Accident Model and Processes (STAMP) models the dynamics of the organizational environment to find and redesign inadequate control processes that lead to failure [203] [162] [61] [127]. The function failure identification and propagation method (FFIP) informs assessment by constructing a graph-based behavioral model to take into account the function interactions, dynamics, and joint fault scenarios [156] which has since been extended using flow-state logic to model undesired flow states [130] and dimensional analysis to incorporate more detailed information about component behavior [48] and adapted for large-scale complex systems [223] and mechatronics [258]. Inherent Behavior of Functional Models (IBFM), which is used in Sections 3.3,6.4, and 5.4, provided a method to automate the creation of a state-based behavior model from the functional model itself [185]. Approaches have additionally been presented which associate the functional model with a fault tree [168] [208] [190], and other methods have been created to focus on the propagation of failures through a functional model [152].

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 simply used to show designers where attention should be paid in making design choices [269]. Approaches 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 [143] [142]. Additionally, an approach has been presented for designing the operational

decision-making in the model to determine when to, for example, route degraded flows to sacrificial subsystems [256]. While these approaches show many of the design changes that can be made within a functional model, and can be used to compare between design alternatives, they do not use this knowledge to formally optimize a design problem.

This work is somewhat novel in that only a few approaches for formal optimization of risk within functional models have additionally been presented. Using the HiP-HOPS risk modelling methodology, design optimization [221] and optimization of fault tolerance [3] has been demonstrated. Additionally, using functional-failure-matrix approaches, the Risk and Uncertainty Based Concurrent Integrated Design Methodology (RUBIC) first introduced the concept of using failure scenarios, probabilities, and costs to optimize risk within a functional model to show where to allocate resources based on function-failure matrices used in FFDM [188]. Additionally, objectives been formed for the allocation of health management within a functional model informed by the effect of adding sensors, reducing failure probabilities, and changing inspection intervals on reducing overall design risk [108]. While these approaches show that some optimization has been performed in the context of function-failure methodologies, no approaches have yet been presented for the graph-based fault models, such as IBFM and FFIP.

2.2.4 Fault Modelling in Probabilistic Risk Assessment

Fault models are used in Probabilistic Risk Assessment to calculate or simulate the probabilities and/or severities of hazards [16]. However, not all fault modelling approaches can be used for resilience quantification using the definition used in Section 3.4, since it relies on a specific representation of time and probability. A review of fault modelling approaches is provided in Refs. [4, 83, 104, 39, 137]. Generally, time representations in fault models vary between completely static simulations (where faults result in immediate effects with no consideration of time e.g. [218]), discrete time simulations (where time is incremented by a set timestep e.g. [6, 43, 110, 86]), and continuous dynamic simulations (where faults propagate through a model with the dynamics of the

system fully incorporated e.g. in a Simulink model [23, 301, 93]).

Fault modelling approaches can additionally be placed into probabilistic and deterministic categories. In a deterministic model, the fault-induced behavior of the system (which may be modelled in a finite state machine, a series of logic gates, or a physics model) is considered a direct result of the fault and thus the same fault scenario always produces the same result. In a probabilistic model, the dynamics of failure incorporates behavioral stochasticity, and thus the outcome of a given scenario is a distribution or event tree [110] of fault effects [7]. Modelling uncertain behavior in system fault models is an open research area, but common approaches include creating a markov model of the probabilities of transitions between different system states [6], querying the model with an event tree [110], creating uncertain distributions for failure parameters for use in Monte Carlo sampling [195], and similar approaches [160, 309].

Finally, fault modelling approaches can be categorized as inductive or deductive based on the way they represent the set of fault scenarios and consequences [301]. Fault models are often used for safety assessment to quantify the overall probability of a given top event that constitutes a safety risk. In this deductive approach, called backtracking, a tree of scenarios is searched to find the probable scenarios that lead to the top event–creating the fault tree for that event [301]. This is contrasted with an inductive approach, where the list of fault scenarios is simulated to determine the set of effects that occur for those faults–creating what is essentially an FMEA [80]. For safety assessment, the deductive approach is preferred because it gives one a comprehensive quantification of safety risk from all identified sources [267, Sec. 4.8].

2.3 Optimization Architectures

The complexity inherent to optimizing large-scale and multi-disciplinary systems has lead to the study of distributed optimization architectures in the field of multidisciplinary design optimization (MDO) [178]. Multidisciplinary Design Optimization (MDO) as a field emerged from the challenge of designing multidisciplinary aerospace systems, where the many subsystems and disciplinary analyses

make it advantageous to use local incentives to optimize subsystems in a way that aligns with high-level objectives [261]. While initial approaches focused on aligning these objectives, early tests of these approaches found that it is also necessary to coordinate the subsystem interactions, otherwise the system will not be fully optimized [252]. Thus, multidisciplinary design optimization approaches focus not just on the alignment of incentives but the coordination of subsystem optimizations [35]. Because aircraft design is a multidisciplinary design problem like this, with multiple models and coupled objectives, interest in the area has resulted in application tools for real problems [92, 79, 47] and frameworks for integrating MDO in design [231, 222].

Multidisciplinary design optimization approaches can be classified based on their problem formulations and decomposition strategies [178]. In distributed formulations of the multidisciplinary design optimization problem, the optimization problem is divided into several interacting optimization problems to be solved by individual algorithms, while in monolithic formulations one single optimization algorithm is used on the entire problem. Distributed formulations are additionally categorized as distributed individual-discipline-feasible (IDF), where individual sub-problem optimizations must optimize coupling variable consistency in addition to sub-system objectives, or multidiscipline-feasible (MDF), where consistency constraints are enforced between subsystems through the statement of the problem [178]. Optimization architectures are also put into hierarchical and non-hierarchical categories, where hierarchical architectures, such as Analytical Target Cascading, have a number of "levels" at which the optimization is decomposed (system, subsystem, sub-sub-system, etc.) while non-hierarchical architectures such as Collaborative Optimization have only two levels: a system-level and disciplinary-level optimization [9].

Unfortunately, the development of a distributed optimization architecture that consistently converges across a wide range of problems is still an open problem, and distributed approaches often have increased computational cost compared to monolithic formulations [178]. Nevertheless, there are differences between each architecture that may make each more or less favorable for a particular problem. For monolithic architectures, IDF formulations can make optimization more difficult by increasing problem dimensionality but faster overall by enabling parallel execution of models [9]. IDF

architectures may also find parts of the design space that would have been inaccessible in an MDF formulation, which may lead to increased solution quality depending on the given problem [274]. Finally, while architectures can increase solution difficulty, the heterogeneous design spaces often approached in complex systems design make it necessary to break the problem into components for ease of solution [9].

Bilevel optimization architectures are often used in multidisciplinary design optimization for large scale complex systems when the optimal decision in one model depends on the optimal decision in another model.

Bilevel optimization was first realized in the field of game theory by economist Heinrich Freiherr von Stackelberg in 1934 [286] that described this hierarchical problem, where one problem–the lower level-level problem–is embedded (nested) within another problem, called the upper-level problem, acting as a constraint. To solve the problem, each iteration of the optimization of the upper level corresponds to a full optimization of the lower-level problem at a specific set of values used in the upper level problem [28]. While these approaches are applicable to certain classes of problems such as competitive games where one wants to know the best strategy to defeat the opponent's best strategy [51], computational cost is an issue when they are used on traditional problems instead of a monolithic approach [28]. Nevertheless, Stackelberg games have been used to represent complex robust design problems where one wishes to design the system to be as robust as possible to adverse events, because these events are analogous to the lower-level "adversaries" in traditional Stackelberg games [243].

Papers have been published attempting bilevel programming in various complex design problems. This approach has been extensively applied in the field of transportation and defense strategy. Labbe, Marcotte and Savard in 1998 proposed a bilevel model of taxation and its application in toll-setting problem in highways. In this bilevel model the leader wants to maximize revenue from taxation schemes, while the follower rationally reacts to those tax levels [159]. Chen and Subprasom [40] formulated a stochastic bilevel programming model for a Build-Operate-Transfer (BOT) road pricing problem under demand uncertainty. Braken and McGill [33] proposed a bilevel optimization model

in defense applications which includes strategic force planning problems and two general purpose force planning problems. In recent years, this approach has been accepted and is being widely used in strategic bomber force structure and allocation of tactical aircraft to missions. Roghanian, Sadjadi and Aryanezhad [244] presented a bilevel multi-objective programming model in enterprise-wide supply chain planning problems considering uncertainties on market demand, production capacity and resource availability. Recently, Biswas et. al. [26, 27] developed a bilevel flexible robust design on complex large scale multi-reservoir system under risk of future shortage of energy, which provides better optimal decision (higher revenue) than a standalone framework. Since bilevel optimization is computationally expensive, mathematical and dimension reduction approaches have also been taken in [26] to reduce the computational cost at minimal loss of information.

Two-stage Optimization is an architecture used in mathematical optimization to consider uncertainty in decision-making [139]. Two-stage optimization was introduced by Dantzig [57] and Beale [22] as a way of solving linear programming problems with stochastic variables. In these formulations, the first stage optimizes the decisions available before the uncertain variable is known, while the second stage optimizes the decisions available after the uncertain variable is realized. However, since taking the uncertainty into account in the first stage requires taking into account the second-stage actions, a recourse function for the optimal second-stage response must be provided. This recourse function is the optimal resilience cost given realized values of the uncertain variables, which can either be derived analytically and/or approximated using a discrete set of scenarios (stochastic programming problems of this type are usually formulated to be linear) [235, 253]. Thus, where a bilevel approaches might find the optimal policy over a space of uncertainty (e.g. finding the best design and operational policy over the set), two-stage approaches find the best policy before and after uncertain variables have been realized. However, inherent to this framework is the idea that the second-stage variables can or will be optimized after the fact. For engineered systems, this may not necessarily be the case: while a PHM system might be able to identify a fault, it may not necessarily be able to optimize the response to the fault in real time, instead using a predetermined contingency management scheme. Thus, Chapter 5 focuses on the optimization of that policy before

the uncertainty is realized (where all variables are optimized together), rather than after (where two-stage approaches are applied).

Recent developments in the field of multidisciplinary design optimization have additionally considered the unique situation of combined plant and control design [10]. Codesign is the special application of multidisciplinary design optimization to the simultaneous design of the plant (or design) of a system and its dynamic control policy or architecture which is used when control and design considerations are coupled. Codesign can be performed in a nested (bilevel) or simultaneous (all-at-once) architecture [96], and has been used in a number of applications for combined plant, control system, and architecture optimization [97]. For example, codesign has previously been applied in aircraft design to the aerodynamics and control of tail-fin controlled supersonic missiles, where the choice of optimal tail planform is coupled with the choice of optimal tail control profile [164]. This situation is similar to the resilience optimization problem, except that in the resilience optimization problem the control of the system must be optimal not only over the nominal scenario, but over the set of fault scenarios. As a result, this Chapter 5 considers an optimization problem where not only is the optimization the system design and operational profile considered, but also the optimization of resilience policy variables.

Chapter 3: Defining Resilience as a Decision-Theoretic Objective

3.1 Motivation

Reducing the impacts of hazards is a key part of the design process. When the consequences of hazards are salient, it becomes important to proactively avoid them through good design, rather than attempting to mitigate them post-design, since a hazardous design outcome constitutes project risk [242]. Up-front early-stage design decisions lead to large increases in project cost if made poorly, since they require more effort to correct later in the design process [273]. As a result, it is important for early high-level decisions to be made strategically to prevent the rest of the design process from being stifled by poorly-considered early design commitments [70].

Thus, the design of resilience in the system needs to occur in the early design phase, when analysis can have the most influence over the final design. In order for the analysis of resilience to be taken into account in early design, it must factor into design decision-making processes, as described in Section 2.1 and summarized in Figure 3.2. In this process, the design problem is defined, concepts investigated and refined, and then a decision is made between the different concepts which defines the requirements and specification that guides the rest of the design. While many frameworks to conduct this early design decision-making exist, as described in Section 2.1.2, cost and utility-based

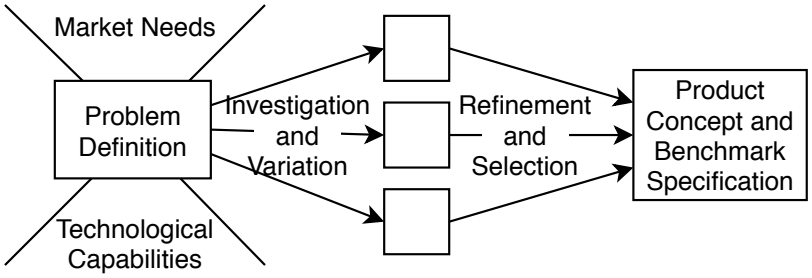


Figure 3.1: The early design process.

design processes have been adopted by the design community because of their decision-theoretic properties (i.e., internally consistent axiomatic basis) and intuitive salience. Thus, in the design of complex prognostics and health management systems used to increase resilience, a financial "case" is often provided where the up-front design costs are weighed against the operational, maintenance, and safety benefits they will provide in the final design [68, 154].

The objective of this chapter is to develop expected cost metrics which quantify the cost of fault simulation results so that the benefit of resilience can be used to inform design. As a single, holistic design measure, these metrics (along with design and operational costs) can then be used as the sole value consideration both to guide early design processes and to be used as an objective in optimization procedures. Using expected cost in typical systematic design process (such as that described in [217, see Chapter 2] and Chapter 2), designers can compare a variety of early functional design concepts and determine how best to make key design decisions about function structure, high-level functional requirements, and solution working principles. Using the cost metric for optimization, solution procedures may be developed and leveraged which allow designers to explore large spaces of design concepts in a way that would be difficult or impractical without computational support (see Chapter 5). For example, using expected cost as the objective, an algorithm could search extensive design catalogues for compatible solutions for each of the given functions of a model, a process that would be painstakingly tedious for a team of designers to perform. Since the assessment of resilience is often performed in a model (see Chapter 4), this work additionally aims to quantify benefits from simulation results. To advance these goals, this chapter presents:

- in Section 3.2, a general decision-theoretic definition of resilience as an expected cost;
- in Section 3.3, an adaptation of expected cost to quantify resilience using simulations developed in the IBFM [185] modeling framework; and
- in Section 3.4, an adaptation of expected cost to quantify resilience based on dynamic simulations developed in the fmdtools fault modelling framework presented in Chapter 4.

To demonstrate these methods, Section 3.5 first shows an simple example of how expected cost can

be used as a part of the overall early design framework of this dissertation in a design problem considering the selection of a wire. Section 3.6 then demonstrates the adaptation of expected cost within the fmdtools modelling framework presented in Chapter 4 and compares the expected cost injection error of a variety of fault sampling approaches which may be used in this framework. Demonstration of the IBFM RISCS scoring in 3.3 is additionally provided in the examples in Sections 5.4 and 6.4. A brief conclusion is then provided in Section 3.7.

The reason multiple adaptations of expected cost scoring are provided here is that different modelling frameworks (or no modelling framework) may be chosen to represent system resilience, depending on the needs of the problem under consideration. While models encourage design consistency and rigor in specifying assumptions, they also take significant resources to set up. Additionally, while the fmdtools fault modelling framework developed in Chapter 4 was meant to address limitations in the IBFM toolkit [185], there are certain use-cases where the IBFM toolkit has an advantage, such as if one wishes to specify a model in text or run simulations without installing the fmdtools dependencies. Since there is still a need in each situation to have a metric which quantifies the cost of resilience for use in decision-making, cost metrics are provided for both.

Finally, rather than just exemplifying the use of the cost model for fmdtools dynamic fault simulations, Section 3.6 additionally assesses the error from using different fault injection strategies for expected resilience quantification. To aid the application of expected cost modelling in fmdtools simulation, where it is intractable to sample every single fault scenario due to computational cost. Computational cost is a major obstacle to designing a resilient system, because the dynamic models used to assess resilience are often computationally expensive [213, 310] and a large set of faults must be simulated to adequately quantify risk [310]. From a decision-theoretic perspective, there is a trade-off between running competing computationally expensive simulations and generating and selecting designs [277, 241]: the increased accuracy of running computationally-expensive simulations may not be worth the opportunity cost of not exploring the design space [87]. Thus, to best inform design where simulations need to be computationally cheap enough to enable quick design iterations while remaining accurate enough to make meaningful decisions, Section 3.6 provides a comparison of

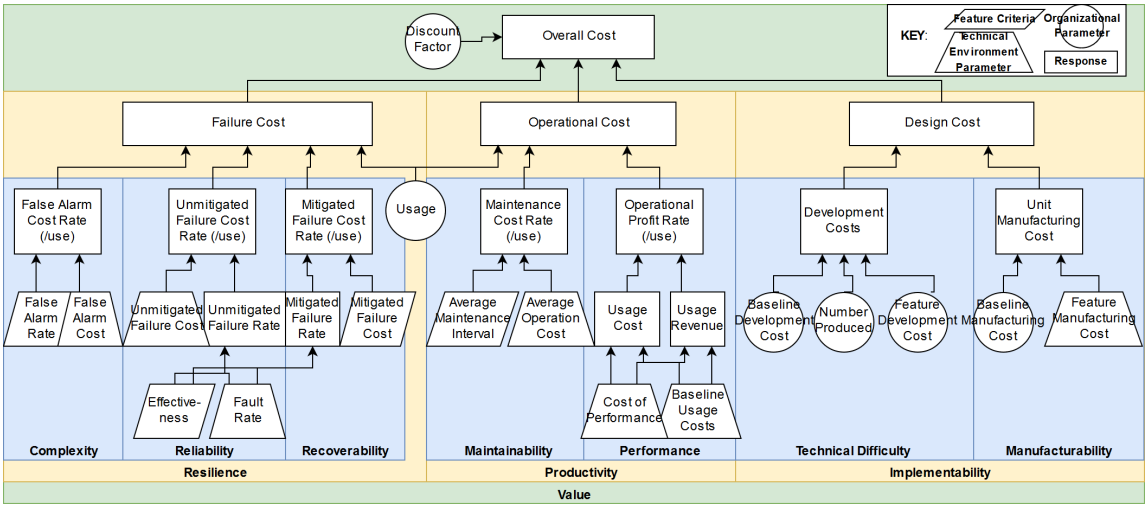


Figure 3.2: Generic consideration of resilience in an overall value model used in early system design.

different injection approaches with differing computational costs which can be chosen by the designer to most efficiently represent the expected cost of faults in the system.

3.2 General Definition

In this work, value assessment is used to consider the trade-offs between the increased resilience of fault-mitigation features such as redundancy, fail-safes, or prognostics and health management and the design and operational costs needed to implement and operate those features. In a generic sense, an overall cost function that considers resilience is:

$$C = C_D + C_O + \mathop{\mathbb{E}}_{s \in S}\{C(s)\} \tag{3.1}$$

where C_D is the design cost (e.g., research and development, manufacturing and integration costs, etc.), C_O is the operational cost (e.g., energy/performance and maintenance costs) and the cost of resilience is the expected value of the cost function $C(s)$ over the set of fault scenarios S. While the respective costs of a system depend on the particularities of the design problem, a generic example

of a value model is shown in Figure 3.2. In this model, all of the positive and negative attributes of resilience (false alarms, unmitigated failures, mitigated failures), productivity (maintenance and operational profit) and implementability (development and manufacturing difficulty) are quantified in terms of their respective overall costs. A cost of a fault scenario $C(s)$ is a result of repairs, performance losses, and disruptions that occur as a result of the fault, essentially:

$$C(s) = C_R(s) + C_P(s) + C_D(s) \tag{3.2}$$

where cost of repair $C_R(s)$ results from damage in the system–the sum of repair costs c_f over the set of faults to repair F,

$$C_R(s) = \sum_{f \in F} c_f \tag{3.3}$$

cost of performance $C_P(s)$ results from the loss of performance $P_n(t) - P_f(t)$ from the fault simulation time t_f to the end of the simulation t_E,

$$C_P(s) = \int_{t_f}^{t_E} C(P_n(t) - P_f(t))dt \tag{3.4}$$

and the cost of disruption $C_D(s)$ is the immediate cost of the system leaving the nominal state and entering a risky or unsafe condition. Sections 3.3 and 3.4 now show how this metric adapted within different modelling frameworks to quantify the expected cost of faults.

3.3 RISCS - Expected Cost Modelling for Static Fault Models

To aid decision-making between different functional models based on fault simulation information in the IBFM fault modelling framework presented in previous work [185], this section introduces the Resilience-Informed Scenario Cost Sum (RISCS), a scoring function which adapts the general definition of expected cost in Section 3.2 to these models. Of particular interest is this function's approach to modeling 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:

$$C = C_D + C_O + C_E \tag{3.5}$$

This scoring is quite similar in form to that devised by Keshavarzi et al. [143] 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 applicability to design factors and integration with function-based fault model definitions. While the scoring function devised previously merely considered the cost of mitigating factors which reduced the probability of end-states [143], the function introduced here is generally-applicable to all mitigating actions and design changes, and is more closely integrated with the results of IBFM simulations, as will be shown in the following sections.

3.3.1 Design Cost

Design costs resulting from design changes depend on the considered design problem. This is because, in general, design costs come from a variety of sources, including research and development, required materials, procurement, manufacturing, and integration. For design purposes, this paper considers that the design costs of a given functional model can be represented as individual costs within each function. As is the approach with risk and failure modes [269], these costs can in turn be estimated based on an organization's past costs for those functions. In this case, the resulting equation for design cost C_D is then:

$$C_D = \sum_{n \in N} C_n \tag{3.6}$$

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

3.3.2 Operation Cost

As with design costs, operating costs must be estimated based on an expectation of the system which may result from past performance or a designer-created parametric model specific to the problem.

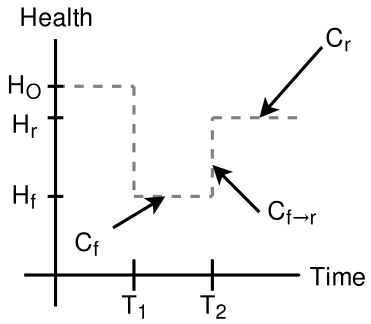


Figure 3.3: Costs associated with a failure event in a resilient system.

While the method for determining the operating costs will vary depending on the considered problem, 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 useful goods created. In general, the operational cost C_O then follows the form:

$$C_O = \sum_{l \in L} C_l - R_l \tag{3.7}$$

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, as will be the case in the case studies, these C_l and R_l terms need not stand for *explicit* costs and revenue in dollars generated, but can also stand for the normative goods, utility, or externalities created and destroyed by the organization. That is, if the organization has a normative goal (e.g., generate science, provide public infrastructure, etc) that does not explicitly lead to more or less revenue, the goods created or destroyed by the organization by pursuing this goal can be quantified and incorprated as if it were a direct cost or revenue.

3.3.3 Fault Scenario Cost

Key to representing resilience in a system is the time-based response to a large number of disruptions or threat vectors [85]. The RISC approach's incorporation of these disruptions is based around IBFM's ability to simulate large numbers of fault events, which are created by initiating a set of fault modes 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 in terms of end-state flow-states and modes, and probabilities, which are determined by the probabilities of the initiating fault modes. The cost of fault events is calculated as an expected cost–the cost of each event is weighted by its probability. The resulting fault event cost C_E follows the general form:

$$C_E = \sum_{e \in E} P_e * C_e \tag{3.8}$$

where P_e and C_e are the respective probabilities and costs of fault event e in the set of considered events E. Assuming the probability of individual fault modes in a scenario are independent, 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:

$$P_e = \prod_{m \in e} P_m * \prod_{n \notin e} (1 - \sum_{m \in n} P_m) \tag{3.9}$$

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 event e occurring, and n is a function that does not have a fault in the event e.

As an expected cost metric, this fault scenario cost bears structural similarities to the Risk Priority Number (RPN) used in Failure Modes and Effects Analysis (FMEA), in that it multiplies the probability of a fault with the severity of the consequences of the fault. Aside from that basic similarity, there are a number of differences between an expected cost metric and an RPN [148]. Unlike the RPN, expected cost uses real probabilities and costs, making it a consistent metric for risk prioritization, and a valid metric to trade off with with design and operational costs. While the criticality rating used in Failure Modes, Effects, and Criticality Analysis is in general a consistent

risk prioritization metric [148], it still lacks the key property of allowing trade-offs with design and operational costs.

To further consider resilience in the IBFM framework, the RISCS approach extends the definitions of expected cost by considering three distinct stages in the system's failure and recovery which in turn map to three distinct costs. These costs are, as shown in Figure 3.3: the cost of failure state C_f, the cost of mitigating or repairing the failure $C_{f\to r}$, and the cost of partial recovery C_r. This definition lines up with common definitions 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 [95]. The resulting fault event cost follows the form:

$$C_e = C_f + C_{f\to r} + C_r \tag{3.10}$$

These cost definitions integrate with IBFM simulations as follows:

- C_f results from the scenario end-state of the associated initiating fault event e,
- C_r results from the scenario end-state of a new fault event simulation, with a chosen set of modes repaired, and
- $C_{f\to r}$ results 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.

This process of running a failure scenario, selecting modes to repair, and running a new fault simulation based on the unrepaired modes is shown in Figure 3.4. It should be noted that this fault re-simulation (and resulting cost) is only necessary for the special case in which only a partial recovery is attempted or possible. When all of the modes are recovered (i.e. a full recovery), no new fault simulation is needed since the end-state will be nominal, resulting in zero partial recovery cost C_r. Alternatively, if it is impossible to repair the modes and/or no recovery is attempted, no additional cost results from recovery or partial repair–instead the costs merely result from the failure state. Calculating each of these costs is discussed in the following sections.

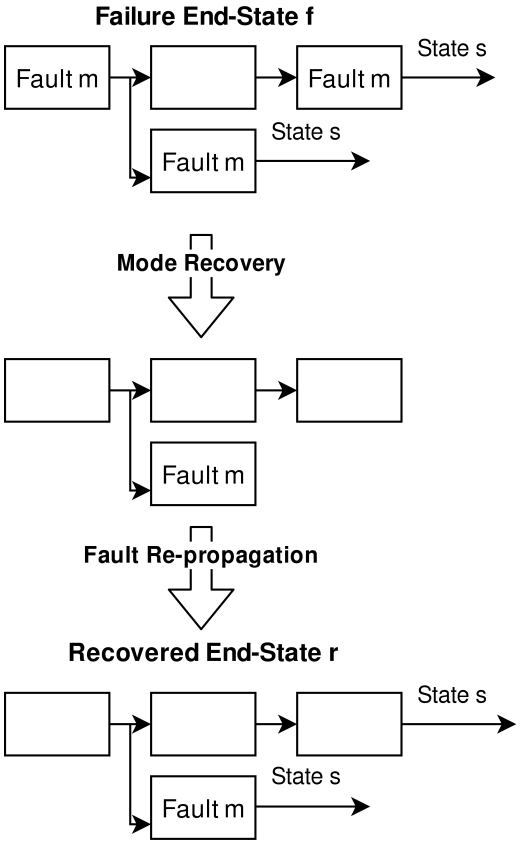


Figure 3.4: Illustration of fault re-simulation required to capture the costs of partial recovery C_r.

3.3.3.1 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. To determine the costs of specific failure events, specific costs must be associated with the flow health 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 3.1. 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 both the severity of the failure state and the amount of time the system is in the failure state. As was discussed in Section 3.3.2, these important flows must be

Table 3.1: Cost rate of an individual flow state for flow l based on combination of rate and effort health states.

Effort Health	Rate Health				
	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	$\bar{c}_l[40]$	$\bar{c}_l[41]$	$\bar{c}_l[42]$	$\bar{c}_l[43]$	$\bar{c}_l[44]$

identified by the designer with costs included.

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:

$$C_f = t_f * \sum_{l \in L} \bar{c}_l[\vec{s}_{f,l}] \tag{3.11}$$

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 of input and output 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.

The recovery time 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 may be 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.

$$t_f = \underset{m \in f \cap \notin r}{max}(t_m) \tag{3.12}$$

3.3.3.2 Mitigation Cost

The cost of mitigation is 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:

$$C_{f \to r} = \sum_{m \in f \cap \notin r} C_m \tag{3.13}$$

As with the other costs, this mode recovery cost C_m can be estimated based on past data or assumptions about the future system regarding the repair/replacement processes required for each function.

3.3.3.3 Partial Recovery Cost

Finally, the cost associated with 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, as shown in Figure 3.4. This partial recovery 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 input and output flows L. This is shown in the equation:

$$C_{r,R} = t_r * \sum_{l \in L} \vec{c_l}[\bar{s}_{r,l}] \tag{3.14}$$

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.

3.3.4 Cost Model Summary

The previous sub-sections discussed how to calculate the costs of a system considering the design, operational, and fault scenario costs using the IBFM simulator, with special consideration of resilience–the ability to recover from failures. While this metric may be calculated differently de-

$$C = \sum_{n \in N} C_n + \sum_{l \in L} C_l - R_l + \sum_{e \in E} (\prod_{m \in e} P_m * \prod_{l \notin e} (1 - \sum_{m \in e} P_m) * (\max_{m \in f \cap \notin r} (t_m) * \sum_{l \in L} \bar{c}_l[\vec{s}_{f,l}] + \sum_{m \in f \cap \notin r} C_m + t_r * \sum_{l \in L} \bar{c}_l[\vec{s}_{r,l}])) \tag{3.15}$$

pending on the considered design problem, when stated as a single expression using the constructions developed in the previous sections, the equation for RISCS takes the form shown in Equation 3.15, where:

- C is the total RISCS
- 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 input or output flow
- R_l is the revenue or utility associated with an input or output flow
- l is a flow
- L is the set of input or output 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

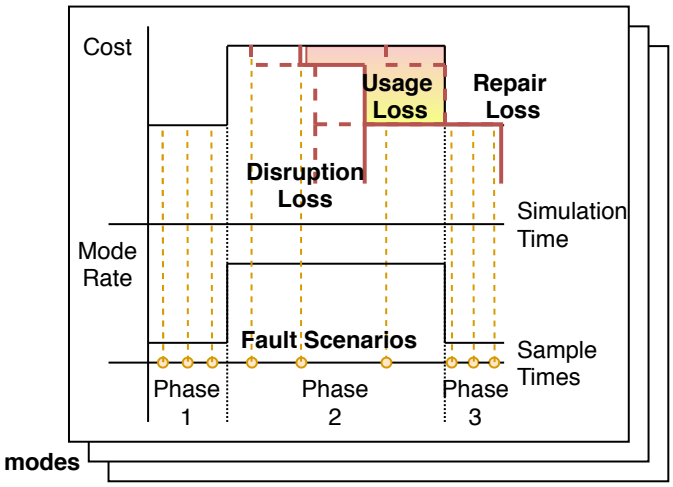


Figure 3.5: Resilience models determine the expectation of metrics over the set of fault scenarios through dynamic simulation.

- 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

3.4 Generalization for Dynamic fault models

Fault injection-based resilience models, such as the fmdtools modelling toolkit described in Chapter 4, quantify the dynamic performance effects of a set of fault scenarios, as shown in Figure 3.5. Scenarios are injected over each phase of operation in the model (in this case, Phase 1, 2, and 3) as well as each mode, with a varying mode rate and time range for each phase. These faults are

simulated at each time to determine the system behaviors and resulting scenario costs. Scenario costs occur as a result of disruption losses (spikes to input/output flows that may cause damage or increased risks outside the system), usage losses (the difference between the nominal performances and the faulty performance), and repair losses (the cost of fixing the damage that occurs as a result of the injected fault) that occur over a fault simulation. The corresponding risk (or expected cost) of a fault is the result of multiplying the simulation cost response with its expected occurrence over the lifecycle of the system. Since the system alternates between different operational modes, both the rates of occurrence of faults and the resulting costs change depending on the time of fault injection. Finally, the total expected value is the sum of the costs of faults. A general statement of this resilience cost C in a dynamic model is

$$C = \underset{s \in S}{\mathbb{E}} \{ \int_{t \in t_s} C(s,t) dt \} \tag{3.16}$$

where s is a scenario in the set of fault scenarios S, $C(s,t)$ is the dynamic cost function (or state of interest) of the system over time t for scenario s, and t_s is the time range of the fault scenario. This definition of resilience integrates with dynamic probabilistic risk assessment process rather than competing with it (as proposed by [18, 224]) because it quantifies resilience (recoverability and robustness of a system over time) in a way that also quantifies risk (expected cost of faults). Thus, to incorporate rate information for a specified set of fault modes, a more detailed statement of resilience cost for this modelling situation is,

$$C \approx \sum_{m} n_l \int_{t_F=0}^{T_E} \lambda_m(t) * (\int_{t=t_F}^{t_E} c_m(t) dt) dt_F \tag{3.17}$$

where:

- n_l is the number of operations over the course of the system's life
- m is a mode
- $\lambda_m(t)$ is the rate density function of a fault mode over the modelled time

- t is time
- t_F is the fault injection time
- t_E is the end of the simulation
- $c_m(t)$ is the cost rate of a scenario per unit time

This equation makes a few assumptions: first, that the reliability is high enough that the rate can be extrapolated out over the life-cycle of the system, second, that all faults are repairable, and third, that the rate density function is constant over the life of the system.

To calculate the cost measure in Equation 3.17 from failure data, a few adaptations need to be made based on what is known by engineers. Often, failure data is presented as a base failure rate for a component λ_c (e.g. from NPRD [60]) which is then distributed between modes (e.g. from FMD [197]) according to the probability distribution $[p_1 \dots p_m \dots p_3]$, as is done in [216]. The rate for each mode is then: $[\lambda_1 \dots \lambda_m \dots \lambda_w] = [p_1 \dots p_m \dots p_w] * \lambda_c$. Additionally, the fault rate of different modes changes based on the operation of the system (e.g. if a component is being used or not, and the amount of use). To account for this, an opportunity vector $[s_{m_1} \dots s_{m_t} \dots s_{m_T}]$ (where $\sum_{t \in T} s_{m_t} = 1$) defines the relative likelihood of a fault occurring in a given phase of operation for each fault. The resulting rates over time are

$$\begin{bmatrix} \lambda_{1_1} & \lambda_{2_1} & \dots & \lambda_{w_1} \\ \lambda_{1_2} & \lambda_{2_2} & \dots & \lambda_{w_2} \\ \vdots & & \ddots & \\ \lambda_{1_T} & \lambda_{2_T} & \dots & \lambda_{w_T} \end{bmatrix} = \lambda_c \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_w \end{bmatrix}^\top \begin{bmatrix} s_{1_1} & s_{2_1} & \dots & s_{w_1} \\ s_{1_2} & s_{2_2} & \dots & s_{2_2} \\ \vdots & & \ddots & \\ s_{1_T} & s_{2_T} & \dots & s_{w_T} \end{bmatrix} \tag{3.18}$$

where $\lambda_{m_{j_t}}$ is an individual rate for mode $j \in w$ for a given phase of time $t \in T$. By representing the rate as constant between operational phases, the integral in Equation 3.17 can be written as

$$C = \sum_{m \in w} n_l \sum_{t \in T} \lambda_{m_t} \int_{t=t_t}^{t_{t+1}} \left(\int_{t=t_F}^{t_E} c_m(t) dt \right) dt_F \tag{3.19}$$

3.4.1 Fault Injection Approaches

To efficiently approximate Eq. 3.19 without simulating every distinct time, a subset of available times is simulated and a weighted average is taken over the set of times in a numerical integration approach, making the resulting cost

$$C = \sum_{m \in w} n_l \sum_{t \in T} \lambda_{m_t} * (t_{t+1} - t_t) * \frac{1}{n} \sum_{n} w_n C_{m_n} \tag{3.20}$$

where $(t_{t+1} - t_t)$ is the time in the phase t, n points to use in the quadrature, w_n is the weighting of a point, and and C_{m_n} is the cost of the mode at point n. A variety of approaches that give these weights are presented and tested in Section 5.5.2, however the following subsections define specialized approaches used in this paper and provide some theoretical justification for their use.

3.4.1.1 A Note on Evenly-Spaced Quadratures

Given Equation 3.17 is linear, there may be a good case to sample evenly, rather than use a specialized quadrature. Suppose the cost rate of each mode $c_m(t)$ is a constant function over time. Then $c_m(t) = c_{m_i}$, making the integral

$$\int_{t=t_t}^{t_{t+1}} (\int_{t=t_F}^{t_E} c_{m_i} dt) dt_F = (t_{t+1} - t_t) * c_{m_i} * (t_E - t_F) \tag{3.21}$$

Since t_F is the time the fault is injected and the fault rate is constant over this interval, $t_F = (t_{t+1} - t_t)/2$, the midpoint of the interval, is the average simulation time and the average simulation cost is $c_{m_t} = c((t_{t+1} - t_t)/2)$, the cost simulated at that point. Thus the full integral is

$$C = \sum_{m} n_l \sum_{i} \lambda_{m_t} * (t_{t+1} - t_t) * c_{m_t} \tag{3.22}$$

Therefore, provided the underlying cost function is constant over the interval, the expected cost of a mode injected over a phase is the same as the cost simulated in the middle of the phase, or the

average of n modes simulated symmetrically about the center.

3.4.1.2 A Note on Likeliest Phase Sampling

The approach with the lowest possible computational expense uses a single point in a single interval to represent all intervals. One approach for this would be to sample the likeliest phase, selecting the phase $t^* = argmax(\lambda_{m_t}(t_{t+1} - t_t))$. The cost would then be calculated

$$C = \sum_{m \in w} C_{m_{t^*}} * n_l * \sum_{t \in T} \lambda_{m_t} * (t_{t+1} - t_t) \tag{3.23}$$

As explained previously, if the cost function is linear, the ideal time to inject a fault (used here) is in the center of the interval.

3.5 Example: Wire Design

To understand how expected cost modelling enables the overall framework of design exploration, modelling, decision-making, and validity testing shown in Figure 1.2 (that is, the overall contribution of this dissertation), consider the following elementary example. In this problem the goal is to design a signal-carrying medium (such as a wire) in a way that minimizes the losses due to faults through resilience.

3.5.1 Modelling

This problem is shown in Figure 3.6, with the functions to embody in the design (`Import, Guide, and Export Signal`), possible failure modes (with rates and costs), and conditions that lead to failure modes. As represented in this model, there are two initiating failure modes in the `Import Signal` function–no input signal and an infinite (shock) signal–and one condition and resulting conditional mode in the `Guide Signal` function: a condition which specifies that the function should

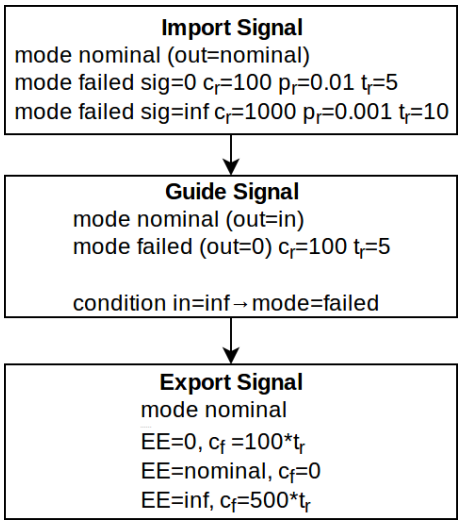


Figure 3.6: Functional model of a signal-carrying medium, with modes, conditions, costs, and probabilities associated to each function.

enter the failed mode when a shock signal is received, and a failed mode which makes the output signal zero if it receives a shock signal. The resulting failure costs are shown in the `Export Signal` function, based on the system exporting a zero, nominal, or shock signal. While the resulting fault behaviors of this system are simple enough to derive by hand, the first step using this framework would be to represent this system in a model and propagate the dynamic behaviors through it to determine the costs of each failure scenario.

3.5.2 Design

The second step in this framework is considering possible design variants which would lead to more or less resilience overall. In this problem there are two design variants: one in which the `Guide Signal` condition exists (Design 1), and one in which it does not (Design 2). These designs each have two scenarios resulting from the initiation fault modes: the scenario in which the incoming signal is failed with a value of zero, and one in which a shock enters the signal (e.g. from a short circuit). In the first scenario, the expected cost is the same for both designs. The zero signal state

propagates from the `Import Signal` function to the `Export Signal` function, resulting in a cost of failure resulting from the cost rate of the failed flow state, the time needed to repair the fault, and the cost of repairing the fault:

$$\begin{aligned} C_{S1} &= P_{f1} * (C_f * t_r + C_r) \\ &= 0.01 * (-100 * 5 - 100) = -6 \end{aligned}$$

In the second scenario, the expected cost is different for each design, due to the differences in fault propagation. In the first design, the shock causes a failure mode in the `Guide Signal` function, resulting in a zero flow state in the `Export Signal` function, and costs from the failed (zero) flow state and the times and costs needed to repair the `Import Signal` and `Guide Signal` functions. In the second design, the shock propagates through the `Guide Signal` function, resulting in costs from a failed (inf) flow state and the time needed to repair the function. These costs are tabulated for each design below:

$$\begin{aligned} C_{S2} &= P_{f2} * (C_f * t_r + C_r) \\ &= 0.001 * (1000 + 100 + 100 * 10) = -2.1 \\ &= 0.001 * (1000 + 500 * 10) = -6 \end{aligned}$$

3.5.3 Selection

The final step in this design framework is to select a the best design based on resilience attributes. In this case, the costs of resilience for each design based on the fault scenarios has already been tabulated, which enables one compare it with operational costs in an overall value assessment. In this example problem, it is assumed that the designs had the same operational costs and design costs of -2 and -1, respectively. The resulting overall cost score is tabulated in Table 3.2. As can be seen, while Design 1 has slightly more design cost, the lower expected cost resulting from *not* propagating the shock flow results in a better overall cost score. Given different numbers for

Table 3.2: Cost comparison of Design 1, as shown in Figure 3.6, and Design 2, with the condition removed.

Costs	Design 1	Design 2
Design	-2	-1
Operational	20	20
Scenario 1	-6	-6
Scenario 2	-2.1	-6
Total	9.9	7

probabilities, repair times, and costs, however, this result would change. For example, if the `Guide Signal` function had a repair time of 100, the resulting repair time would make the expected cost of failure of Scenario 1 for Design 1 -11.1, making the overall cost score worst than that of Design 2. Alternatively, if said design had commensurately lower design or operational costs, this design would still be superior using this method. This demonstrates how expected cost can be used to trade off the modeled dynamic failure response, failure costs and probabilities, and design and operational costs to determine if a resilient feature should be added to a design.

3.5.4 Validity Determination

To determine the validity of this design process, the affect of uncertainty on the design choice is considered. In this simple problem, we consider there to be one uncertain assumption: the probability of Scenario 2, which has a probability of 0.9 that the assumption used was correct, and a probability of 0.1 that it was incorrect. The resulting cost for scenario 2 is:

$$\begin{aligned} C_{S2} &= P_{f2} * (C_f * t_r + C_r) \\ &= 0.01 * (1000 + 100 + 100 * 10) = -21 \\ &= 0.01 * (1000 + 500 * 10) = -60 \end{aligned}$$

To test the effect of this scenario on the design process, we can see how the assumption scenarios affect the overall tabulation of expected design cost and see how the designs compare over the

Table 3.3: Validity determination of the decision in Table 3.2 to an uncertain scenario.

Scenarios	Nominal		Poor Assumption		Expected	
Costs	D1 - Nom.	D2 - Nom.	D1 - Scen.	D2 - Scen.	D1 - Exp.	D2 - Exp.
Design	-2	-1	-2	-1	-2	-1
Operational	20	20	20	20	20	20
Scenario 1	-6	-6	-6	-6	-6	-6
Scenario 2	-2.1	-6	-21	- 60	-3.99	-11.4
Total	9.9	7	-9	-47	8.01	1.6
Probability	0.9	.9	0.1	0.1		

expected cost over these scenarios. As shown, in the scenario with the different assumption, the design choice of Design 1 is still justified over Design 1. Thus, when the expected cost of uncertain scenarios is considered, Design 1 is still preferable, meaning that the design process is valid over this assumption.

3.5.5 Wire Conclusion

This problem illustrates how the overal resilience-based design framework presented in this dissertation operates at a high level. However, most problems are more complicated than this very simple example and are thus difficult or tedious to solve by hand. In practice, engineered systems often have multiple functions with complex behavioral interactions that affect how failures will propagate over time. While this problem only took a few calculations to solve, in the design of a complex engineered system the design considers more functions, each of which with several modes and complex dynamic behaviors, and the choices to consider are more complex than a simple choice of two options. In these situations, it becomes necessary to use a computational framework to organize and model these complexities, rather than perform every analysis by hand. To enabling this process to be performed in more realistic, complex engineered systems, Chapter 4 and 5 leverage computation to enable efficient and rigorous resilience modelling and exploration.

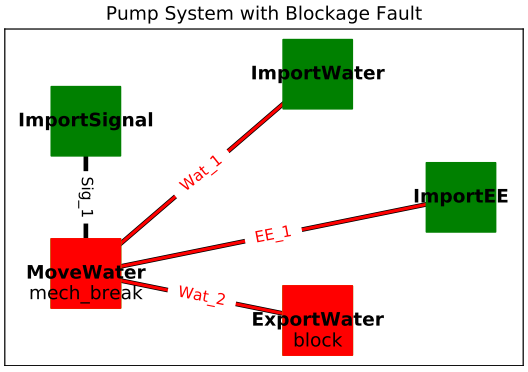


Figure 3.7: Considered Pump System with resulting degraded functions and flows.

3.6 Example: Verification of Expected Resilience Quantification in Pump System

This section demonstrates the dynamic expected cost modelling approach of Section 3.4 and compares different sampling methods based on their error over various cost functions. This is done in a model of a blockage faults in the pump system model shown in Figure 3.7. This model consists of five functions–`ImportSignal`, `ImportWater`, `MoveWater`, `ImportEE`, and `ExportWater`, where `MoveWater` is the pump itself and the rest of the functions represent sources or sinks for the flows, which are `Sig_1`, the signal to turn the pump on or off; `EE_1`, the electrical source powering the pump; `Wat_1`, the flow of water into the pump; and `Wat_2`, the flow of water out of the pump. This model was implemented in the `fmdtools` software package presented in Chapter 4, and is available at Ref. [121]. The modelled behavior of the system is shown in Figure 3.8 with and without faults. When the system behaves nominally, it turns on at $t = 5$ minutes, causing water to flow and current to be drawn until $t = 50$ minutes, when it shuts down. In the modelled fault scenario, a blockage in the pipe is injected at $t = 10$ minutes, causing the water flow rate to decrease, the pressure to increase, and motor current to increase. After a delay (e.g., resulting from an off-flow tank or the ability of the pump housing to sustain leaks for a given amount of time), the motor breaks (resulting in a new fault), the flow of water decreases to zero, and the current decreases also. This fault behavior has three modelled cost functions:

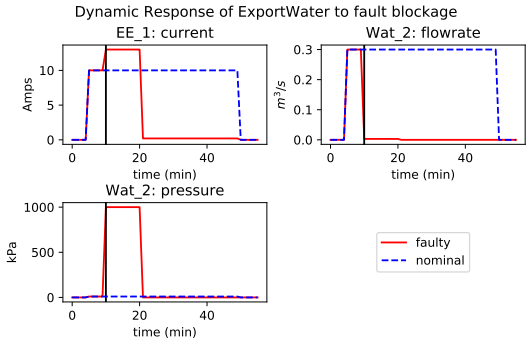


Figure 3.8: Behavior of Pump System Resulting from a Blockage at t=10 minutes

- Repair costs $c_m(t)_r$, where t is time in minutes:

$$c_m(t)_r = \begin{cases} 5000 & \text{if new fault} \\ 0 & \text{otherwise} \end{cases} \tag{3.24}$$

- Cost of decreased water flow $c_m(t)_w$, which follows the following equation, where $w_n(t)$ is the rate of flow in the nominal state and $w_f(t)$ is the rate of flow in the faulty state:

$$c_m(t)_w = 750(w_n(t) - w_f(t)) \tag{3.25}$$

- Cost of increased current draw $c_m(t)_e$, which follows the following equation, where $e_f(t)$ is the faulty current draw and $e_n(t)$ is the nominal current draw:

$$c_m(t)_e = 14 \sum_{t \in E} (e_f(t) - e_n(t)) \tag{3.26}$$

  where $E = \{i | e_f(i) - e_n(i) > 1.0, i \leq t\}$ is the set of fault times with the increased current.

Simulating the fault over each time-step of the model with each cost function results in the total cost responses in Figure 3.9. This cost is different than the individual values of each of cost function for each metric shown in Figure 3.8; it instead shows the total cost of simulation at each injection time, not the values of the cost functions over a single simulation. As shown, repair cost responses

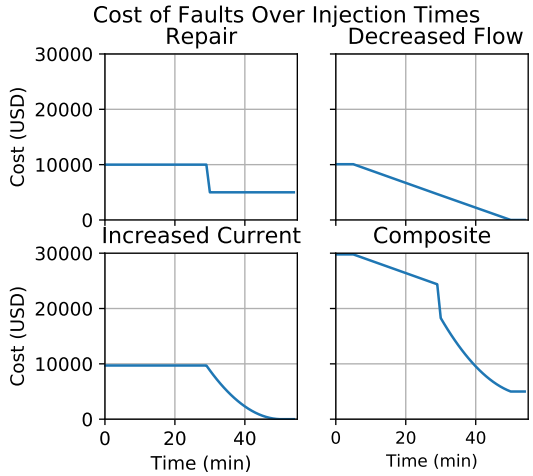


Figure 3.9: Cost responses over fault injection times for the blockage fault. (Delay = 20 M)

Table 3.4: Average Error Over All Points for each sampling method

MC (Std.)	MC (AV)	Evenly-Spaced	Likeliest	Gauss-Legend.
0.0742	0.0201	0.01901	0.3893	0.0069

are a step function, where the pump breaks when the fault is injected on the left side of the step, and does not break when the fault is injected on the right side of the step because the pump is turned off first. The cost of low water flow is a linear decrease over the interval the pump operates, since it results from the constant water loss over the interval. Finally, the cost of increased current is a constant function up until a point, after which it has an exponential decrease, since it results from the accumulated damage on the electric system, a process that only occurs during the delay between the blockage and the pump breaking.

The following comparisons test two overall approaches to sampling a resilience model in a design process–sampling *a priori* without knowing the form of the underlying cost function (e.g. where a discontinuity is), and sampling *a posteriori* given full knowledge of it.

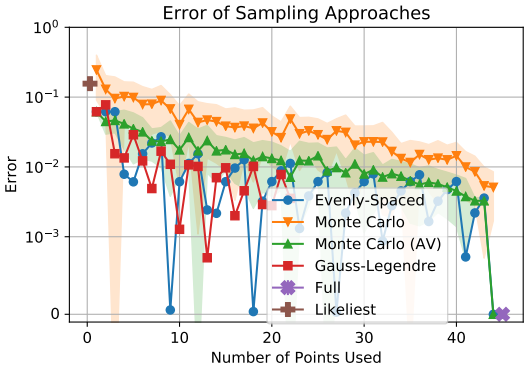


Figure 3.10: Error of sampling approaches over the number of samples used.

3.6.0.1 A Priori Approaches

In the *a priori* situation, the designer uses the sampling strategy to approximate the expected value of resilience without knowing the underlying form of the resilience function. A variety of methods can be used to to this, including placing a single sample in the likeliest phase, using the Monte Carlo technique (where samples are chosen randomly), using the Monte Carlo technique with the method of antithetic variates (which chooses an additional sample symmetric to the randomly-taken sample–see [77, Sec. 6.4.2]), sampling the interval evenly, or by sampling using a quadrature. Figure 3.10 compares the error and efficiency of these sampling approaches over the number of points used (averaged over 20 samples for the Monte-Carlo approaches), using Gauss-Legendre as the comparison quadrature, and Table 3.4 shows an average of sampling error over all points used. As shown, increasing the number of points reduces error for all approaches, however, the Gauss-Legendre reaches acceptable error ($< 1\%$) for all $n \geq 2$ points. While the Monte Carlo method performs poorly, average error improves to parity with the evenly-spaced quadrature when using the method of antithetic variates. However, because of the performance variance when using the antithetic variates approach, one would still prefer using the evenly-spaced quadrature in practice. This comparison demonstrates the advantage of using numerical integration for resilience sampling compared to the commonly-used Monte Carlo method–when the underlying model is deterministic (that is, there is no underlying modelled stochastic process to quantify), Monte Carlo methods

Table 3.5: Tested Quadrature Weights and Node Locations

Name	Source	Weights	Node Location
Gauss-Legendre	25.4.29, [2]	$w_i = \frac{2}{(1-x_i^2)[P_n'(x_i)]^2}$	$x \ni P_n(x) = 0$
Fejér 1	[289, 262]	$w_i = \frac{1}{n}(\gamma_0 + 2\sum_{m=1}^{n-1} \gamma_m cos(m\theta_i))$	$x_i = cos(\theta_i), \theta_i = \frac{(2i-1)\pi}{2n}$
Fejér 2	[289, 262]	$w_i = \frac{2sin(\theta_i)}{n+1} \sum_{s=0}^{n-1} \lambda_s sin((s+1)\theta_k)$	$x_i = cos(\theta_i)$, $\theta_i = \frac{i\pi}{n+1}$
Gauss-Lobatto	25.4.32, [2]	$w_i = [\frac{2}{n(n-1)}, \frac{2}{n(n-1)[P_{n-1}'(x_i)]^2}, \frac{2}{n(n-1)}]$	$x \in \{-1, 1\} \cup \{x \ni P_{n-1}'(x) = 0\}$
Even Spacing	NA	$w_i = \frac{1}{n}$	$x_i = -1 + i(\frac{2}{n+1})$

introduce unnecessary sampling error because the points are randomly selected.

Now that the advantage of numerical approaches for sampling resilience models has been shown in general, the following demonstrates the performance of several commonly-used quadrature approaches on the same pump resilience quantification problem. The quadrature approaches used are Gauss-Legendre, Fejér 1 and 2, Gauss-Lobatto, and a simple uniformly-weighted evenly-spaced quadrature. The weightings and nodes for each of these quadratures is given in Table 3.5 for n points, where the index $i = [1, ..., n]$, $P_n(x)$ is the Legendre polynomial of order n, γ_n is the weighted moment of the Chebyshev polynomial of order n, the interval the quadrature is spaced over is $[-1, 1]$, and the weights as implemented are scaled such that $\sum w_i = 1$. The `quadpy` Python package [247] was used to implement these approaches, which provided the node locations and weightings for each quadrature at a given number of points. To sample the discrete-times in the model, the node locations were rounded to the nearest time-step. To best show how each approach approximates different cost functions, they are compared over differing cost functions, numbers of points used (1 to 14), and the delay time between the blockage of the pump outlet and the pump breaking.

Figure 3.11 shows the resulting comparison of quadrature accuracy and simulation cost. As shown, the integration error depends on the number of points used and the underlying cost function. Generally, none of the quadratures perform very well over the (step function) repair costs, with error decreasing erratically with the number of points used. On the other hand, nearly all of the quadratures perform well at approximating the linear flow costs, which is expected for all symmetric quadratures, as explained in Section 3.4.1.1. As a function of number of points, the error of the

approaches when approximating current costs is nearly as variable as repair costs, but decreases more quickly. When the costs are put together, these variable repair and current costs have a large impact on the performance of different approaches, since both are difficult to reduce. The main difficulty with *a priori* approaches is thus approximating discontinuous or non-smooth cost response functions.

Table 3.6: Average Error of Quadratures Over Delays and Points

	Gauss-Legendre	Fejér	Fejér 2	Gauss-Lobatto
Repair	0.034412	0.037347	0.032904	0.043741
Decreased Flow	0.000064	0.000102	0.000058	0.000218
Increased Current	0.016802	0.024868	0.014384	0.036042
Composite	0.014566	0.017100	0.010562	0.019561
Average	0.016461	0.019855	0.014477	0.024891

Table 3.6 summarizes these results using the average error of each quadrature approach over the set of numbers tested. As shown, the Fejér 2 quadrature rule outperforms the other quadratures over all costs. This is somewhat unexpected, because Gauss-Legendre has been shown both theoretically and empirically to have less error for a given set of points on smooth functions (Fejér is typically used because calculating the nodes and weights is faster computationally) [279]. This may be for a number of reasons, including the cost function being discontinuous/non-smooth, the particular delay times chosen being more favorable to Fejér, and error introduced from discretization. Of these possible explanations, error induced from discretization seems most likely because it also explains the error from approximating the water costs, which as shown in Section 3.4.1.1, should be zero because the underlying cost function is linear (see Fig 3.9). However, there is still error because the exact locations of points cannot be used (points are instead rounded to the nearest discrete time-step), and because the range itself has a finite number of possible points (which is especially relevant for the beginning and end-phases, which only have five time-steps total). These results would seem to show, then, that the Fejér 2 quadrature is more robust to discretization, however a more detailed

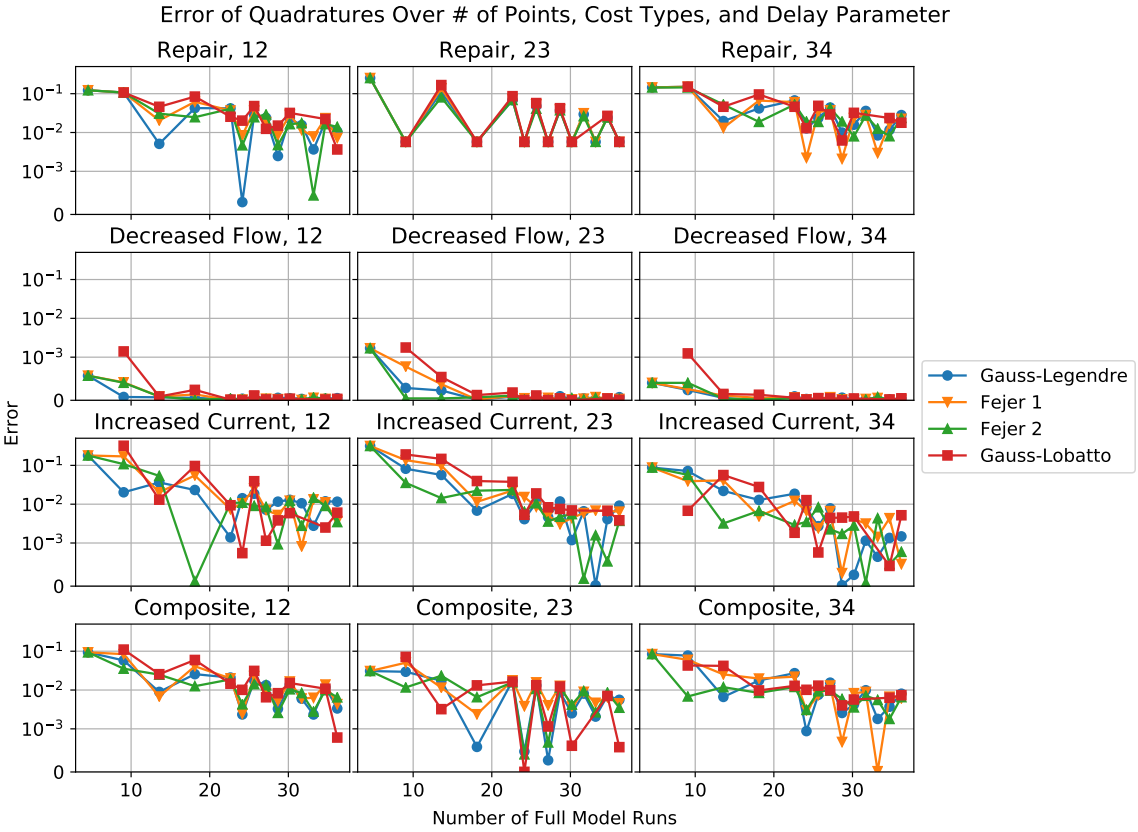


Figure 3.11: Ability of sampling approaches to approximate the full integral

study should investigate whether that is true for all problems (and provide the mechanism) or if a particular aspect of this problem and implementation is favorable to Fejér 2.

3.6.0.2 A Posteriori Approaches

In the *a posteriori* approach one seeks to minimize the error of the sampling quadrature given an already-known cost function. This section compares two *a posteriori* approaches over the composite cost function in Figure 3.9:

- a "best point" approach where the interval is approximated by the single point that minimizes error between the actual value and the point value

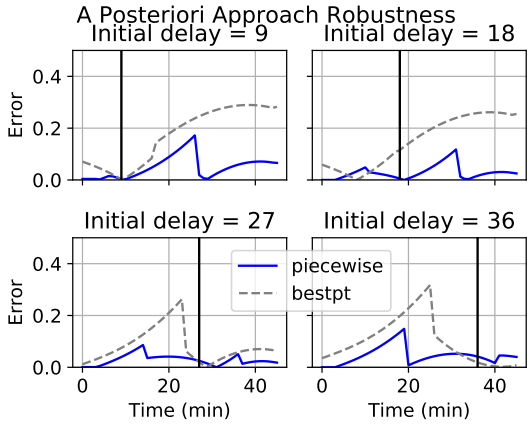


Figure 3.12: Robustness of a posteriori approaches to design changes.

- a "piecewise" approach where the interval is further split into sub-intervals approximated by a quadrature (in this case a single point) based on discontinuities in the cost function

To evaluate the error and robustness of each approach to model changes, the error in the cost function was compared over the set of possible delay parameters over four different possible values for the initial delay parameter used to form the initial quadrature. As shown in Figure 3.12, for most delays each approach has a fairly low amount of error at the initial delay, however the approximation is not perfect because for the "best point" approach there either is not a single point that represents the interval, and for the "piece-wise" approach the sub-intervals functions are nonlinear, increasing center-point approximation error. However, if the intervals in the piecewise approach were in turn approximated with a quadrature, this error would be reduced at the expense of more sample points. Additionally, while error increases for both approaches as the delay parameter deviates from the initial delay, the upper bound using the "piecewise" approach is much lower because it is approximating the function with two points instead of one. Nevertheless, this shows that even if one finds the best possible quadrature to approximate the cost function given there is an underlying discontinuity, design changes that shift the discontinuity will increase the error of the quadrature significantly.

3.6.1 Discussion

Considering resilience to be a statistical expectation has a number of implications for resilience-informed design processes. In this situation, it is desirable to reduce simulation cost as much as possible so that iterations can occur quickly while keeping accuracy high enough that the decisions being made are still valid. When working with a deterministic model, numerical integration approaches are the ideal solution to this problem, however the best approach depends on the following characteristics of the considered problem:

- **Simulation Cost Desirability:** If the model is relatively expensive to simulate, using fewer points is desirable to speed up iterations. On the other hand, if the model is inexpensive it may be unnecessary to use an efficient quadrature. Additionally, an inexpensive model makes it relatively easy to perform the *a posteriori* quadrature approaches, since the start-up cost to running the full set of simulations is not insurmountable.

- **Needed Simulation Fidelity:** If a high level of fidelity is needed, one might prefer using more points in a quadrature, while otherwise one might be more willing to approximate the cost with as few points as possible. For example, the model presented and used here is very high-level and there are many uncertainties not taken into account, such as behavioral parameters, rates, costs, and model abstraction. In a situation with these uncertainties, it may be a better use of computational and design resources to increase the fidelity of model inputs rather than spend more time increasing quadrature accuracy.

- **Cost Function Form:** As discussed in Section 3.4.1.1, if the cost response function is linear, it can be approximated with a single point. If it is nonlinear, on the other hand, it may be desirable to use a quadrature instead. Finally, if there are significant discontinuities due to conditional faults, it may be more desirable to run an initial set of simulations to find the location of the discontinuity to form the quadrature before proceeding–otherwise one would be forced to increase the number of points.

- **Design Changes Pursued:** Finally, it is important to consider how the design changes

pursued change the cost function. If the form of the cost function changes as a result of the design change, the quadrature approach will need to remain accurate over the change while quantifying a difference, if present, between design variants. As shown in Figure 3.12, if there is a step function cost of repairs, changing the location of a discontinuity will increase the error of a sampling approach, and, unless the delay crosses the points, there will be no change in approximated value of repairs, making it impossible to detect the effect of a design change to repair costs. On the other hand, if the design change solely increases the costs and rates (and not the intervals), the quadrature should remain effective and accurate.

3.7 Conclusions

This chapter presented a framework for considering resilience in early design using an expected cost function to trade-off between the expected design, operational, and fault response costs. This cost function is constructed to integrate with a scenario-based fault simulations which determine the cost of each failure event based on its effects: in Section 3.3 to a timeless simulation in the IBFM toolkit, and in Section 3.4 to a model of dynamic effects over time. Expected cost-based modelling is then demonstrated in two examples. The first example of a wire design in Section 3.5 demonstrated how expected cost enables the overall design framework presented here by allowing one to use fault scenario results to trade resilience with design and operational costs to enable decisions. The second example of quantifying costs in a simple pump model, provided in Section 3.6, demonstrated and compared different fault sampling approaches for use in dynamic models (e.g., in the fmdtools models in Chapter 4) to quantify the expected cost functions in Section 3.4. This was done because resilience metrics based on fault injection responses in a dynamic model are sensitive to injection time, and to most efficiently represent the statistical expectation of fault responses for these metrics. While some methods were shown to reduce error compared to others, choosing the best numerical integration approach depends on a number of design considerations, including the computational expense of simulations, needed simulation accuracy, underlying cost model form, and the effect of

design changes to be evaluated in the model. In summary, this chapter presented an expected cost metric for resilient design, provided two adaptations of this metric to different simulation modelling paradigms, and presented examples demonstrating and studying how this metric enables the overall resilience quantification and decision-making in the context of a fault simulation of the system.

Chapter 4: A Dynamic, Object-Oriented Fault Propagation Framework for Resilience Assessment

4.1 Motivation

Modelling hazards in early design involves representing the system in a high-level function structure to identify hazards and develop resulting design requirements [269]. In the design of aircraft, this process is called functional hazard assessment and follows the ARP4761 guideline [12, 8]. Model-based functional hazard assessment has been an active research area [152, 157, 207], with many new methods focusing on how to model different aspects of the system, such as human errors [125], dynamic behaviors, new flows resulting from failures [131], and operational decision-making [257]. However, there has been less demonstration of how to use this information to compare design alternatives, and the research codes underlying these methods have not been shared within the research community. To enable this resilience-based design, then, there is an opportunity to develop a tool that enables one to assess the resilience of a model without re-implementing underlying data structures and fault propagation methods.

The goal of the fmdtools project is to aid in the application of risk and resilience-based design frameworks by providing an open-source environment for modelling, simulation, and analysis of a new system in early design process, as shown in Figure 4.1. In doing so, it seeks to enable the practical application of early model-based functional hazard assessment frameworks while explicitly enabling the consideration of resilience. This toolkit has a number of potential applications to considering PHM in aerospace systems by supporting cost-benefit analysis (e.g., [103]), which can used to allocate resources for PHM [306] and assess design alternatives (e.g., prevention or recovery schemes) [113]. While most traditional risk assessment methods focus on how faults lead to system-level failures, the goal of the fmdtools project is to represent the full set of dynamic effects that

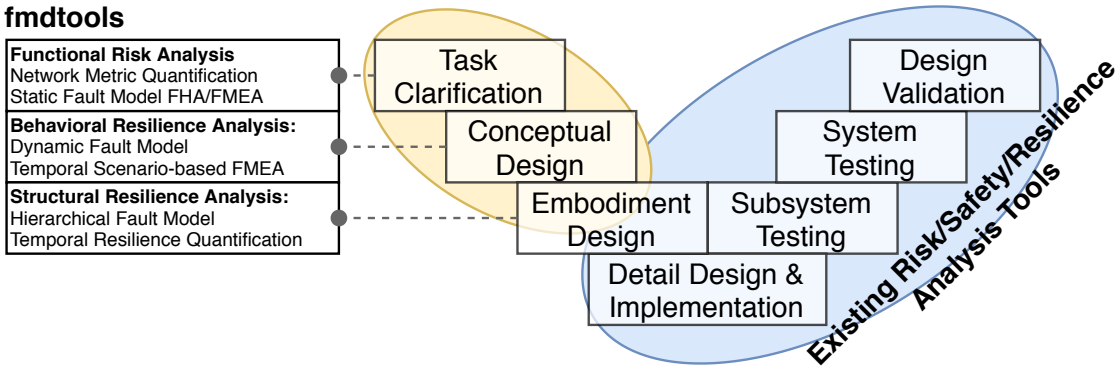


Figure 4.1: fmdtools is intended specifically to provide fault analysis methods that enable the consideration of risk in early conceptual design processes

result from a given fault scenario, so that one can compare the effect of different recovery actions. This requires advancing the modelling paradigm often used in early design.

For instance, one major difficulty in modelling failure propagation is representing the system in a way that captures the full set of effects which would be caused by a fault [104]. While a number of formalisms have been developed that enable this (e.g., modelica [37], simulink/lustre [135], etc), these models are often computationally expensive and are more applicable for later design stages when one has the complete set of system behaviors, as shown in Figure 4.1. As a result, model-based functional hazard assessment methods used in early design (i.e., on the left side of the V-model in Figure 4.1) are often coded directly in a base language (e.g., MATLAB/Python). In this setting, it is convenient to express the system using a *procedural, directed-graph* representation of system behaviors where each function is defined in a method and where the output flows of each function are used as inputs to the next function in the graph. The main problem with this representation is that it can only represent flow propagation that occurs in a single direction–from the source functions where flows originate to the sink functions where flows terminate. Defining a model in a base language also makes it difficult to then structure the model in a way that logically organizes

faults, states, and behaviors of functions to be be easily understood, visualized, and modified. To improve on this approach, fmdtools uses a *object-oriented, undirected-graph* representation of system behaviors, where each function's states, behaviors, and faults are defined in a function class and a model class is used to connect adjacent functions so that behaviors can propagate in any direction through the model graph and one can easily parse the structure of the system model.

The rest of the chapter first outlines the outlines the general modelling framework and methods provided in the fmdtools project in Section 4.2 and then provides two examples of modelling case studies: a multirotor drone model in Section 4.3 and a wildfire response model in Section 4.4. The multirotor drone model provides a cross-section of the methods across different levels of model fidelity as well as the methods described in Section 4.2–demonstrating its use in a typical design process. The fire response model, on the other hand, demonstrates the versatility of the fault propagation toolkit to modelling complex scenarios by showing its use in a dynamic system-of-systems context with specialized assets and behaviors.

4.2 Methods and Algorithms

The fmdtools toolkit aims to provide a design, analysis, and simulation environment that enables the incorporation of resilience into a system design. To accomplish this, it provides a number of tools to represent, simulate, and analyze the system as it progresses through the design process, as shown in Figure 4.2. As a result, it can accommodate a number of modelling and analysis use-cases to progress from the early, abstract representations of the design (e.g., network and static propagation models) to more detailed representations of the system structure (e.g., dynamic and hierarchical propagation models) and behaviors in the same modelling environment, as shown in Figure 4.3.

The full implementation of this work is provided in a publicly-available repository, along with examples and documentation at `github.com/DesignEngrLab/fmdtools` (or [121]). While an exhaustive description of every method and class is out of the scope of this work, it will discuss some of the underlying concepts and structure of the toolkit. The fmdtools toolkit is organized into

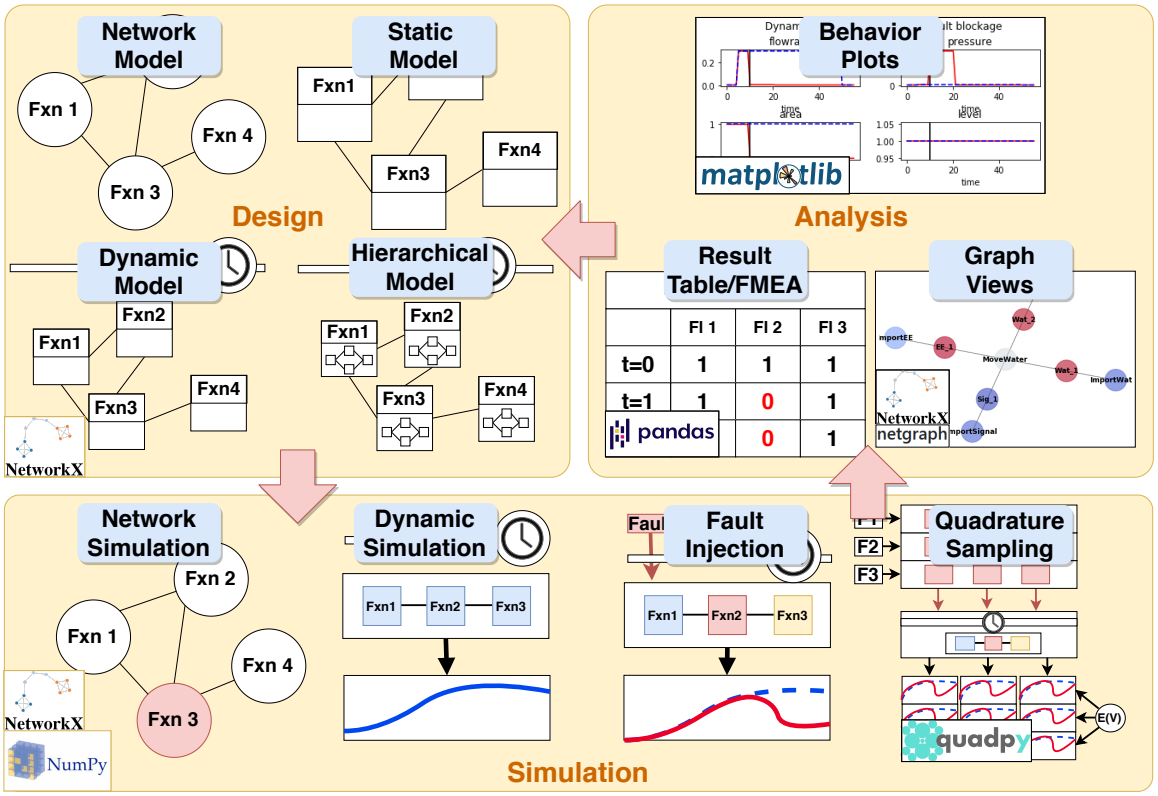


Figure 4.2: The fmdtools design, simulation, and analysis environment.

different modules used for model representation, simulation, and analysis. The `modeldef` module provides the classes to define a system model from functions, flows, and components as well as a fault sampling approach for resilience quantification. The `faultsim` module then provides methods for propagating faults in a model and quantify network metrics. Finally, the `resultdisp` module provides a number of methods to process and visualize simulations of the model, including behaviors over time, FMEA tables, fault graphs, and heatmaps.

4.2.1 Model Representation

In this work a system consists of functions (modules that perform a task), components (specific solutions to a function), and flows (variables) that are connected with each other in a bipartite

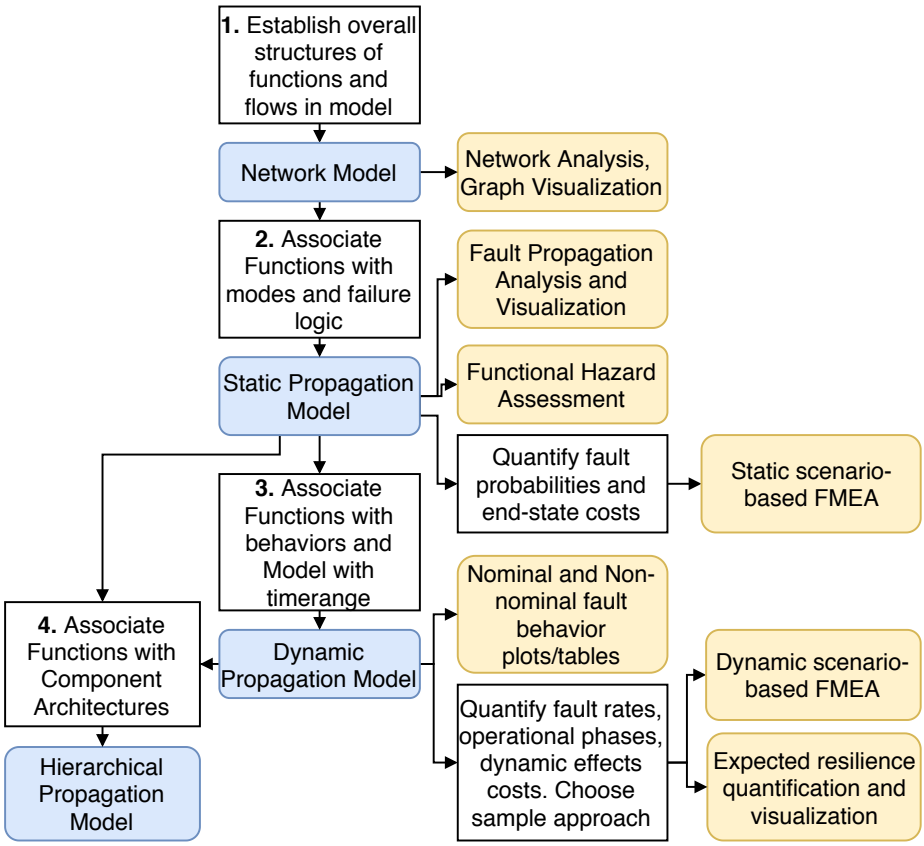


Figure 4.3: Fault model types and analyses enabled by fmdtools. Note that since model types build on each other, the analyses from less detailed model types (e.g. static propagation models, network models) can apply to more detailed model types (e.g. dynamic propagation models, hierarchical propagation models).

graph. In a model (itself defined by a user-defined class), functions, flows, and components are represented by objects instantiated from user-defined classes that are connected by a graph, as shown in Figure 4.4 for a model of a wire. In this representation, each function (e.g., Transport EE) consists of its associated flow objects (e.g., EE 1, EE 2), internal state variables, set of faults, behavior methods, and constituent component objects (if a component representation is used). Flows are in turn defined by dictionaries of states with corresponding values and components are defined by internal state variables, a set of faults, and behavior methods. Model objects are then composed of their constituent functions, flows, and components as well as a graph object used to track the

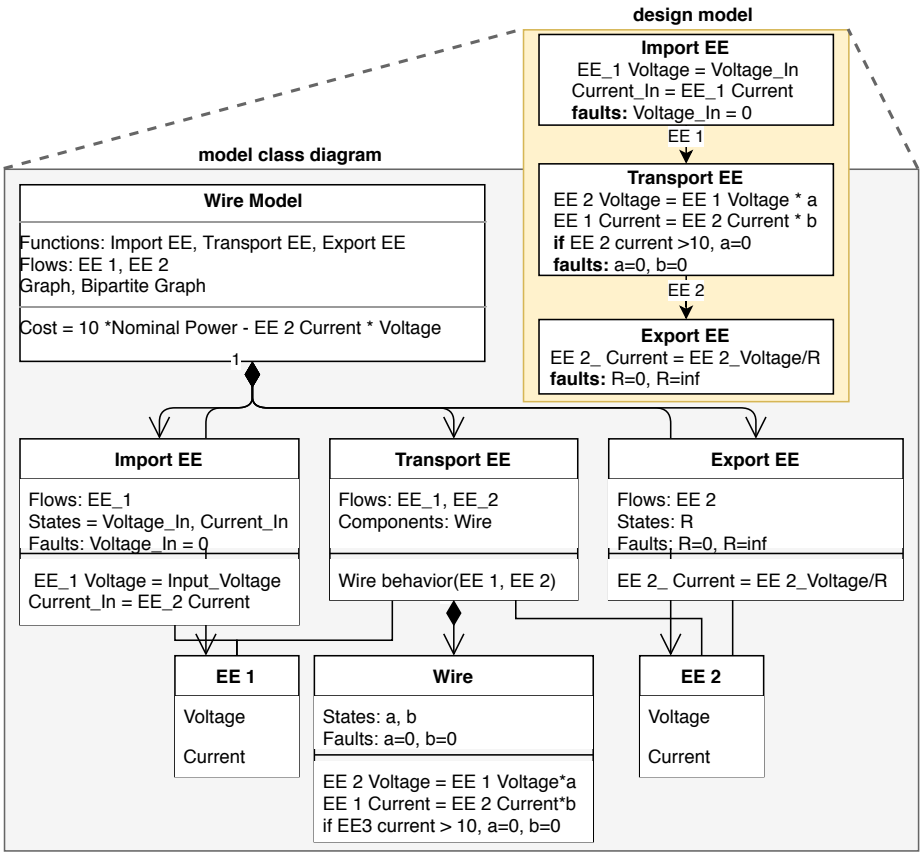


Figure 4.4: Example model representation. A Model is composed of function objects with internal states and faults, (optional) instantiating component objects and relationships to flow objects.

connections between functions and flows and a classification method used to quantify the costs of a fault scenario propagated in the model.

This model is constructed in fmdtools by defining a subclass that extends the `Model` class to represent a system of interest. This model class is constructed by defining the simulation parameters (e.g. units, timesteps, starting and ending simulation times, design parameters), adding each flow in the model, adding each each function in the model and connecting them with flows to construct the model graph, and creating a classification method which determines the rate, cost, and expected cost of a given scenario given the results and parameters for that simulation–the end-state of functions and flows (e.g. values and faults present), the scenario properties (e.g. rates, faults, etc.), and the history of model states (values for flows over time). By instantiating this class, one can then use the resulting object to model the evolution of system states over time for a given set of defining parameters.

4.2.1.1 Functions

In design, functions represent the high-level tasks performed by the system [217], as describe in Section 2.2.3. In fmdtools, the `FxnBlock` class is used to represent these functions, which constitute the main building block of the system model defining faults and system behaviors. To use this class, users define a subclass for each function which uses inherited `FxnBlock` attributes to represent the properties of the function. At its most basic, a user-defined function class is defined by the flows going in and out of the function, a behaviour method which relates values of input flows to values of output flows, and a set of faults which modify the input-output relationship defined in the behavior method. However, more attributes can be defined, including states (internal variables of the function tracked in the model history), conditional fault methods defining input/output flows or function states result in faults, timers that can be used to express delays in behaviors, and components, as described in Section 4.2.1.3.

To enable expected resilience quantification, each fault to inject in the function can be associ-

ated with a probability model that expresses the likelihood of the fault occurring, as decribed in Section 3.4. This probability model is defined by a function-wide rate, the proportion (or rate/probability) of faults resulting from the mode, an opportunity vector expressing the relative likelihood of a fault occurring at a specific phase of operation, and a specification of whether these values are a rate (with units) or a probability. Each fault can additionally be given a cost of repair that can be used to calculate the costs of fault scenarios.

4.2.1.2 Flows

Flows represent the states (e.g., energy, material, or signal) with which the functions interact to achieve the overall goal of the system. In fmdtools, the `Flow` class is used to represent these states, which can either be extended in a user-defined subclass (if there are special properties the designer wishes to represent) or instantiated using a dictionary with the name of the flow, name of the values characterizing the state of the flow, and initial quantity for each value. Because of the undirected nature of the model graph and associative relationships between functions and flows, flows are accessible (i.e., values can be changed) in all connected functions.

4.2.1.3 Components

While functions represent the task a system performs, components can be used to represent realizations of that function. Often in risk or resilience-based design, one is interested in designing component architectures which will fulfill an overall function, even when an individual component fails [306]. To represent this, fmdtools uses the `Component` class, which also is composed of the main attributes of the `Function` class, including internal states, a behavior method, and a set of fault modes. Similarly, to use the `Component` class in a model, one must define a subclass for the modelled component with its own states, fault modes, and behavior method. However, unlike the `FxnBlock` behavior method (which acts on `FxnBlock` attributes and does not return anything), `Component`

behavior methods explicitly take the inputs of the component behavior as input and return outputs of the behavior as output. This is done to enable the designer to relate the inputs and outputs of the components in the architecture fulfilling a function with each other and function states in the behavior methods of its corresponding function.

4.2.2 Resilience Simulation

The fmdtools toolkit has two main approaches for assessing the resilience of a model: quantifying network metrics of the system architecture and propogating faults in the system model. Network metric quantification, described in Section 4.2.2.1 and implemented in the `networks` submodule, enables consideration of the structural resilience of the system before specifying the fault logic in the system. Fault propagation, described in Sections 4.2.2.2 and 4.2.2.3 can then be used to assess the resilience of a model given the model has faults to propagate in each function (or the function of interest) and the functions each have the necessary fault logic and/or behaviors.

4.2.2.1 Network Metric Quantification

The network metric quantification tools in `fmdtools` enable the early assessment of the failure tolerance and resilience of a design. Network metric quantification is performed using the `networks` submodule, relying on the internal `networkx` graph representation of the functions and flows of the model. Network analysis is an emerging early design methodology for predicting the likely failure tolerance of a design without the need for high fidelity models [88, 189]. In this approach, engineered systems are represented as networks in which nodes represent functions, parameters, or components depending on the specific network formalism. In short, network analysis enables analysis of the topology that emerges from the connectivity between system elements. The representation of connectivity between system elements is similar to that of a design structure matrix (DSM), and network theory enables visualization and powerful analyses of emergent network properties.

Networks are used for both *a priori* assessment of resilience properties as well as for quantifying the network's response to simulated faults.

Resilience Properties of Networks Known relationships between structure and failure tolerance [32] in complex networks enable the *a priori* assessment of a network's resilience properties prior to simulation. The structure of a network is intrinsically related to its functionality; structural vulnerabilities are therefore relatable to loss of functionality. In networks, failure tolerance is typically studied by attacking or removing nodes and measuring the resultant degradation of the network structure. That is, similarly to how a loss of functionality in one component in an engineered system leads to decreased overall performance, degradation or removal of one node in a network leads to an alteration of the network's topology. In this way, a network's resistance to failure is relatable to an engineered system's resistance to failure. Certain networks are more prone to degradation due to node removal than others. Likewise, certain nodes are more prone to causing degradation than others. In component networks, failure or removal of a component node implies loss of functionality in that component, and the analysis therefore relates to the effects of the failure of that function.

A common method for characterizing a network is studying its degree distribution. In an undirected network, the degree of a node is its the number of connections (edges). Nodes with high degree (hubs) tend to be more critical in retaining a network's functionality. A network's degree distribution is a histogram of the degrees of all nodes in the network. Degree distributions that follow a bell-curve tend to be more vulnerable to random node removal, whereas degree distributions that follow a power law tend to be more vulnerable to targeted node removal [21] (i.e., removal of a hub). High degree nodes are identifiable using `find_high_degree_nodes`. Degree distributions are provided using `degree_dist`.

The modularity and community structure of a network also have significant bearing on the network's failure tolerance. Modules, or communities, are tightly coupled groups of nodes. The modularity of a network, the degree to which a network exhibits a modular structure, is typically measured using Q-modularity [204]. Nodes that connect modules – bridging nodes – are functionally important to a network's failure tolerance [290]. This is comparable to, for example, identifying

high severity failures in FMEA. Networks with high modularity have been found to be less robust overall [291]. `find_bridging_nodes` is provided to identify bridging nodes in the network model. `calc_modularity` is provided to compute the modularity of the network model.

Average shortest path length (ASPL) is a measure of the efficiency of the spread of information through a network. Under attack, networks with low ASPL are more likely to retain short to moderate length paths between any given pair of nodes, whereas networks with high ASPL are more likely to disintegrate significantly under attack. ASPL is defined as the mean of the sum of all edge weights along the shortest path between each pair of nodes in a network and is available as `calc_aspl`.

Simulation-Based Analysis of Network Resilience In addition to using a network's structure to predict its response to failure, it is also possible to use simulation-based approaches for network analysis. First, robustness coefficient simulates the effect of failure in the network. This approach, implemented as `calc_robustness_coefficient`, measures the changing size of the largest connected component of a network during successive node removal [284]. A second simulation-based approach is an SFF (susceptible-failed-fixed) epidemic spreading model, which is able to explicitly represent node recovery [189]. Rather than attacking or removing nodes as in the robustness coefficient, this model considers nodes to be in a susceptible, failed, or fixed state. Failed nodes may cause susceptible nodes to fail, similarly to infected persons spreading an epidemic. After a node is fixed, it is unable to fail again by the same cause (immunity). The SFF model is available as `sff_model`.

4.2.2.2 Fault Propagation

Propagating of faults in a model has two major aspects: static fault propagation and dynamic fault propagation. Static fault propagation is the process of determining the immediate effects of a fault in a system, as illustrated in Figure 4.5. First, all of the behavior methods are run. If a new value occurs in one of the functions (e.g., because of fault injection) or its associated flows, that function and functions adjacent to the changed flows are added to a list of functions to update. The

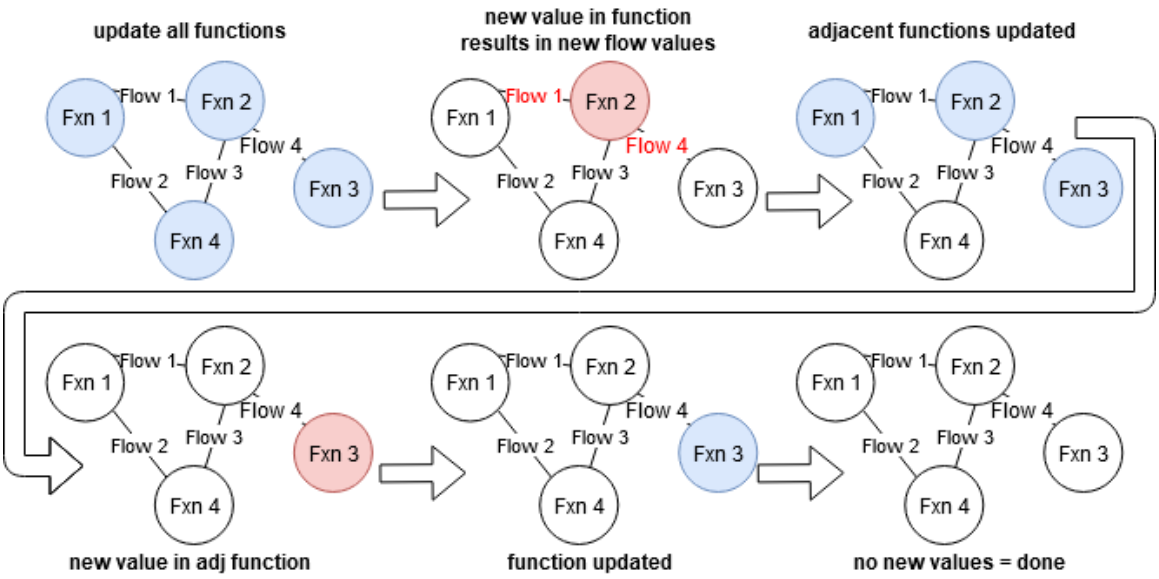


Figure 4.5: Illustration of static fault propagation. Function behavior methods are iteratively run in a list until the states of the system no longer change.

behavior functions for those functions are then run and new functions to update are again added if they receive new input values. This process is run iteratively until the system reaches a stable end-state, if a stable end-state is possible. Thus, one necessary property for fault models written in this framework is stable fault behavior–behaviors in one function should not change behaviors in other connected functions that will in turn change the original behaviors in the original function repeatedly, indefinitely.

As shown in Figure 4.4, functions have associative relationship with flows, meaning functions have full access to (and can change the values of) the states of both "input" and "output" flows. Because the propagation of system states is undirected, functions have the ability to propagate new system states to any other functions in the graph–not just the function that would be placed "next" in the sequence of tasks to perform. However, because of the undirected system representation, conflicts between function behaviors can occur when different functions specify different values for the same flow state, resulting in a non-convergent system state at a given time-step. This must be avoided in model setup, which can be achieved by representing flows with states that propagate

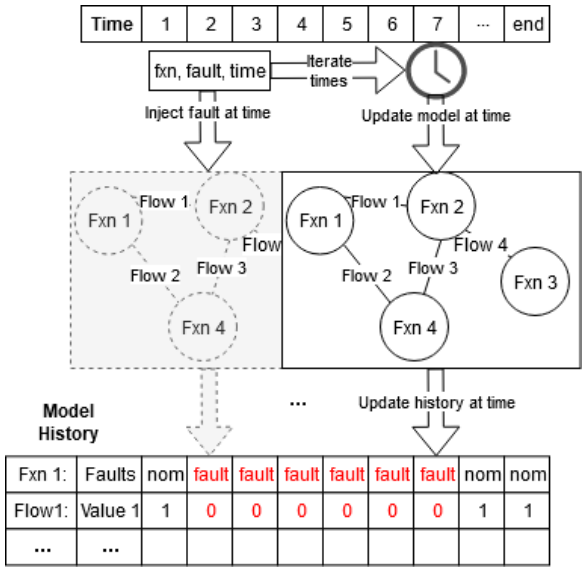


Figure 4.6: Dynamic fault propagation. A model is iteratively updated at each discrete time-step from fault injection to the end of simulation.

forward through the model graph (i.e. "effort" variables such as voltage or potential in a bond graph representation) and states that propagate backwards through the model graph (i.e. "flow" variables such as current or rate).

Dynamic fault propagation is the evolution of states in the model over time necessary to quantify resilience as a time-dependent property of a system. The implementation of dynamic fault propagation used here is illustrated in Figure 4.6. As shown, the static propagation procedure is iteratively run over a set of time-steps from fault injection time to the end of the simulation time. To fully assess resilience, a history is kept of all of the states of the model (flow values, function state values, faults in functions, etc.) over the set of time-steps. Based on a simulation like this, one can then quantify metrics for the simulation such as dynamic costs, recovery time, or worst state over time.

4.2.2.3 Fault Injection

Depending on the scope of the analysis, one might be interested in simulating faults in different ways. Before injecting faults, it is important to determine that the model performs as expected by simulating the system without any faults. Then, while setting up faults and fault behaviors (and in systems with single faults), one can propagate a single fault at a given simulation time to verify that the simulated behavior matches the expected behavior for that fault. Once faults are encoded, the list of faults can be propagated in the system at times defined in the model. While this approach lets one see the consequences of faults injected at set times, it may not be for calculating expected resilience metrics, since it neglects joint fault scenarios and when in time faults are most probable.

To quantify the mathematical expectation of fault-injection based resilience models, the `SampleApproach` class can be used to define the set of fault scenarios to propagate in the model, as illustrated in Figure 4.7. This class uses the dynamic probability model set up in the model, functions, and components, along with user-defined parameters to create a list of fault scenarios which will be used to represent the statistical expectation of the defined faults. This approach can be defined over the set of faults to include, the number of joint faults to inject, and the probability model for the joint faults (e.g., assuming independence or a conditional probability), and the times to inject the faults . The set of injection times is determined by two main parameters: the phases defined in the model (and opportunity vectors for each fault in the probability model), and the set of times within each phase. Within each phase, these times can be specified as every discrete timestep, an evenly-spaced approach with a set number of points, a randomly-spaced approach with a set number of points, or an approach using a given quadrature defined in the `quadpy` software package [249]. Additionally, Sample Approaches can be refined post-hoc based on a set of simulation results to represent the behaviors with a small set of sample points. Using these approaches, one quantify expected metrics iteratively with as few fault simulations as possible.

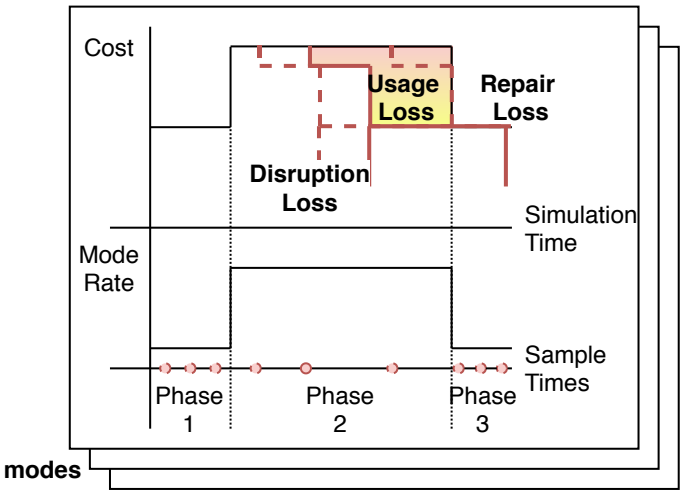


Figure 4.7: Injecting faults according to a fault sampling approach.

4.2.3 Resilience Analysis and Visualization

Using the models defined in Section 4.2.1 and simulations in Section 4.2.2, one can then perform analyses on the results. The fmdtools toolkit provides a number of different convenience methods using existing Python libraries, including `matplotlib` [123], `networkx` [84], and `pandas` [219] to makes sense of the fault behaviors modelled in the system. To assist with this analysis and visualization, the results of the simulations are processed to summarize the state of different aspects of the system as nominal or faulty. This process results in three categorizations for functions, flows, and components: nominal, when the data structure behaves as it does in the nominal state; degraded, when the data structure has a different behavior than it does in the nominal state; and faulty, when a component or function has a fault. This approach to result processing enables high-level visualization of the status of model structures without the user defining bounds or conditions for each variable to be listed as faulty or degraded. Using this representation, one can make a number of plots of the model graph structure, system behaviors over time, and tabular summaries of results.

4.2.3.1 System Graph Plots

To visualize the propagation of faults in the system, fmdtools provides a number of methods that display a graph view of the system using `matplotlib` [123], `networkx` [84], and `netgraph` in the `graph` sub-module. Graph views of the system enable one to see the structure of the model as well as desired states or properties of the functions and flows. Two main graph representations can be plotted: a default graph representation where the flows are plotted as edges between function (which are nodes) and a bipartite graph representation where both functions and flows are nodes and edges are the associative relationships between them. To visualize the model graph in an intuitive layout, methods calling `netgraph`'s `InteractiveGraph` class are provided which enable one to place nodes manually (rather than relying on an algorithmic layout). While the default representation is more intuitive to interpret–especially for simple systems–the bipartite representation often makes a better use of space–especially when a flow is connected to more than two functions–because there is less more freedom to ensure that edges do not overlap. Using the graph view, one can then visualize graph metrics as shown in Figure 4.8, the state of the model at the end-state or a particular time (or set of times) in the model history as shown in Figure 4.11, and various model run statistics defined by heatmaps (e.g. expected degradation time, maximum number of faults, etc.). These visualizations give one a view of how faults and behaviors propagate at the system level.

4.2.3.2 Dynamic plots

When modelling a dynamic system, it is often important and necessary to plot particular states over time in order to see how the behavior evolves over time. Methods in the `plot` sub-module use `matplotlib` [123] to show the evolution of chosen states of the model over time, with (if the simulation was a run of a fault scenario) faulty states overlaid over the nominal states over time to aid assessment of the fault-induced behavior, as shown in Figure 4.12. In addition to modelling system behavior in a particular modelling scenario, time-based plots also have the ability to visualize the cost responses given by the simulations at each injection time. This plot, shown in Figure 4.13,

can be used visualize how the sample approach defined in Section 4.2.2.3 approximates the expected resilience costs by showing the rate over time for a particular fault, as well as the modelled cost and quadrature weight for each sample point. Since these plots are plotted in `matplotlib`, well-known commands and interfaces can be used to edit and save the plots.

4.2.3.3 Tables

Finally, it is often helpful to be able to view the results of fault simulations in tabular form. While simulation results are typically returned as nested dictionaries, fmdtools provides convenience methods to view this information as a `pandas` [219] table to enable results processing and display. Based on the processed results, one can also make tabular summaries of simulations, such as the number of functions and flows degraded over time or in a particular simulation. Tables are most helpful for summarizing the results of a set of simulations, where they can provide an FMEA-style assessment of the functions and flows affected as well as the rate, cost, and expected cost of each fault simulation, as shown in Table 4.2, which can be generated to delineate between or summarize fault effects over each phase. Since these tables are implemented in `pandas`, existing interfaces can then be used to display and/or save results (e.g., as a `.csv`).

4.3 Example: Drone Modelling

This section illustrates how one can use the fmdtools software package to aid the conceptual design of a real system by providing analyses that increase in fidelity and detail with the design process. A number of examples are provided in the repository, including a conceptual model of a pump, a dynamic modelling of virus spreading, a human-operated tank system, and a static model of an electric power system.

This example considers the design of a multi-rotor drone which must fly to a given location and return to its destination. The functional model of this system is shown in Figure 4.11, which includes

Table 4.1: Network metrics for default (function) network representation of example drone model.

ASPL	Modularity	Robustness Coefficient
1.44	0.12	95.85

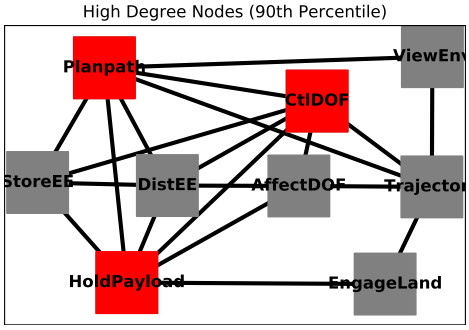


Figure 4.8: Visualization of high degree nodes in default (function) network representation of example drone model.

the rotor lines, structure, electrical power source and distribution, and path planning of the system. In this example, we first quantify metrics about the system structure, then use a static representation to generate a high-level FMEA and visualize fault propagation, then create a dynamic version of the model to visualize fault behaviors over time and quantify the effects of injecting faults at different simulation times, and finally use a hierarchical model to compare the dynamic fault responses and resulting resilience of different component architectures.

4.3.1 Network Representation and Analysis

First, the model is analyzed using network metrics and algorithms. In `fmdtools`, it is possible to analyze various network representations of the model, although only one network representation will be shown in each step. Each network analysis function has options to analyze the default (function) network, the bipartite (function-flow) network, the parameter network, or the component network. The bipartite network is treated as a unipartite-like network for analysis [89]. Analysis of the various network perspectives provides a more complete understanding of the model's resistance to failure.

The network metrics for the function network are given in Table 4.1. Low (close to zero) mod-

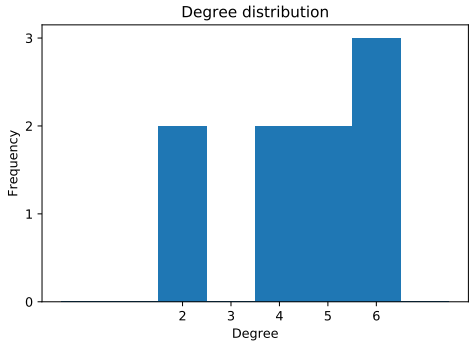


Figure 4.9: Degree distribution of default (function) network representation of example drone model.

ularity indicates a high degree of interconnectivity and has been correlated with high robustness [291]. High robustness coefficient and low ASPL indicate high robustness to random node failure. High degree nodes are highlighted in red in Fig. 4.8. Nodes with high degree, or hubs, are more likely to cause significant performance degradation if in a fault state. Based on this analysis of the system topology, we can conclude that the functions with the most opportunity to affect other functions at a topological level are path planning, control systems, and the structure of the drone, since these functions have the most connections with other subsystems. The degree distribution of the function network is presented in Fig. 4.9. The relatively homogeneous degree distribution in Fig. 4.9 indicates that the network is not particularly robust to failure of critical nodes. The SFF model for the function network is provided in Fig. 4.10. This model demonstrates the system's topological robustness to a cascading failure, given a user defined failure rate, fix rate, and start node (first node to fail). If various design alternatives are being considered, their relative resilience can be compared using the SFF model.

4.3.2 Static Representation and Analysis

To identify how faults lead to failures and begin to quantify risk in the system, the model is elaborated with flow attributes, function states, and failure logic, creating a static propagation model. As modelled in this system, deviations in the input of the `Trajectory` function block lead to a crash,

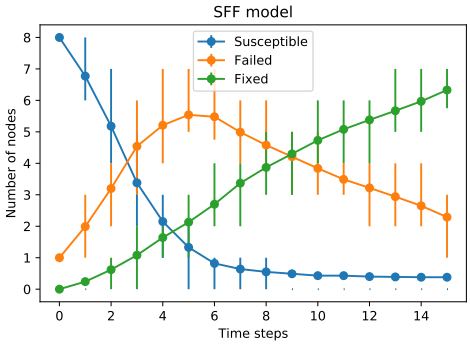


Figure 4.10: SFF model applied to function network representation of example drone model.

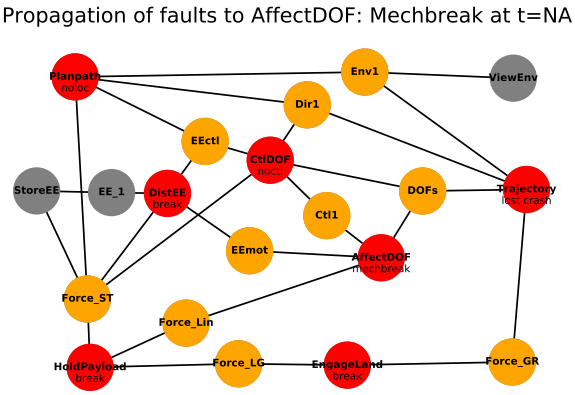


Figure 4.11: Static fault effects to the motor breaking: the drone crashes.

which in turn propagates faults through the structure to initiate faults in the rest of the functions. Using this model (and an underlying fault probability model), one can create a FMEA-like table of fault effects, rates, and costs, as shown in Table 4.2, to show which faults have the highest impact on the design. As shown, most faults in this model lead to a crash, making the chosen rate the driving factor of expected cost. One of the faults under consideration in this model is a mechanical break causing the `AffectDOF` function to lose the ability to control the degrees of freedom of the drone. This fault is visualized in Figure 4.11, showing how fault leads to a crash and in turn faults in the other functions. This fault will be used to motivate analysis and design through the rest of this example.

Table 4.2: Automatically-Generated Scenario-Based Static FMEA from model

Fault	Degraded Functions	Degraded Flows	Rate	Cost	Exp. Cost
StoreEE nocharge	StoreEE, DistEE, CtlDOF...	Force_ST, Force_Lin, Force_GR...	1e-05	183300	183300
Planpath degloc	DistEE, CtlDOF, Planpath...	Force_ST, Force_Lin, Force_GR...	8e-06	193000	154400
DistEE short	DistEE, CtlDOF, Planpath...	Force_ST, Force_Lin, Force_GR...	3e-06	186000	55800
AffectDOF ctlbreak	DistEE, AffectDOF, CtlDOF...	Force_ST, Force_Lin, Force_GR...	2e-06	184000	36800
AffectDOF ctlup	DistEE, AffectDOF, CtlDOF..	Force_ST, Force_Lin, Force_GR...	2e-06	183500	36700
DistEE break	DistEE, CtlDOF, Planpath...	Force_ST, Force_Lin, Force_GR...	2e-06	183000	36600
CtlDOF noctl	DistEE, CtlDOF, Planpath...	Force_ST, Force_Lin, Force_GR...	2e-06	183000	36600
AffectDOF short	DistEE, AffectDOF, CtlDOF...	Force_ST, Force_Lin, Force_GR...	1e-06	186200	18620
AffectDOF mechbreak	DistEE, AffectDOF, CtlDOF...	Force_ST, Force_Lin, Force_GR...	1e-06	183500	18350
AffectDOF openc	DistEE, AffectDOF, CtlDOF...	Force_ST, Force_Lin, Force_GR...	1e-06	183200	18320
Planpath noloc	Planpath, Trajectory	Ctl1, DOFs, Dir1	2e-06	60000	12000
CtlDOF degctl	CtlDOF	Force_GR, Force_LG, Ctl1, DOFs	8e-06	10000	8000
AffectDOF propbreak	DistEE, AffectDOF, CtlDOF...	Force_ST, Force_Lin, Force_GR...	3e-07	183200	5496
AffectDOF propstuck	DistEE, AffectDOF, CtlDOF...	Force_ST, Force_Lin, Force_GR...	2e-07	186200	3724
HoldPayload break	DistEE, CtlDOF, Planpath...	Force_ST, Force_Lin, Force_GR...	2e-07	183000	3660
ViewEnv poorview	ViewEnv		2e-06	10000	2000
EngageLand deform	EngageLand		8e-06	1000	800
HoldPayload deform	HoldPayload	Force_ST, Force_Lin	8e-07	10000	800
DistEE degr	DistEE	Force_GR, Force_LG, EEmot...	5e-06	1000	500
EngageLand break	EngageLand		2e-06	1000	200
AffectDOF ctldn	AffectDOF	Force_GR, Force_LG, DOFs	2e-06	500	100
AffectDOF mechfriction	AffectDOF	EE_1, EEmot	5e-07	500	25
AffectDOF propwarp	AffectDOF	Force_GR, Force_LG, DOFs	1e-07	200	2

4.3.3 Dynamic Representation and Analysis

In the dynamic model, the drone is given dynamic states and behaviors which iterate over time–in this case the position, velocity, charge, and perceived location of the system. This can then be used to model how the system behaves in faulty and nominal scenarios as shown in Figure 4.12 for the mechanical break fault. In the nominal case, the drone flies out to a location and returns to land, while in the faulty case the system crashes soon after the fault is injected, leaving it far from the

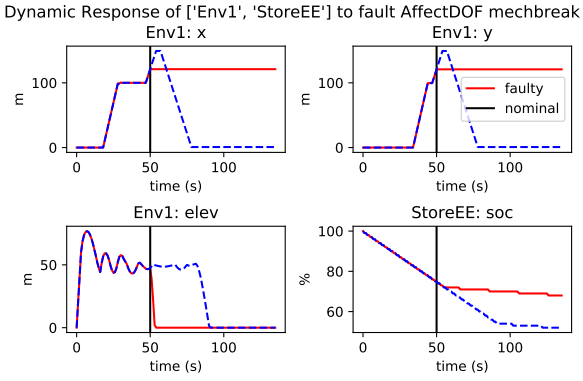


Figure 4.12: Dynamic behaviors of a motor breaking.

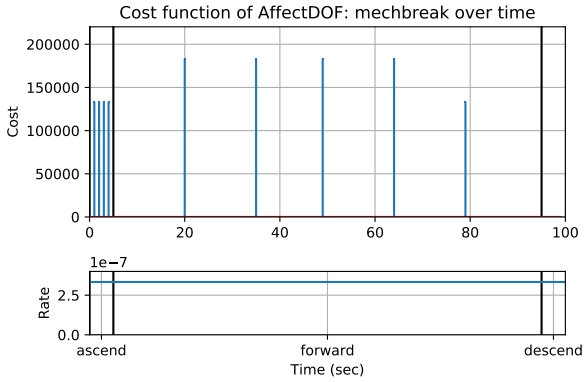


Figure 4.13: Modelled cost over time of the motor breaking over the operational interval.

landing location.

While this single-fault injection can help one understand how the system behaves, a fault injection approach can be used to quantify the expected effects of a fault which could occur at different points in the simulation. This is shown in Figure 4.13. As shown, the cost is high in the ascent and forward flight phase ($134K-$184K) since the fault then leads to a crash, and low in the descent phase ($500), since the drone has already landed. Additionally, the cost increases in the middle of the forward flight phase since the system crashes far from its landing location, which incurs additional cost in the model. While these results are consistent with the results of the static model at the point in time considered in the model (forward flight), they also show how a higher-fidelity dynamic model model can elicit a more nuanced consideration of fault effects.

4.3.4 Hierarchical Representation and Analysis

Given the effects of failures in this system, it is important to consider how they can be mitigated through the component architecture. In the drone model, for example, one has the ability to consider whether to realize the AffectDOF function with a quadrotor, hexarotor, or octorotor architecture (a more complete exploration of these variables is provided in the example in Section 5.5.1). This example considers the quadrotor and octorotor architectures.

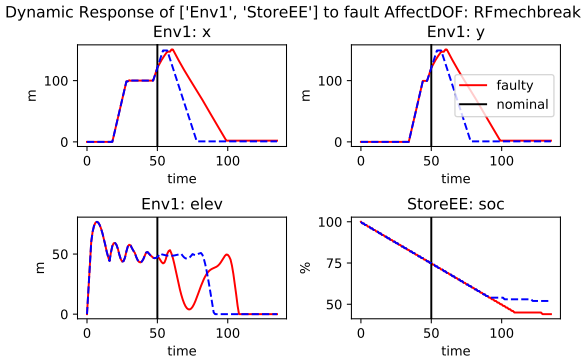


Figure 4.14: Fault behavior of drone with an octorotor architecture

To compare how each architecture adapts to faults, the dynamic behaviors over time can be plotted. When considering the break fault, the quadrotor architecture reacts identically to the fault as in Figure 4.6, since losing one motor causes the system to lose stability. However, when the drone has an octorotor architecture, the system behaves as shown in Figure 4.14, faltering due to lost thrust but ultimately recovering and landing in the desired location. Thus the octorotor architecture is more resilient to this fault scenario.

However, to make a decision about component architectures on the basis of resilience, the expected cost of all fault scenarios for both architectures must be compared and weighted against the operational and implementation costs. To quantify this cost, each of the faults associated with the realized function (AffectDOF) are injected in the model according to a sampling approach. In this case, while the octorotor component architecture mitigates a number of scenarios due to the increased system redundancy, it does not mitigate every fault (e.g. control errors), and in fact increases the chance of other faults (e.g., electrical problems that propagate to the battery) because the larger number of components provides more opportunities for the fault to occur. Statistics from this fault approach are shown in Table 4.3. As shown, while the number of scenarios increases in the octorotor architecture, the number of scenarios which lead to a crash (and overall crash rate) is much lower, resulting in a lower overall resilience cost. This shows how fmdtools can be used to assist resilient design decision-making in the early design process.

Table 4.3: Cost of rotor faults in each architecture

	Quadrotor	Octorotor
Number of Scenarios	104	208
Number of Crashes	46	24
Crash Ratio	0.44	0.12
Crash Rate	2.4e-6	0.8e-6
Resilience Cost	45565	19359

4.4 Example: Wildfire Response Model

This section illustrates how one can use the fmdtools software package in a complex modelling study with interacting dynamic components, environmental and design input parameters, and specialized analyses. One of the advantages of the fmdtools toolkit is that because its models are written in base Python, it is easy to extend to develop complex simulations with specialized model structures for the type of model being simulated. This case study exemplifies this through the simulation and visualization of grid-based fire model, which were developed by extending the base model definitions (enabling a grid matrix with properties for each grid point) and specialized analysis and visualization modules. This section provides a summarized description of this model, which developed as a part of the SMARt-STEReO (System Modeling and Analysis of Resiliency in STEReO) project at NASA Ames Research Center to model the effect of increased technological capabilities to wildfire response. Since the purpose of this section is merely to illustrate, the details have been truncated–full details are provided in [111], including background/context, simulation animations, and extensive verification of model behaviors under nominal and faulty conditions.

4.4.1 SMARt-STEReO Model Overview

The purpose of the SMARt-STEReO model is to simulate the performance and resilience of wildfire response at a high level to understand the effect of different operational concepts and strategies. Aerial support plays a major role in fighting wildfires. Airtankers, helicopters, and other aerial assets support the construction of fire-lines, gather data, and transport crews and equipment. However,

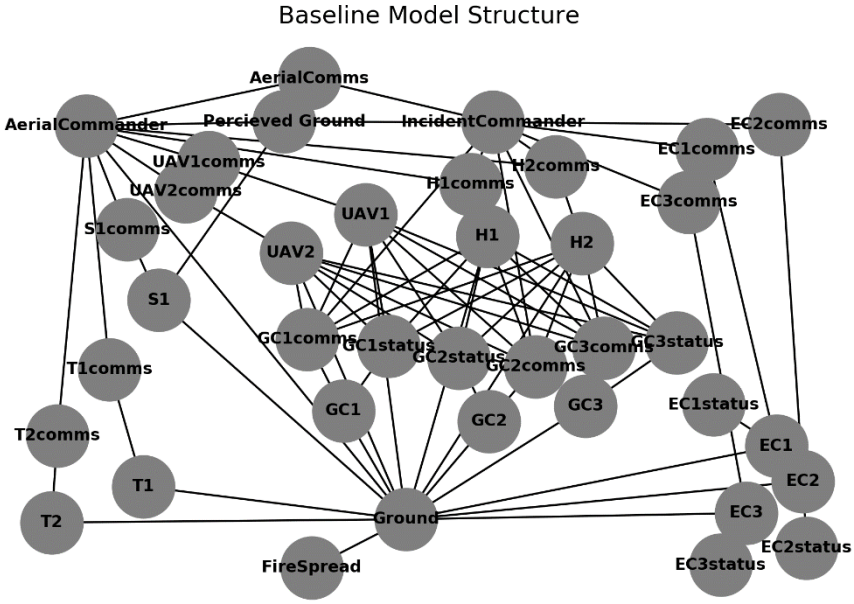


Figure 4.15: SMARt-STEReO model structure

without a model, it may not be clear how changes to the system (e.g., adding UAVs or more sophisticated communication protocols) will improve the performance or resilience of the system. The SMARt-STEReO model consists of a model of fire propagation and a number of interacting models for the different response assets which work to fight the fire. This overall structure is shown in Figure 4.15, which shows flow connections (i.e. data structures) between each function class (i.e., different behaviors which perform a task).

As shown, the model is made up of the:

- Fire model, which encompasses the `Ground` flow–which includes the fire-grid, a grid of points with the state of the fire fuel, flammability and propagation at a given time and the locations of assets on the ground (e.g., airports, ground bases, etc.)–and the `FireSpread` function, which propagates the fire to new locations at each timestep. The resulting fire propagation is defined by the given map and wind parameters, including the area (size of the map grid, number of points to use) and grid attributes comprising:
  - fuel (number of executions a fire is present on a pixel),

  - flammability (number of executions before an adjacent fire spreads to a pixel),
  - altitude (which modifies fire propagation speed),
  - fuel type (which modifies flame length),
  - flame length (which modifies fire heat production), and
  - obstacle (which determines where the fire cannot spread e.g. fire break)

  Each of these attributes are assigned at the beginning of the simulation using pre-determined patterns (uniform, uphill, downhill, etc.) or a random map generation (described in the next section). At a single execution of the fire model, the flammability of all grid points next to the fire is reduced by a certain amount until the attribute drops to zero, at which the point(s) are added to the fire. Additionally, fuel is reduced from pixels where the fire is burning and the fire is put out at pixels where the fuel has reached the threshold of zero. To enable fast speeds, this model is executed four times per model time-step.

- Ground crews (`GC1, GC2, GC3`), which perform land cuts to create firebreak on the fire-grid. The state of a particular ground crew is held in the corresponding `GCstatus` flow, which enables UAVs and Helicopters to change the state of the ground crews. Ground crews communicate with the incident commander and their assigned helicopters using `GCcomms` to share pickup/drop-off locations for constructing the fire-line.

- Engine Crews (`EC1, EC2, EC3`), which also perform land cuts to create firebreak. The difference between ground crews and engine crews is that ground crews are carried by helicopter while engine crews move on their own but can only access certain sides of the map–the right and lower sides in the baseline model. `ECstatus` is a flow which contains the state of the crew (following the convention used for ground crews) while `ECcomms` is a flow used to communicate cut locations with the Incident Commander.

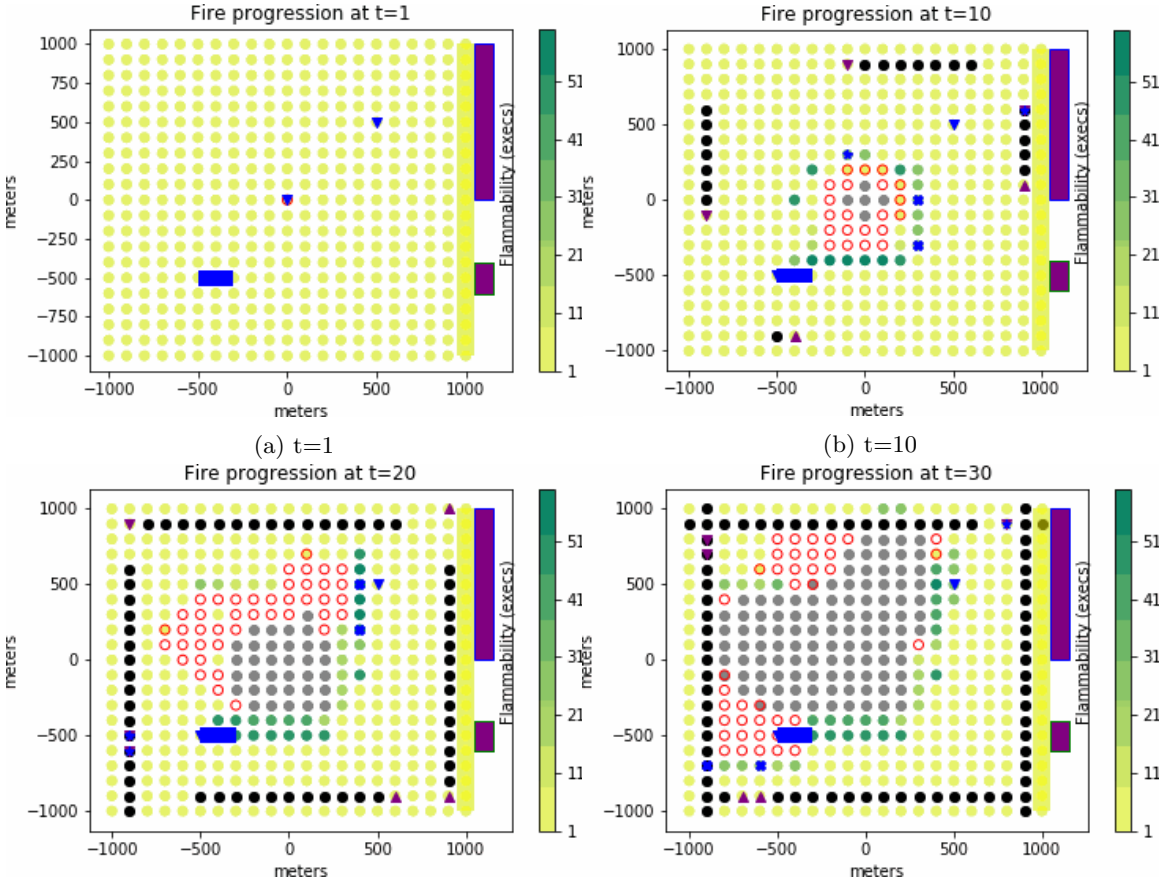
- Helicopters (`H1, H2`), which deliver ground crews to specific locations on the ground, perform water drops on/near the fire, and deliver supplies to ground crews. For each helicopter there

is an `HComms` flow which is used by the aerial commander to communicate drop locations when the helicopter is in drop mode.

- Supply UAVs (`UAV1, UAV2`), which re-supply groundcrews with supplies when they run out. While UAVs have a flow which communicates with the aerial commander, it is currently unused.
- Tankers (`T1, T2`), which drop retardant near the fire to slow it down. `TComms` is used to communicate drop locations and tanker readiness between the aerial commander and the tanker.
- The aerial commander/supervisor/lead plane (`AerialCommander`), which detects the current state of the ground in a particular part of the map and relays it back to the incident commander and assigns drop locations to the tankers and helicopters. `AerialComms` is a flow used to communicate the flight path of the aerial commander with the incident commander, as well as to receive the highest-threat parts of the fire to slow down with drops. `Perceived Ground` is a flow used to communicate the perceived state of the ground with the incident commander
- Surveillance planes/UAVs (e.g., `UAV1`), which also detect the current state of the map and relay it to the incident commander, updating a different part of the Perceived Ground flow. `Scomms` is used to communicate the desired flight path of the surveillance plane given the path of the lead plane.
- Incident commander (`IncidentCommander`), which uses the state of the ground to determine where to construct the fire-lines and send ground crews, and which edge of the fire the tankers should focus on.

4.4.2 Integration Demonstration and Parameters

When put together, these component models make up an integrated model of fire progression and response. This response can be seen in Figure 4.16. As shown, ground crews (purple upside-down triangles) and engine crews (purple right-side up triangles) attempt to enclose the fire on all sides



(a) t=1

(b) t=10

(c) t=20

(d) t=40

Figure 4.16: Nominal Simulation of stereo model

bycreating fire break while the tankers (each noted with a blue X) helicopters (blue stars) slow down the fire by increasing the spread time of particular pixels between the fire and gaps in the fire-line. However, while the main fire slowing and containment occurs because of the tankers and ground/engine crews, these behaviors are enabled by the function of the other modelled assets, specifically the surveillance of the fire by the lead plane and surveillance UAV (upside-down blue triangles), the relaying of ground crew and supplies to given locations by the helicopters and supply UAVs, the evaluation of threats to the fire-line by the incident commander, and the planning of drops by the lead plane. The response ends when the fire is either completely enclosed or the fire stops spreading.

However, this result only constitutes one output from the simulation at a single set of model parameters. In order for the model to generalize to different situations, and to be able to test different responses, the model must be parameterized over ranges of input scenario and response parameters. The operational response parameters are given in Table 4.4, with their ranges and baseline values. As shown, these parameters include both structural parameters (i.e., the number of assets used, which results in a different number of functions and overall model structure), effectiveness parameters (e.g., tanker drop size, rest per timestep), and strategic parameters (e.g., state information used, fireline priority). In the default setting, a small number of highly-effective tankers, helicopters, ground crews, engine crews and other assets are used to contain the small fire on all sides.

In addition to response parameters, there are parameters which can be changed to change the firefighting situation and conditions. These can be broken up into two kinds–parameters which change the fire grid distributions (e.g., how the fuel is allotted over the gird) shown in Table 4.6 and those which change the other input parameters in Table 4.5. While some of these parameters are not changed (grid size, spacing, etc), in general the baseline parameters and uniform map (i.e., the simulation in in Figure 4.16) are used to verify the model while the the parameters are varied over many simulations in order to ensure that the behaviors and function of the system generalizes to many scenarios. That is, a desirable response generalization would be for a capable set of aircraft and ground crews to be able to contain the fire over a variety of different maps that are within their

Table 4.4: Baseline Input Response Parameters

Parameter	Ranges	Baseline Value
Number of Tankers	0 or more	2
Tanker Drop Size	1 or more	3 pixels
Tanker Drops Per Refuel	1 or more	3
Tanker drop effectiveness	0 or more	22 execs
Number of Ground Crews	0 or more	3
Maximum Number of Supplies	1 or more	10 timesteps
Number of Engine Crews	0 or more	3
Rest Per Timestep	1 or more	5
Number of Helicopters	0 or more	2
Number of Supply UAVs	0 or more	1
Surveillance Lag	0 or more	0 timesteps
State Information Used	'all' or 'dist'	all
Danger Fire-line priority	0-1	0

Table 4.5: Additional Model Parameters

Parameter	Ranges	Default
Windspeed	0 or more	0
Wind heading	0-2pi	0
Sides accessible by engine crews	[] to [r, l, u, d]	[r, d]
Sides which must be protected to prevent imminent danger	[] to [r, l, u, d]	Right side
Grid Size	(any)	2000x2000
Grid Spacing	1+	100 m
Grid type	Uniform, random, uphill, downhill	uniform
hline Fire-line	0-grid edge	900
Initial Fire Location	Any valid grid point	(0,0)

capacity.

While the full demonstration of these parameters is out of the scope of this work (and is both the subject of further study and provided in more detail in [111]), some simulation results are provided here for demonstrative purposes.

One major consideration in aerial firefighting for the smart-stereo project is the effect of increased communication throughput (i.e. communicating the position and conditions of the fire on the ground). Currently, there are limitations on communication throughput as a result of aircraft-to-ground and aircraft-to-aircraft communications taking place over voice radios. To model the ability

Table 4.6: Random Map Generation Parameters

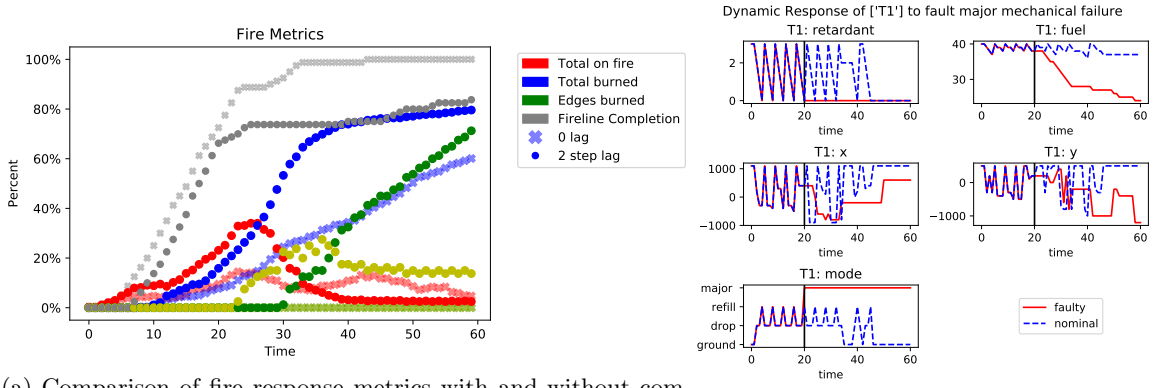
Attribute	Distribution	Parameters
Fuel	Uniform	(0, 60)
Fire	N/A	Only at (0,0)
Flammability	Uniform	(1,16)
Flame Length	Constant	0
Altitude	Constant	0
Fuel Type	Uniform	(1,16)
Obstacle	Binomial	(0.9: 0, 0.1: 1)

to pass information between the ground and aircraft faster, the communications lag parameter can be varied, which changes how quickly (in number of time-steps) the state of the fire grid is communicated between the surveillance aircraft and the incident commander on the ground. As shown in Figure 4.17, more lag results in a slower, less effective fire response where the fireline is eventually breached by the fire.

Another major consideration for aerial firefighting is the resilience of different responses to faults or adverse events that may happen during a mission. Since wildfire response is a highly uncertain, variable, and dangerous environment that nevertheless is very important to perform consistently to protect people and property, it is important that the response continues or recovers in spite of faults. Currently, in the SMARt-STEReO model, faults have been associated with each asset to assess the resilience of the response in these scenarios. Figure 4.17 shows an example of fault behavior propagation affecting a single Tanker's ability to perform its mission. As shown, the failure causes the tanker to no longer move, leading it to no longer be able to perform drops.

4.4.3 Discussion

The SMARt-STEReO model provides an integrated model of fire propagation and response. As with any model, there are assumptions and limitations which prevent the model from exactly matching real-life fire propagation. Nevertheless, this model can already provide some strategic insights for understanding aerial firefighting. As presented, a variety of different effects can be represented in the



(a) Comparison of fire response metrics with and without communication lag

(b) Propagation of mechanical fault in tanker

Figure 4.17: Example SMARt-STEReO simulation results.

modelling framework, including increased communications capabilities (e.g., from increasing state information or removing communications lag) and the response of the system to faults. Future work will systematically show how these model effects generalize across a number of input scenarios.

Finally, the smart-stereo model shows the fmdtools modelling toolkit can be used to model a highly complex, distributed system. The stereo model consists of over 1000 lines of code and has 9 unique interacting function classes (8 response functions and the fire model), communications and decision-making between response assets, and a number of parameters which can be modified to represent different firefighting situations. It is expected that complex use-cases like this will drive future feature development of fmdtools to enable efficient specification and simulation of uncertain parameters.

4.5 Conclusions

The fmdtools modelling framework presented in Section 4.2 enables the consideration of resilience in the design process by providing a set of model classes, simulation methods, and analyses that enable the consideration of risk through the functional, behavioral, and structural stages of the design process. As demonstrated in the drone example in Section 4.3, this enables one to analyze the system

as the design becomes more detailed. fmdtools additionally has a number of features that make it relatively unique among fault simulation toolkits, including Python-based model definition, undirected fault propagation, and simple model parameterization. This environment not only conducive to design, but may be easily modified and extended as needed to fulfill research needs, since all model attributes and simulation/analysis methods are defined in open-source Python code. These features are exemplified in the fire response model in Section 4.4, where the underlying model is built on the fmdtools function/flow classes with a number of extension classes to model and visualize the assets and fire propagation. This demonstrates how expressive and adaptable the fmdtools toolkit is to future resilient design applications.

While the usefulness of this work is apparent from the demonstrations shown here, there are a number of possible extensions that would increase its practicality in design. First, safety is an important aspect of resilience that has specific requirements to consider not explicitly taken into account in this work. That is, while this work is concerned with quantifying the costs of faults, safety procedures require one to quantify the overall cost of the entire set of failure scenarios, which often requires taking a deductive approach [12]. Future work should address this by providing an approach to identify top-level failure events and quantify the risk of those events. Additionally, while the models used in this work are deterministic, failures can often have probabilistic effects that must be taken into account to accurately quantify overall risk. Future work should incorporate probabilistic state transitions into the model to enable non-deterministic fault propagation. Finally, while defining models directly in Python code increases model expressiveness, it forces one to use a stand-alone model and may make the toolkit difficult to use without the relevant programming knowledge. Future work should provide an interface for defining these models in an existing modelling tool-chain or model formalism (e.g. AADL) so the same model used by other design and analysis processes can be used to model resilience.

Chapter 5: Optimizing Model Resilience in a Value-based Framework

5.1 Motivation

There are a number of ways one could use the modelling framework and decision-making approach in Chapters 3 and 4 to design a system to be resilient. The most simple approach would be to simply use the cost score to compare a few different options, as has been shown in the examples in Section 3.5 and 4.3. While this approach is suited to early design, when one typically has a few concepts to compare, it limits the designer's ability to explore the design space: one can only explore so many concepts before the exploration and comparison becomes tedious. Since the fault models presented in Chapter 4 already defined as easily-parameterizable software code, there is an opportunity, then, to use computation to enable a more comprehensive search of the space.

The aim of this chapter is to enable the reader to leverage optimization in the design process as shown in Figure 5.1. In this process, 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 Chapter 4 using the fmdtools framework (or using IBFM, see: [185]). The designer then associates a cost model with the various modes and model states and defines the changes to be explored within the optimization problem. This expected cost-based objective is then optimized by an appropriate algorithm by running the fault simulations (and design and/or operational models), calculating the expected costs, 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 variant.

However, deciding how to optimize a resilience-based optimization problem is difficult in early design because the design space is large, the variables to explore are heterogeneous, and there are complicated interactions between the design and operational cost models of the system and the resilience

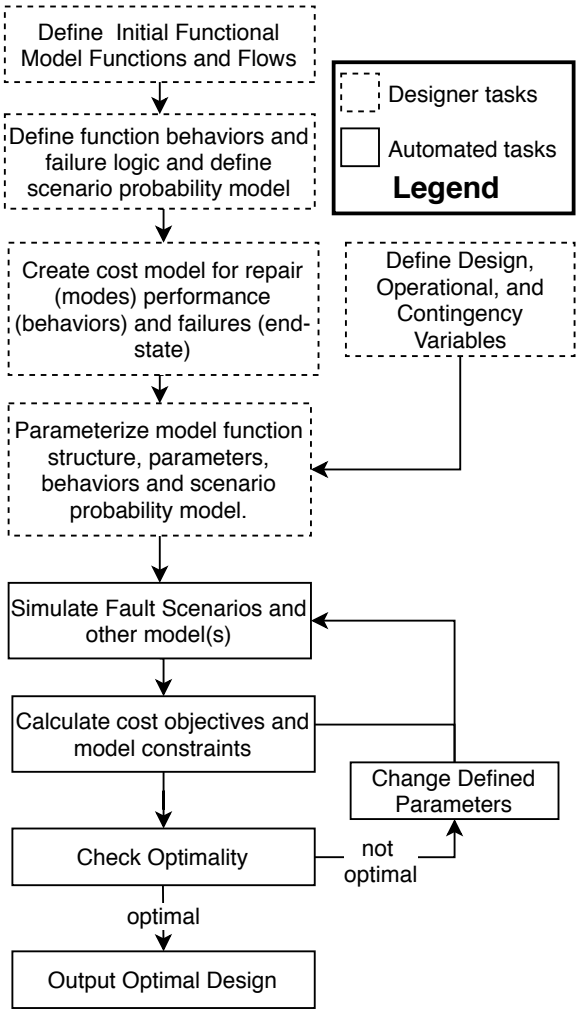


Figure 5.1: Overall framework of using optimization to achieve resilience in design.

model that make them difficult to optimize together. Consider the space of possible variables to explore shown in Table 5.1, which compiles newly-identified model variables with variables identified in previous function-based fault modelling and optimization approaches (see: [143, 256, 108, 188]). Optimizing each variable will require different solution strategies depending on the variable type. Structural design variables such as redundancy, function order, and flow paths will likely need to be approached using a graph grammar-based computational synthesis framework, as has been shown in [94, 266], since the function structure is fundamentally a graph. However, the internal-functional, operational, or contingency parameters may require different algorithms depending on how the variable is represented and parameterized. For example, a gradient-based search may be performed over the parameters of the assumed realization of the function (e.g. sizing, quality, etc), while an evolutionary or direct search method is used to determine the modes to recover or conditional logic. Additionally, as shown on right side of the table, a key difficulty common to many different design changes is predicting how a change will effect the design and operating costs of the functional model given the different couplings which may occur, for example, due to the working, constructional, and system interrelationships which are developed later in the design process. Such couplings are prevalent in the embodiment design stage of highly-coupled engineered systems, and provide significant challenges to design coordination [119] and present challenges to the validity of optimizing over a single variable type. As a result, it is desirable to optimize each (design, operational, and resilience) set of variables in an integrated framework that enables each variable to be optimized with the most effective corresponding solutions strategy.

While using optimization methods may make exploring the design space easier, it may be difficult to understand how use them effectively in the early design of resilience. To remove this ambiguity and clarify the problem, this chapter defines the resilience optimization problem, analyzes its attributes and presents frameworks to solve it in a systematic manner. To better understand the resilience optimization problem, this chapter formulates the integrated resilience optimization problem and presents architectures which may be use to solve the resilience optimization problem in a systematic way, considering the interacting design, operational, and resilience models in Section 5.2. Examples

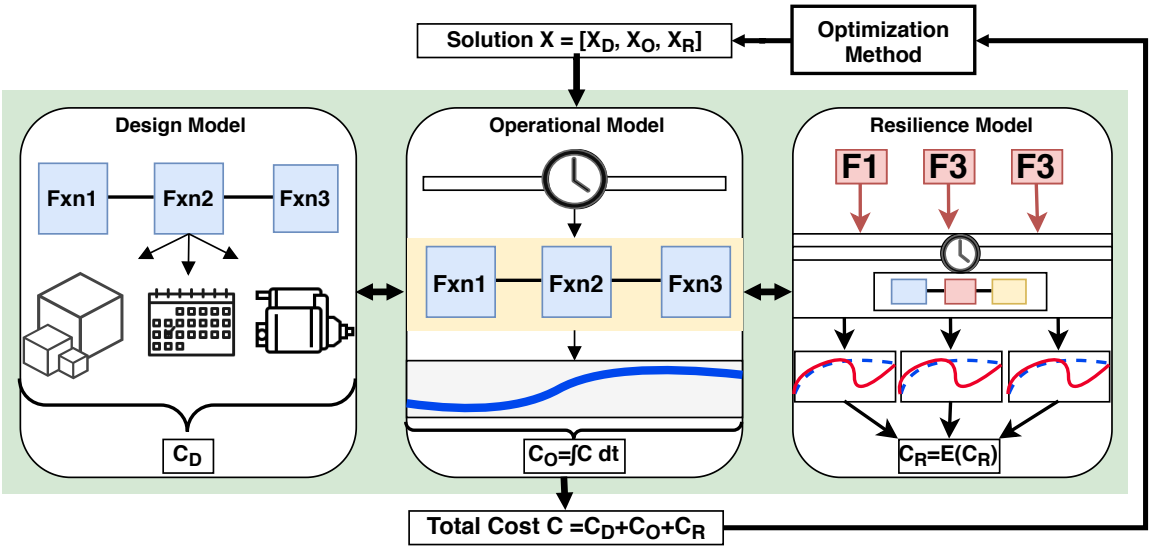


Figure 5.2: Integrated Resilience Optimization framework pursued in this work where design, operational, and resilience variables are jointly optimized.

then provide both a demonstration of the overall resilience optimization framework and a study of the methods and architectures presented. The monopropellant system example in Section 5.4 first demonstrates the use of resilience optimization in the context of an overall concept selection process for a monopropellant system, showing the advantage of considering the optimization early in the design process. The drone optimization example in Section 5.5 then demonstrates the integrated design, operations, and resilience optimization framework and compares the architectures presented for this in Section 5.2 in terms of effectiveness and computational cost. Section 5.6 then briefly reflects on the contribution and examples presented in the chapter. The decomposition approach presented in Section 5.3 is demonstrated in Section 6.4.

5.2 Problem formulation and statement of architectures

In the Integrated Resilience Optimization formulation of the resilience-based design problem shown in Figure 5.2, the features and architecture, nominal control and operations, and contingency management of the system over a set of fault scenarios are optimized using a single cost function which

Table 5.1: Design , operational, and resilience variables which may be considered to optimize resilience, their related costs, difficulties to optimization, and value.

Variable Type	**Design Change**	**Modelling Difficulties**	**Value**
Design-Structural	New Structure	Determining context and purpose, necessary functional relationships, behavior of functions in new contexts.	Most impact and ability to explore novel solutions.
	Redundancy/Number of Functions [143]	Predicting the effect of potential performance couplings on operational costs.	Reduces impact of individual failures. Enables behavioral consideration of entire redundant function chains, rather than those internal to a function.
	Function Order [143]	Predicting how function order might change design and operating costs. Determining behavioral impact of different function order.	Potentially inexpensive solution to lowering risk or reducing failure effects.
	Routing alternative flow paths (e.g. unused waste flows as inputs) [143]	Determining if flow path's effect on function behavior given a function's flow input requirements.	Ability to create resilience with less inherent cost increase than in other strategies (e.g. redundancy, excess capacity).
Design-Functional	Component Redundancy	Predicting effect of potential performance couplings.	Easy to model and optimize in certain situations. Enables consideration of redundancy without changing model structure.
	Assumed Realization/ Function Resources [188]	Potential internal and external compatibility couplings	Ability to represent trade-off between cost and quality (mode probabilities and costs as well as function costs).
	Function Modes	Couplings with assumed realization.	Ability to represent differences in behaviors of functions.
	Conditional Logic [256]	Predicting design cost of flexibility required to allow different decisions to be made.	Enables representing the response of control systems and built-in robustness compensating for failures as well as sacrificial subsystems, etc.
Operations	Mission Profile	Representing profile in all scenarios when there may be multiple possible missions.	Ability to determine more or less risky operational settings.
	Operational Margin	Modelling trade-off between margin and performance/efficiency	Ability to determine how to operate the system to be more or less prone to hazards.
Contingency/ Resilience	Modes to recover	Computational expense in determining recovery in every scenario.	Ability to represent resilient operational decision-making and repair.
	Maintenance and Health Management [143] [108]	Determining effect on failure probability given time representation. Difficult to model with fault propagation.	Ability to represent operational ability to lower fault probability.
	Emergency procedures and recovery policy	Predicting function of emergency procedures in hazardous and uncertain scenarios	Can reduce impacts of fault scenarios. Often needed for a safe design.

follows the form provided in Chapter 3. In a generic form, this may be stated as:

$$
\begin{aligned}
& min_{\mathbf{x_D},\mathbf{x_O},\mathbf{x_R}}\ C_D(\mathbf{x_D}) + C_O(\mathbf{x_D},\mathbf{x_O}) + C_R(\mathbf{x_D},\mathbf{x_O},\mathbf{x_R}) \\
& where\ C_R = \sum_{s\in S} n * r_s * C_s(\mathbf{x_D},\mathbf{x_O},\mathbf{x_R}) \\
& s.t.\ g_D(\mathbf{x_D}), g_O(\mathbf{x_D},\mathbf{x_O}), g_R(\mathbf{x_D},\mathbf{x_O},\mathbf{x_R}) \leq 0 \\
& h_D(\mathbf{x_D}), h_O(\mathbf{x_D},\mathbf{x_O}), h_R(\mathbf{x_D},\mathbf{x_O},\mathbf{x_R}) = 0
\end{aligned}
$$

where C_D, h_D, and g_D are design cost objective and constraints (which are the result of the design cost model in terms of design variables $\mathbf{x_D}$), C_O, h_O, and g_O are the operational cost model in terms of the design variables and operational variables $\mathbf{x_O}$, and C_R, h_R, and g_R are the resilience cost in terms of the design, operational, and resilience variables $\mathbf{x_R}$. Additionally, while the design and operational cost models may be of arbitrary form, the resilience cost model is given as the expected cost of the set of scenarios S with cost C_s and per-use rate r_s over the number of uses of the system n (n and r_s may also be given as functions of design, operational, and resilience variables if needed).

This problem has a repeated and exploitable structure which can be used to organize the problem in terms of optimization architectures. Since each subsequent model is a function of the variables in the previous model, it is convenient to model each subsequently (design, then operations, and then resilience). To describe these optimization architectures, this section will use the notation (e.g., for decision and response variables) and extended design structure matrix specification (see Figures 5.3-5.6) of Lambe and Martins [161, 178]. When using the multidiscipline-feasible optimization architecture (see [178]), the equality constraints h are substituted into the model such that the set of variables $\mathbf{x}$ may be considered in two sets–the decision variables $\mathbf{x}$ and the response variables $\mathbf{y}$. This enables one to easily leverage optimization in the context of a simulation, where there are defined input parameters which can be changed in the simulation (e.g., lengths, widths, materials, etc.) and output responses which result (e.g., mass, performance). Additionally, in this formulation there is no feedback between the disciplines in terms of constraints (that is, h_D does not depend on operational

or resilience variables). As a result, achieving variable consistency with coupled analyses–the main difficulty associated with using a multidiscipline-feasible optimization architecture [178]–is not an issue. Thus, the exploration of architectures undertaken in this work starts by stating the problem in this form:

$$
\begin{aligned}
&min_{\mathbf{x_D},\mathbf{x_O},\mathbf{x_R}}\ C_D(\mathbf{x_D},\mathbf{y_D}) + C_O(\mathbf{x_D},\mathbf{x_O},\mathbf{y_D},\mathbf{y_O}) + C_R(\mathbf{x_D},\mathbf{x_O},\mathbf{x_R},\mathbf{y_D},\mathbf{y_O},\mathbf{y_R}))\\
&\quad where\ C_R = \sum_{s\in S} n * r_s * C_s(\mathbf{x_D},\mathbf{x_O},\mathbf{x_R},\mathbf{y_D},\mathbf{y_O},\mathbf{y_R})\\
&\quad\quad s.t.\ g_D(\mathbf{x_D},\mathbf{y_D}), g_O(\mathbf{x_D},\mathbf{x_O},\mathbf{y_D},\mathbf{y_O}), g_R(\mathbf{x_D},\mathbf{x_O},\mathbf{x_R},\mathbf{y_D},\mathbf{y_O},\mathbf{y_R}) \leq 0
\end{aligned}
$$

where $\mathbf{y_D}$, $\mathbf{y_O}$, and $\mathbf{y_R}$ are all response variables of the design, operational, and resilience models. This formulation, also referred to as the **all-in-one optimization** structure, is shown in Figure 5.3 using the extended design structure matrix notation. As shown, at each iteration of the optimization a new set of design, operational, and resilience variables is provided and the design, operational, and resilience analyses are each run in sequence such that the design responses feed into the operational model and the operational responses feed into the resilience model. These models provide the design, operational, and resilience costs which are then used as the objective and constraint values in the optimization solver for each iteration.

The next sections describe additional architectures (sequential optimization, bilevel optimization, and a lower-level decomposition strategy) which may be used to solve this problem by splitting the optimization into an upper-level design and operational analysis optimization and a lower-level resilience analysis optimization. While combined optimization of design and operations has been explored previously in the co-design literature, the novel contribution to the optimization of resilience provided here is the joint optimization of the resilience policy with other considerations.

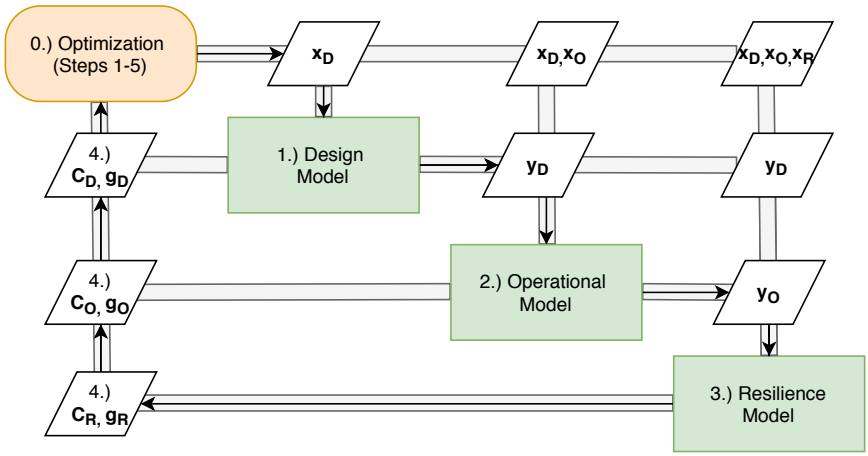


Figure 5.3: Extended Design Structure Matrix of the All-in-One optimization architecture.

5.2.1 Sequential Optimization

In the sequential optimization architecture, the optimization is split into two stages, as shown in Figure 5.4. In the first stage, the design and operational costs are optimized in terms of design and operational variables, using the same multidiscipline-feasible approach used in the all-in-one model. After that optimization is completed, the lower-level optimization is performed on the resilience variables using the optimal design and operational variables found in the upper-level optimization. Stated mathematically, the top-level optimization solves the following optimization problem:

$$min_{\mathbf{x_D},\mathbf{x_O}}\ C_D(\mathbf{x_D},\mathbf{y_D}) + C_O(\mathbf{x_D},\mathbf{x_O},\mathbf{y_D},\mathbf{y_O})$$

$$s.t.\ g_D(\mathbf{x_D},\mathbf{y_D}), g_O(\mathbf{x_D},\mathbf{x_O},\mathbf{y_D},\mathbf{y_O}),$$

which optimizes design and operations of the system concurrently in terms of design and operational variables. The lower-level optimization is then:

$$min_{\mathbf{x_R}}\ C_R = \sum_{s \in S} n * r_s * C_s(\mathbf{x^*_D}, \mathbf{x^*_O}, \mathbf{x_R}, \mathbf{y^*_D}, \mathbf{y^*_O}, \mathbf{y_R})$$

$$s.t.\ g_R(\mathbf{x^*_D}, \mathbf{x^*_O}, \mathbf{x_R}, \mathbf{y^*_D}, \mathbf{y^*_O}, \mathbf{y_R}) \leq 0$$

which optimizes resilience costs in terms of resilience variables given the first-stage optimal design and operational decision and response variables $\mathbf{x^*_D}, \mathbf{x^*_O}, \mathbf{y^*_D}, \mathbf{y^*_O}$

This problem formulation assumes that the design/operational and resilience costs are, in effect, independent–that the design and operational variables have no effect on the resilience of the system. As a result, one would not expect it to find a global optimum when upper-level variables effect resilience costs, and if there are lower-level constraints which depend on the upper-level variables it may not even converge to a feasible design. However, the fact that each level is only optimized once results in much lower computational cost than in the other approaches.

5.2.2 Bilevel Optimization

In the bilevel approach, the optimization problem is again split into an upper-level design and operational optimization and a lower-level resilience optimization, as shown in Figure 5.5. However, unlike the sequential approach, in the bilevel approach the lower-level resilience cost is used in the upper-level optimization. As a result, each iteration of the upper level results in a lower-level optimization of the resilience variables for that design and operational policy. Thus, at each value of design and operational variable, the lower-level optimization is run over the resilience model to find the optimal cost and feasibility that the resilience variables can achieve at that set of design/operational variables. Stated mathematically, the upper-level optimization solves the following optimization

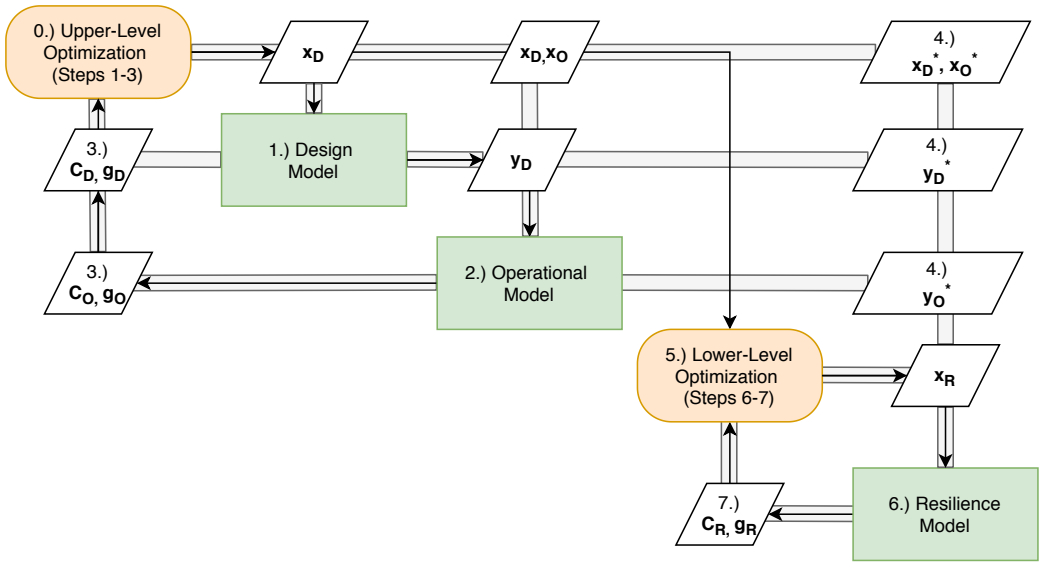


Figure 5.4: Extended Design Structure Matrix of the sequential optimization architecture.

problem:

$$min_{\mathbf{x_D},\mathbf{x_O}}\ C_D(\mathbf{x_D},\mathbf{y_D}) + C_O(\mathbf{x_D},\mathbf{x_O},\mathbf{y_D},\mathbf{y_O}) + C_R^*(\mathbf{x_D},\mathbf{x_O},\mathbf{y_D},\mathbf{y_O})$$

$$s.t.\ g_D(\mathbf{x_D},\mathbf{y_D}), g_O(\mathbf{x_D},\mathbf{x_O},\mathbf{y_D},\mathbf{y_O}), g_R^*(\mathbf{x_D},\mathbf{x_O},\mathbf{y_D},\mathbf{y_O}) \leq 0$$

where $C_R^*(\mathbf{x_D},\mathbf{x_O},\mathbf{y_D},\mathbf{y_O})$ and $g_R^*(\mathbf{x_D},\mathbf{x_O},\mathbf{y_D},\mathbf{y_O})$ are the responses of cost and constraint functions which come from optimizing the lower-level model at each iteration. This lower-level optimization is then:

$$min_{\mathbf{x_R}}\ C_R = \sum_{s\in S} n * r_s * C_s(\mathbf{x_D},\mathbf{x_O},\mathbf{x_R},\mathbf{y_D},\mathbf{y_O},\mathbf{y_R})$$

$$s.t.\ g_R(\mathbf{x_D},\mathbf{x_O},\mathbf{x_R},\mathbf{y_D},\mathbf{y_O},\mathbf{y_R}) \leq 0$$

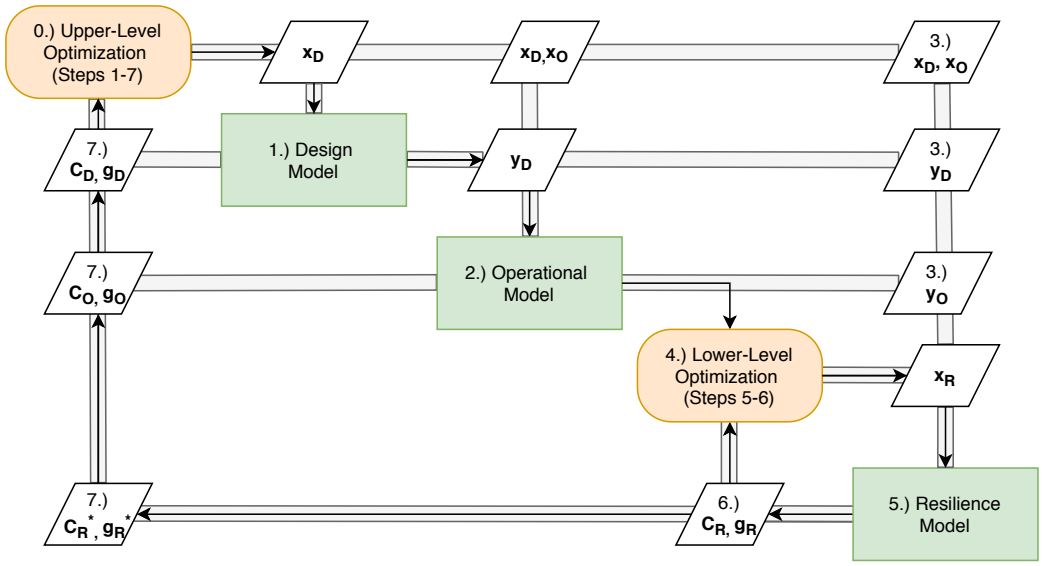


Figure 5.5: Extended Design Structure Matrix of the bilevel optimization architecture.

which optimizes the resilience costs in terms of resilience variables given values for the design and operational decision and response variables. One of the difficulties with the bilevel approach is computational cost. Because each upper-level iteration results in a full optimization of the lower-level variables, it may be difficult to solve using traditional methods (e.g., the finite difference method where finding a search direction requires taking at least one evaluation per variable). Another property of the bilevel approach is that it enables one to cut out infeasible parts of the design space by only conducting the lower-level optimization when the upper-level design is feasible. While a similar approach is achievable in the all-in-one structure by not evaluating the resilience model if the design/operational models are infeasible, in an exhaustive search the all-in-one structure must still search the full set of lower-level variables for a design, leading to wasted iterations.

5.2.3 Lower-level decomposition

One property of the lower-level resilience model in the bilevel and sequential formulations is that the scenario costs C_s are independent, since each scenario represents a separate model evaluation at a

particular realization of uncertain variables. Thus, if the problem is truly unconstrained at the lower level (i.e., no g's relating simulation results, such as a reliability or failure probability requirement and each decision variable x_R is able to be mapped to a specific set of fault scenarios), each resilience variable can be optimized independently in its own problem. Additionally, this approach can only be used with the bilevel or sequential approaches where the upper-level design and operational variables are held constant during the lower-level optimization. In this approach, shown in Figure 5.6, the lower-level optimization of resilience variables is split into multiple sub-problems. Thus, the cost of resilience is found by optimizing the problem:

$$C_R^*(\mathbf{x_D}, \mathbf{x_O}, \mathbf{y_D}, \mathbf{y_O}) = C_{Rr}(\mathbf{x_D}, \mathbf{x_O}, \mathbf{y_D}, \mathbf{y_O}) + \sum_{c \in r} min_{\mathbf{x_{Rc}}} C_{Rc}(\mathbf{x_{Rc}}, \mathbf{x_D}, \mathbf{x_O}, \mathbf{y_D}, \mathbf{y_O})) \tag{5.1}$$

where c is the index of the resilience variable x_{Rc} corresponding to the set of scenarios S_c and resulting sum total cost C_{Rc}, and the cost C_{Rr} is the residual cost of scenarios not related any resilience variables. While this term is not necessary to find the optimized set of lower-level variables, it is necessary to give an accurate response from the lower-level model to the upper-level model for the resilience costs. As such, it must be run at least once per optimization of the lower-level model. This residual cost has the form $C_{Rr} = \sum_{s \notin S_c \forall c \in r} C_s(\mathbf{x_D}, \mathbf{x_O}, \mathbf{y_D}, \mathbf{y_O})$ while the resilience cost for each variable has the form:

$$C_{Rc}(\mathbf{x_{Rc}}, \mathbf{x_D}, \mathbf{x_O}, \mathbf{y_D}, \mathbf{y_O}) = \sum_{s \in S_n} n * r_s * C_s(\mathbf{x_c}, \mathbf{x_D}, \mathbf{x_O}, \mathbf{y_D}, \mathbf{y_O}) \tag{5.2}$$

This decomposition reduces the space of the optimization problem since each sub-problem is of much lower dimensionality than the combined problem. Additionally, because each optimization only occurs over a subset of the full number of scenarios, there is a reduced iteration computational cost. However, it can only be used in situations when the lower-level model is unconstrained. While this is the case when one only considers the cost of scenarios in the optimization, it is not the case when there are additional constraints (e.g., overall failure probability requirements).

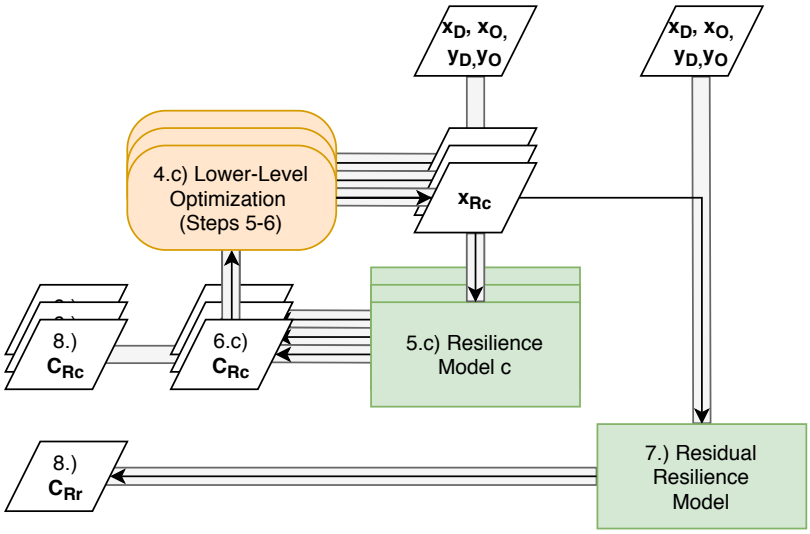


Figure 5.6: Detail of Design Structure Matrix of the Bilevel approach with a Lower-level decomposition

5.3 Decomposition Approach for Preventative Measures

A decomposition strategy can also be leveraged at the design level, provided the resilience optimization problem fulfills certain properties, when dealing specifically with fault prevention strategies. Consider a version of the resilience optimization problem with the form:

$$min_{\mathbf{x_n}} C = C_D(\mathbf{x_n}) + C_O(\mathbf{x_n}) + C_R(\mathbf{x_n}) \tag{5.3}$$

where the design vector $\mathbf{x_n}$ is a vector of variables related specifically to the design and operations of function n. To consider this cost sum at the local functional level, it needs re-written in terms of the variables related for that function. This decomposed, parameterized cost sum has the form:

$$C = \sum_{n \in N} (C_{dn}(x_n) + C_{on}(x_n)) + \sum_{n \in N} \sum_{s \in S_n} (N_s(x_{n \sim s}) * C_s) \tag{5.4}$$

where n is a function in the set of functions N, C_{dn} is the design cost of that function, C_{on} is the operational cost of that function, x_n are the variables in function n related to risk and cost, $x_{n \sim s}$ is a

risk-related variable in function n associated with the scenario s in the set of all scenarios associated with the function n, N_s is the expected number of times the scenario will occur of the life of the product (or, if the event may only occur once, the probability) and $C_{s\sim m}$ is the cost of the scenario.

This decomposed form relies on three assumptions about this future system, that:

- Design and operational cost may be parameterized in each function. For this to be true, there should be no significant or sensitive cost-related design and operational parameter couplings. While not true of many real-world systems, this assumption is appropriate to design in the early design phase, where the design is relatively unconstrained and preliminary in nature. However, if disciplinary coupling is a concern, approaches have been developed to incorporate the cost of coupled component parameters to allow for optimal coordination of design work [140] that could be used in conjunction this framework to determine the local design and operational costs (although it would require more communication between disciplines).

- Scenarios only result from faults in single functional faults. This neglects all faults that originate in multiple functions simultaneously. The consideration of these faults must then take place separately, or the function-based fault model must be developed in a way that allows them to originate from a single function block.

- Either the design changes explored do not effect the cost of a given scenario, or local scenario costs can be constructed for a function's local failure effects that correlate with the overall cost of failure in the system for that given scenario (functionality which may or may not be provided, depending on risk modelling software used)

These assumptions limit the ability of Equation 5.4 to be used on all problems as a comprehensive risk-based design metric, especially in resilience-based design scenarios where one wishes to change the system's fault response. However, the value this form of the equation is that it enables the designer to make local design decisions with very little computational effort, which is important for an efficient, parallelized design process. This will be further discussed in Section 5.5.3.

A further simplification may be used in the case where only single faults occur in each function

(which may be more convenient to consider in early design):

$$C = \sum_{n \in N} (C_{dn}(x_n) + C_{on}(x_n)) + \sum_{n \in N} \sum_{m \in M_n} (N_m(x_{n\sim m}) * C_{s\sim m}) \tag{5.5}$$

where m is a fault mode in the set of fault modes M_n in function n, N_m is the expected number of times the mode is to occur, $x_{n\sim m}$ a the risk-related variable associated with the mode m in function n, and $C_{s\sim m}$ is the cost of the scenario associated with that mode. This equation may be useful when the toolkit can only inject single faults within functions, or when a simple preliminary analysis is desired; however, Equation 5.4 should be used in the general case to allow for scenarios resulting from multiple fault modes to be considered.

5.3.1 Optimization Approach

Since the overall objective has been decomposed such that each part of the sum is related only to a single function, the optimization framework is shown in Figure 5.7. In this framework, the expected cost of risk is associated entirely independently within each function. This cost C_n is:

$$C_n = C_{dn}(x_n) + C_{on}(x_n) + \sum_{s \in S_n} (N_s(x_{n\sim s}) * C_s) \tag{5.6}$$

This can be optimized using a method applicable to the parameterization used (The same can be performed for the single fault version (Equation 5.5) except with $C_{s\sim m}$ instead of C_s). In practice, these problems may be trivially simple or of low dimensionality, enabling exhaustive searches to be used. Additionally, because all of the objectives occur at the local level, no subsequent runs of the system-level risk model need to be performed, which enables quick computational performance if needed. Alternatively, if this process is performed manually by a designer, it may prove to be a simple heuristic to choose between different design options.

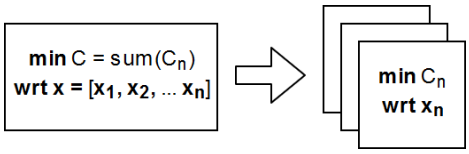


Figure 5.7: Decomposed optimization framework.

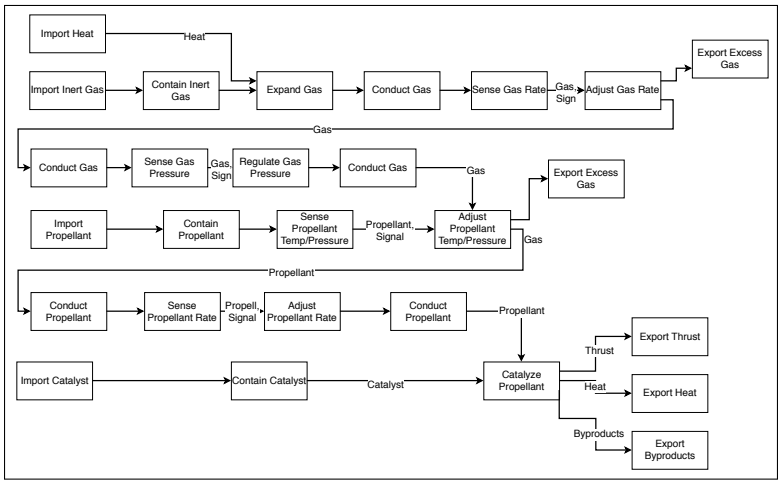


Figure 5.8: Functional Model of Base Monopropellant System

5.4 Example: Design and Optimization of Monopropellant System

To demonstrate the value of using optimization in the early design process, this section applies the RISCS expected cost score (see Section 3.3) to the design of a monopropellant orbiter, using a concept generation process to create design variants with different recovery features (see Figure 5.12a-5.14), and an algorithm to optimize the controlling functions within one of the variants (see Figure 5.14).

Monopropellant system design was previously considered in [143] and introduced as an example system in [267]. Monopropellant propulsion systems are named as such because they do not require a separate oxidizer, and are commonly used in spacecraft for attitude control, and sometimes to provide primary thrust. The functional model of the monopropellant system is shown in Figure 5.8. Heat is applied to an inert gas to expand, and the gas is regulated to an appropriate temperature and pressure. The expanded gas then pushes a propellant over the catalyst. As propellant passes over the catalyst, it reacts, resulting in thrust.

Table 5.2: Cost flow state matrix for the monopropellant thrust function, in billions.

Effort Health	Rate Health				
	Zero	Low	Nominal	High	Highest
Zero	4.5	4	3.5	4.25	5
Low	4	2.5	1.0	0.75	5
Nominal	3.5	1.0	0	1.0	5
High	4.25	0.5	1.0	2.5	5.5
Highest	5	5	5	5.5	5.5

Table 5.3: Generated Monopropellant Designs over Different Mission Utilities.

	Ctr1 Low	Ctrl1 High	Ctrl2 Low	Ctrl2 High	Ctrl3 Low	Ctrl3 High	Ctrl4 Low	Ctrl4 High
Feature Used	0	0	1	1	1	1	0	1
Feature Cost	550000	5000	50000	50000	50000	50000	2050000	5000

The overall value generated by this system is a result of the quality of the thrust function, and the costs are a result of any design costs incurred by each function. In the formulation of the RISCS score considered in this application, only these trade-offs are considered for simplicity to illustrate the approach. As a result, only a few components must be considered in the cost function–design and failure state costs–since operational costs are assumed to be constant between concepts. Additionally, because the operational context of the system is 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. The resulting cost function is:

$$\sum_{n \in N(\vec{x})} C_n(\vec{x}) + \sum_{e \in E} P_e * (\bar{C}_l[\vec{s}_{f(\vec{x}),l}]) \tag{5.7}$$

where the notation is consistent with that outlined in Section 3.3, 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), $\bar{C}_l$ is the cost matrix for the thrust function (the only desired output flow) which has values shown in Table 5.2 which are again scaled to consider different failure costs, and $\vec{x}$ is the given design.

5.4.1 Optimization of Controlling Functions

To demonstrate the value of using optimization in the early design process, this section applies the RISCS function (see Section 3.3) to the optimization of controlling functions within the functional model of the variant of the monopropellant system shown in Figure 5.14. 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 high-level requirements for 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`. When the system is realized, these might be manifested as 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 5.9: Example controlling function conditions and modes.

To illustrate, in the function definition shown in Figure 5.9, the modes `EqualControl`, `IncreaseControl`, and `DecreaseControl` each refer to behaviors in which the controller 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`).

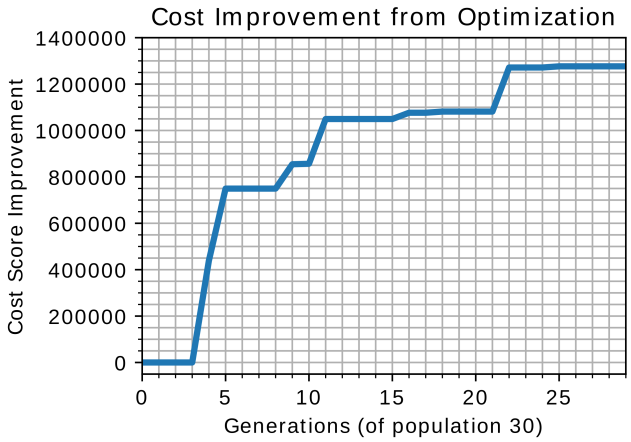


Figure 5.10: Cost optimization of the functional model using the evolutionary algorithm, showing how value can be increased using the presented optimization framework.

This problem is readily encoded as an integer vector and can be solved using an evolutionary algorithm following the general optimization framework shown in Figure 5.1. Although many integer programming approaches are possible (indeed, a direct search method may be more efficient, and the space of designs is small enough to be searched with a brute force method), the evolutionary algorithm was used in this paper in which 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.

As can be seen in Figure 5.10, the use of this algorithm increases the overall RISCS significantly from the baseline design cost. This results in a design shown in Table 5.3. As can be seen, while expensive recovery options are avoided (such as attempting to increase the final flow of propellant in Controller 4), less expensive recovery features are added in order to achieve the best RISCS score. This demonstrates the ability of the RISCS score to integrate with an optimization framework to systematically explore the space of design variants.

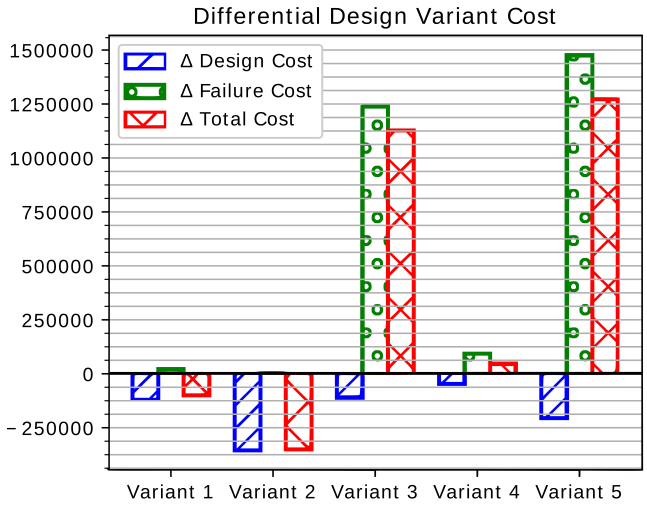


Figure 5.11: Differential costs of design variants based on fault simulation.

5.4.2 Comparing Model Structures

This section applies the RISCS function to compare different design concepts for adding resilience to the monopropellant system. Each of these variants are shown here, and were constructed by adding additional functions, conditions, and behaviors to the baseline IBFM model to account for the added resilient feature. The design variants considered were:

1. **Redundant Gas Tanks:** The `Contain Inert Gas` and `Expand Gas` functions are made redundant, as shown in Figure 5.12a.

2. **Redundant Thrusters:** The `Contain Catalyst` and `Catalyze Propellant` functions are made redundant, as shown in Figure 5.12b.

3. **Auxiliary Heat-recovery system:** An auxiliary system powered by the heat from the combustion from the propellant is added to expand the gas in case the heat source is lost, as shown in Figure 5.13a.

4. **Redundant Pressure Regulators:** An additional pressure regulator is added which activates when the operating pressure regulator fails, as shown in Figure 5.13a.

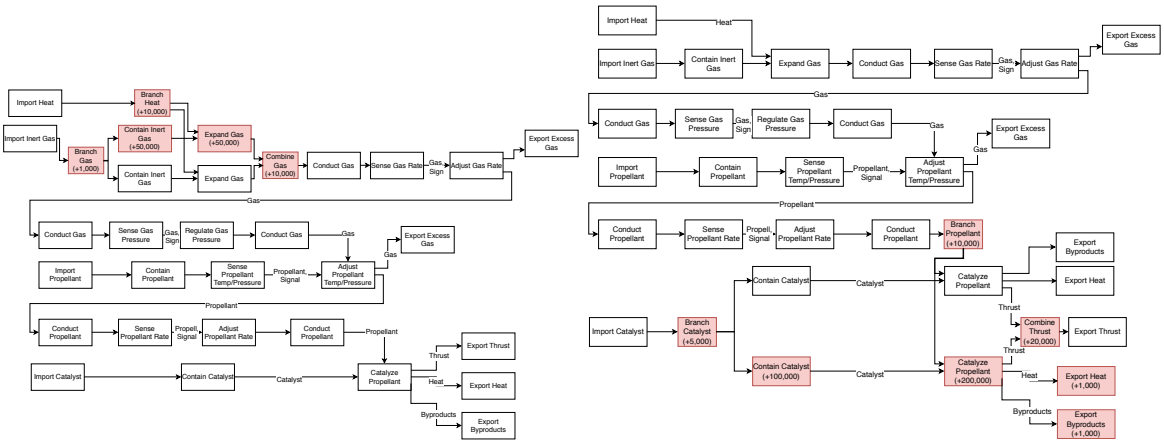
5. **Optimized Control Features:** The control features optimized in Section 5.4.1 are used in the design, as shown in Figure 5.13b.

As can be seen in the functional models for each design variant, each resilient features adds initial design cost. In order to calculate the design scoring in Equation 5.7, however, each model must be run to determine the resulting failure costs. The results for failure costs resulting from each of these model runs is shown in Figure 5.11, along with the initial design costs and total costs. For clarity, these are displayed as differential costs from the baseline, to show which features "pay off" by reducing failure costs above their increase in design cost, and which do not.

As shown in Figure 5.11, as simulated in the model, Variant 1 and Variant 2 (redundant gas tanks and redundant thrusters, respectively), are not worth their design cost because they negligibly improve the overall failure cost. This is because, in the model, these systems have a low probability of faults and relatively high design cost. On the other hand, the features in Variant 3, 4, and 5 do pay for themselves in terms of failure cost. In Variant 3, because the `Import Heat` function was modeled with a relatively high fault probability, the heat recovery feature was able to reduce the impact of this fault substantially. In Variant 4, the redundant pressure sensor is able to increase the scoring function simply because the design feature (a sensor) is relatively inexpensive, even though the change in failure cost is low. Finally, in Variant 5, while the design feature is relatively expensive, it is able to reduce a large amount of failure cost, allowing the feature to justify itself in terms of the cost score.

5.4.3 Discussion

The previous sections showed the optimization of a monopropellant system's recovery policy and required flexibility (Section 5.4.1) and the elaboration and selection of design features (Section 5.4.2). While this demonstrated the general usefulness of expected cost scoring for concept selection and resilience optimization in design, it also highlights an important challenge for this process. As shown, in the concept selection process, the design with optimized control and recovery features achieved



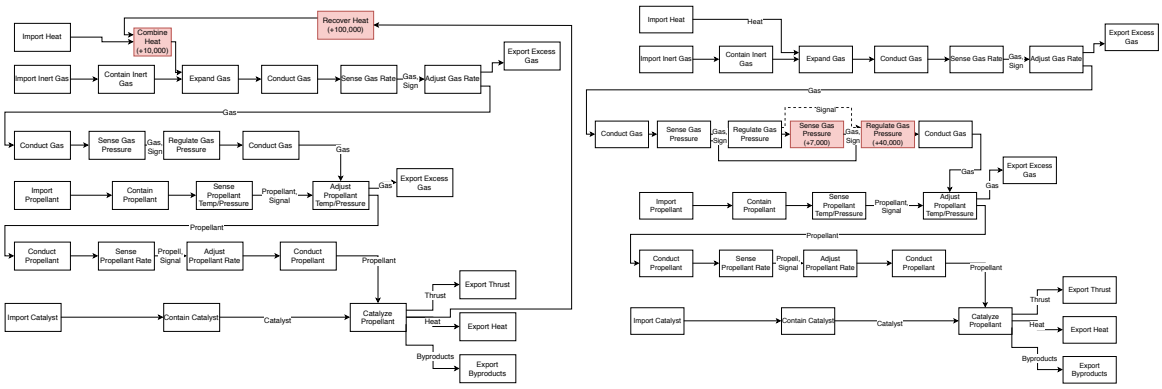
(a) Design Variant 1: Redundant Gas Tanks. (b) Design Variant 2: Redundant Thrusters.

Figure 5.12: Design Variants 1 and 2.

better overall value than the other design concepts. Thus, it is important to recognize the potential of contingency management in design, for two reasons. First, it may be that certain faults can be prevented using contingency management without costly design features or operational effects, so considering contingency management early can enable one to determine adding expensive design features is necessary. Second, a fair comparison of the resilience of different design concepts requires knowing how each can be reconfigured to adapt to faults, and while some concepts may not see improvement from contingency management, others may become much more resilient. This is why it is important to consider design, operations, and contingency management in a single optimization framework.

5.5 Example: Drone Optimization and Architecture Comparison

To demonstrate the utility of using an integrated resilience optimization framework in design and compare the optimization architectures introduced in Section 5.2, this section considers a multirotor drone case study representative of a typical early resilience-based design scenario. This model has been implemented in the fmdtools software package [120] and is publicly-available in the examples



(a) Design Variant 3: Heat Recovery System. (b) Design Variant 4: Redundant Pressure Regulators.

Figure 5.13: Design Variants 3 and 4

repository [122]. Note that this drone model, while similar in structure to the model in Section 4.3, is both parameterized over more variables and has revised behaviors and a different cost model. Thus, conclusions from one model (e.g., optimal architecture) may not apply from one model to another.

5.5.1 Model Description

The structure of the multirotor model is shown in Figure 5.15. This case study focuses on the design of the rotor line architecture (AffectDOF), battery pack architecture (StoreEE), flight planning (Planpath), and recovery (ManageHealth) functions, however, modes from all functions (see Table 5.6 are included in the optimization. The objective is to design a resilient architecture, operational profile, and resilience policy which will minimize the cost of failures while maximizing operational revenue and minimizing design cost. As shown in Figure 5.15, the model consists of the following functions:

- `StoreEE`, the battery system;
- `DistEE`, the circuitry supplying power from the battery to the lines and on-board electronics;
- `AffectDOF`, the architecture of EMAs, motors, and propellers used to propel the drone;

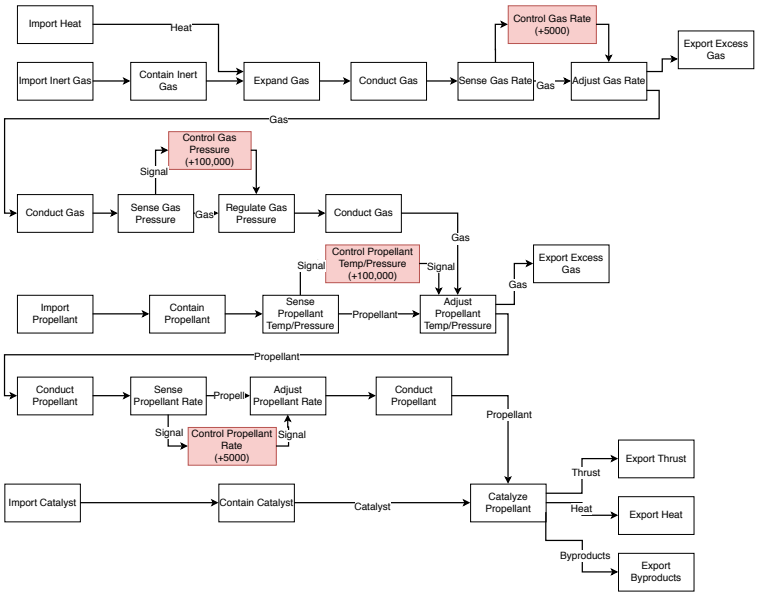
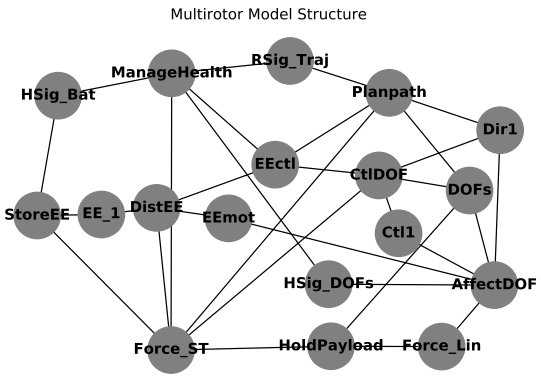


Figure 5.14: Design Variant 5: Optimized Control Features.

(a) Multirotor Drone [263]



(b) Model Structure

Figure 5.15: Representation of multirotor drone (left) in an model simulation (right). The Battery (StoreEE) powers the system while the rotor lines (AffectDOF) use the system control commands (Ctl1 and Dir1) to change the position of the aircraft (DOFs).

- `PlanPath`, the on-board navigation and path planning software and electronics;
- `CtlDOF`, the control system used to stabilize the drone;
- `HoldPayload`, the structure and landing gear of the drone; and
- `ManageHealth`, the health management system used to detect and reconfigure the flight in case of faults.

as well as the flows:

- `EE_1`, the flow of electricity from the battery;
- `EEmot`, the (high-current) flow of electricity to the motors/EMAs;
- `EEctl`, the (low current) flow of electricity to the controllers and electronics;
- `DOFs`, the position and velocity of the drone;
- `Dir1`, the directional commands given from path planning to the controller;
- `Ctl1`, the throttle commands given from the controller to the motors/EMAs;
- `Force_Lin`, the force from the landing gear to the motors;
- `Force_ST`, the force from the landing gear to all the other subsystems;
- `HSig_Bat`, the health signal for the battery;
- `HSig_DOFs`, the health signal from the motors/EMAs; and
- `RSig_Traj`, the reconfiguration signal used to change paths in case of faults.

The design of the battery and line architectures is motivated by minimizing design cost, minimizing weight (to increase operational performance), and enabling reconfiguration in case of the individual battery and rotor failures in Table 5.6 (to increase resilience). To achieve this in the battery, four structures are considered: monolithic, parallel, series, or series-parallel. When a battery fault occurs, a parallel architecture enables a reconfiguration which keeps the same voltage at

Table 5.4: Lookup Tables for Design Architecture Costs and Properties

	monolithic	series-split	parallel-split	split-both
cost ($)	0	50000	50000	100000
weight (kg)	0.4	0.5	0.6	
	quad-copter	hexa-copter	octo-copter	
cost	0	100000	200000	
weight (kg)	1.2	1.6	2.0	
drag factor	0.95	0.85	0.75	

a lower maximum current draw while a series architecture enables a reconfiguration which can keep the same current draw at a lower voltage. Series-parallel enables both types of reconfiguration at the expense of increased weight and cost. Three line architectures are also considered–a quad-copter configuration with four rotors, a hexa-copter configuration with six rotors, and an octo-copter configuration with eight rotors. While these line architectures increase resilience by enabling the system to remain stable in the case of individual line faults, they also increase weight and design cost (which can also harm resilience). The resulting design cost of the system is:

$$C_D = c_b(x_B) + c_l(x_L) \tag{5.8}$$

where the cost functions c_b and c_l, as well as the response variables of the design model y_D are given by lookup tables (see Table 5.4) for each value of design variables $\mathbf{x_D} = [x_B, x_L]$–the battery pack architecture x_B and line architecture x_L, respectively.

The purpose of this drone is to surveil a given area. To perform this task, it must fly over the area at a specific operational altitude and sense (e.g., photograph) the area below it at that altitude. As shown in Figure 5.16, different operational altitudes lead to different flight-plans which cause the drone to fly for a longer or shorter amount of time. When the drone flies lower, it must fly for longer, but it also senses more information, yielding higher operational revenue. This relationship is quadratic and related to the number of points viewed, according to the operational cost function:

$$C_O(x_h, x_D, y_D) = -n * \sum_{p \in A} 0.5 + \frac{2}{y(p)_h^2} \tag{5.9}$$

where C_O is the operational cost (which, being negative, is a revenue in this case), n is the number of flights, p is an individual viewed grid point in the set of viewed gridpoints shown in Figure 5.16, the operational variable x_O is the input flight altitude x_h, and the resulting altitude at each point in the simulation is $y(p)_h$. In addition to costs, the design of the operations is subject to constraints:

- It must return to the starting location with at least 20% charge, or $g_{O1} = (20 - l(t_e) < 0$, where l is the life of the battery in the nominal simulation and t_e is the end-time of the simulation.
- It must return with no faults present, or $g_{O2} = \sum_{f \in N} f \leq 0$, where f is a fault and N is the nominal simulation
- It must stay within a feasible range of operational altitudes over the flight simulation, or $g_{O3} = \sum_{t=0}^{t_e} max(0, (h(t) - 122)) \leq 0$, where t is the simulation time, t_e is the end-time for the simulation, $h(t)$ is the altitude, and 122 is the maximum allowable altitude in meters.

Finally, to mitigate failures in the drone, it may be helpful to have different recovery strategies for specific detectable faults. In the multirotor case, two sets of faults are considered: faults in the rotor lines and faults in the battery (see Table 5.6). When these faults are detected, the drone has a number of options for flight recovery: continuing the mission, returning to base, flying to the closest safe landing point, or landing immediately (options 0,1,2,3 in the model). The resilience costs of the system depend on the performance of resilience policy, flight-plan, and design architectures over the given set of fault scenarios:

$$C_R(x_D, y_D, x_O, y_O, x_{bp}, x_{lp}) = \sum_{s \in S} n * r_s * C_s(x_D, y_D, x_O, y_O, x_{bp}, x_{lp}) \tag{5.10}$$

where C_R is the overall resilience cost, s is a scenario in the set of fault scenarios S, n is the number of flights, r_s is the per-flight scenario rate, C_s is the cost of a scenario, and the resilience variables x_R are x_{bp}, the battery reconfiguration policy, and x_{bp}, the line reconfiguration policy. The set of scenarios S is shown in Table 5.6, which corresponds to the set of single-component fault scenarios defined in the drone model which are initiated in the middle of the flight. It should be noted that this is a subset of the full-set of fault scenarios, which would include both combinations of components

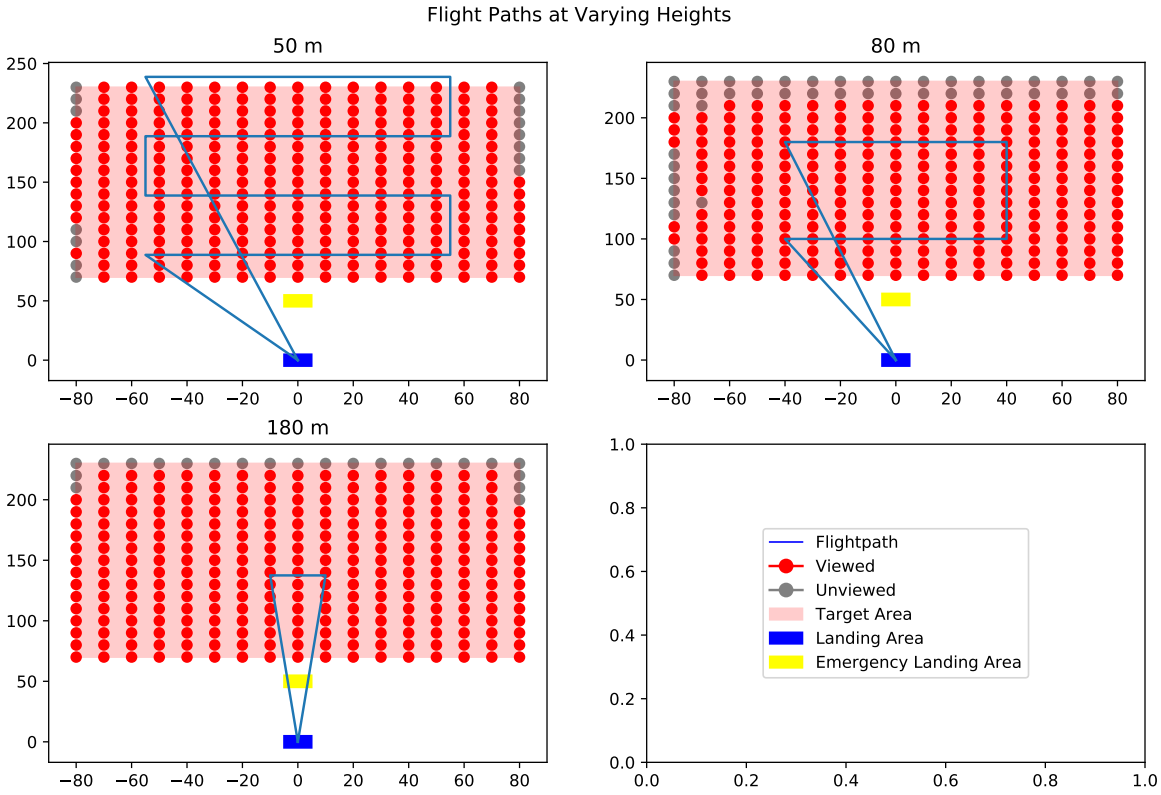


Figure 5.16: Flight Plans of Multirotor at Different Operational Altitudes. A higher operational altitude leads to a shorter flight but results in lower-quality imagery.

and a set of fault injection times. This subset is used here to lower computational time in the optimization for the purpose of demonstrating the architectures.

The cost of each scenario comes from the flight, landing, and repair costs of each modelled simulation, as shown below. Note that the input parameters ($x_D, y_D, x_O, y_O, x_{bp}, x_{lp}$ are all left out for of the equations for clarity to demonstrate the form of the scenario model. These scenario costs are:

$$C_s = C_f + C_l + C_r \tag{5.11}$$

where C_r is the repair cost:

$$C_r = min(1500, \sum_{m \in M} c_m) \tag{5.12}$$

which is the sum of the cost c_m of the modes M present in the final scenario, with a cutoff of \$1500 (the cost when the system is replaced rather than repaired). The flight costs C_f come from the safety cost of flying with a fault and the viewed value of the mission:

$$C_f = c_{fs} * t_f * -10 * \sum_{p \in A} 0.5 + \frac{2}{y(p)_h^2} \tag{5.13}$$

where c_{fs} is the safety cost to the airspace of flying with a fault present (which is determined by the schedule in Table 5.5 based on the location), t_f is the amount of time flying with the fault, and $\sum_{p \in A} 0.5 + \frac{2}{y(p)_h^2}$ is the value of the viewed points in the scenario (which is the same as the per-mission operational cost).

Finally, the landing costs C_l come from the impact to safety from landing in the target area c_t as well as the cost of landing in an area with property restrictions c_p (which is anywhere but the landing areas).

$$C_l = c_t + c_p \tag{5.14}$$

where c_t and c_p are determined from the schedule based on the flight scenario in Table 5.5.

Table 5.5: Safety cost schedule ($USD) for model effects in different design scenarios.

	target cost	outside target cost	property cost	flight cost
rural	960	960	1000	5
urban	4800	4800	10000	100
congested	7368000	4800	100000	1000

Table 5.6: Drone faults to be mitigated by recovery variables in quadcopter architecture with no recovery in the urban scenario flying at 30 m.

fault scenario	rate	unsafe flight time	landing location	event cost (USD)
StoreEE no charge	2.8e-05	1	over target	14914.6
StoreEE low charge	9.8e-05	6	outside target	15461.4
StoreEE cell shortage	1.5e-05	1	over target	14914.6
StoreEE cell degradation	1.5e-05	1	over target	14914.6
StoreEE cell mechanical fault	1.5e-05	1	over target	14914.6
StoreEE cell lost charge	8.4e-05	1	over target	14914.6
AffectDOF rotor short	3.1e-06	1	over target	15514.6
AffectDOF rotor open circuit	3.1e-06	1	over target	15214.6
AffectDOF rotor lost control signal	6.2e-06	1	over target	15114.6
AffectDOF rotor mechanical fault	3.1e-06	1	over target	15514.6
DistEE short	2.33333e-06	1	over target	15114.6
DistEE degr	3.88889e-06	1	over target	15014.6
DistEE break	1.55556e-06	1	over target	15014.6
CtlDOF noctl	2.8e-06	1	over target	15814.6
CtlDOF degctl	1.12e-05	1	over target	16014.6
Planpath noloc	2.8e-06	18	over target	17714.6
Planpath degloc	1.12e-05	18	over target	17714.6
HoldPayload break	1.55556e-07	1	over target	16314.6
HoldPayload deform	6.22222e-07	1	over target	16314.6
ManageHealth falsemasking	1.16667e-07	8	nominal	1654.86
ManageHealth lostfunction	2.33333e-08	8	nominal	1654.86

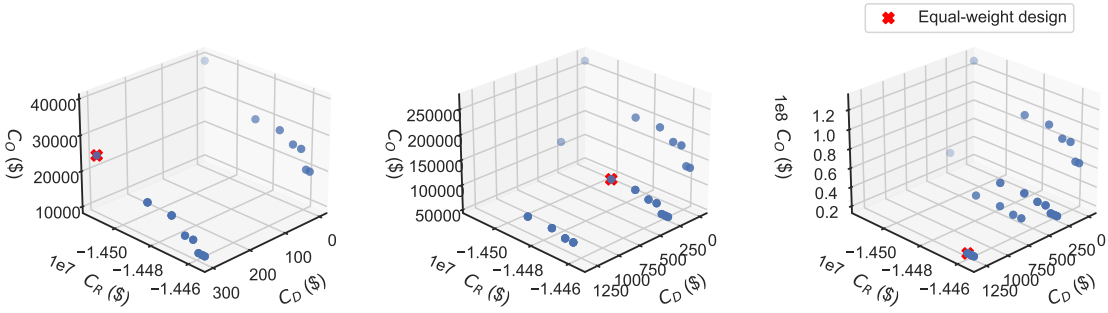
Table 5.7: Multirotor design problem variable names and values.

Variable	Values	Description
x_B	[0,1,2,3]	Battery Architecture (monolithic, parallel, series, series-parallel)
x_L	[0, 1,2]	Rotor Architecture (quadrotor, hexarotor, octorotor)
x_h	[20, 30 ... 120]	Operational Altitude
x_{bp}	[0,1,2,3]	Battery Fault Policy (continue, return, land safely, land immediately)
x_{lp}	[0,1,2,3]	Line Fault Policy (continue, return, land safely, land immediately)

Thus, to summarize, the optimization problem may be stated as:

$$
\begin{aligned}
min_{\mathbf{x}}\ f(\mathbf{x}) =& C_D(x_B, x_L) + C_O(x_B, x_L, y_B, y_L, x_h) + C_R(x_B, x_L, y_B, y_L, x_h, y_O, x_{bp}, x_{lp}) \\
s.t.\ C_D =& c_b(x_B) + c_l(x_L) \\
C_O =& -n * \sum_{p \in A} 0.5 + \frac{2}{y(p)_h^2} \\
C_R =& \sum_{s \in S} n * r_s * C_s(x_B, x_L, y_B, y_L, x_h, y_O, x_{bp}, x_{lp}) \\
g_{O1} =& 20 - l_N(t_e) < 0 \\
g_{O2} =& \sum_{f \in N} f \leq 0 \\
g_{O3} =& \sum_{t=0}^{t_e} max(0, (h(t) - 122)) \leq 0 \\
where\ x_B \in& [0,1,2,3], x_L \in [0,1,2], x_h \in [20, 30...120] \\
& x_{bp} \in [0,1,2,3], x_{lp} \in [0,1,2,3]
\end{aligned}
$$

where $C_D(x_B, x_L)$ is the design cost (evaluated in a lookup table), $C_O(x_B, x_L, y_B, y_L, x_h)$ is the operational revenue (evaluated in a nominal run of the simulation), $C_R(x_B, x_L, y_B, y_L, x_h, y_O, x_{bp}, x_{lp})$ is the resilience cost which is taken over the set of fault simulations, and the inequality constraints are all evaluated in a nominal run of the drone simulation with the operational model. Finally, the decision variables $\mathbf{x} = [x_B, x_L, y_B, y_L, x_h, y_O, x_{bp}, x_{lp}])$ are summarized in Table 5.7.



(a) Rural Scenario (b) Urban Scenario (c) Congested Scenario

Figure 5.17: Plots of the Pareto front of Design, Operational, and Resilience Costs in different flight scenarios.

5.5.2 Results

To demonstrate and compare the optimization architectures described in Section 6.2, this section applies each of them to the drone optimization problem in Section 5.5.1. This is performed in two parts: First, the design space is explored to show how the drone optimization represents challenges in integrated resilience optimization, having trade-offs between design/operational and resilience costs and couplings between design/operational and resilience variables which make the problem difficult to solve. Second, the different architectures are compared in terms of their optimization effectiveness and efficiency. Based on this comparison at the chosen problem parameters, some of the issues which influence the choice of optimization architectures in a particular design scenario are discussed.

5.5.2.1 Trade-space Analysis

As discussed in the problem description in Section 5.5.1, the drone design model can be simulated in different design scenarios which each have different failure costs because of the mission of the drone. This trade-off changes the Pareto front of the design space shown in Figure 5.17. In all three scenarios, the design cost is minimal and does not effectively influence the solution, which is

a result of the design cost assumptions (that R&D costs will be spread out over a large batch and the primary costs thus come from the manufacturing). The different rows in the pareto plots are from different optimal designs (combination of battery pack and rotor architecture) which result in different optimal costs. As shown, only four costs are on the Pareto front, even in the congested scenario, so some designs do not increase operational or resilience costs compared to the minimum design-cost no-redundancy design–in this case the joint-split battery pack architecture and octorotor line architecture.

However, the main effects of design features are different resilience costs (from increased redundancy in failure scenarios) and different feasible operational altitudes. There is a very clear relationship, for each design, however, between between the operational and resilience costs which results from the operational altitude of the flightplan. This is because operational height increases operational cost (because of less information detail), while decreasing resilience cost (because of the remaining battery, ability to leave the target area in time, etc.). Because of the squared term in the operational cost (Equation 5.9), the resulting relationship between operational and resilience costs also follows inverse quadratic function, making the gradient of operational costs steeper than that of resilience costs when the altitude is low.

These design trade-spaces lead to different optimal equal-weight designs, as shown in Figure 5.17. In the rural scenario, the unplanned landing (i.e., crashing) costs are low, meaning the majority of the resilience costs are repair costs, which are not enough to influence the solution significantly–resulting in a solution ($\mathbf{x}^* = [2, 0, 30, 1, 2]$) which maximizes operational revenue. While this solution has some redundancy in the battery, which reduces resilience costs without increasing operational costs, it also flies as low as possible, which increases operational revenue at the expense of failure costs. When there is a battery fault, it returns to base immediately to prevent a loss of power in flight. When there is a rotor fault, on the other hand, it flies to the closest landing point to mitigate the resulting crash. In the urban scenario, the crashing costs are much higher, resulting in an optimal solution ($\mathbf{x}^* = [2, 0, 40, 1, 1]$) that is a compromise between resilience and operational costs. In this solution, a slightly higher flight altitude is taken to reduce resilience costs at the expense of operational cost.

The flight height in this design additionally makes returning to the base a viable strategy in the case of a rotor fault. Finally, in the congested scenario (where the drone is flying over a crowd), the resilience costs overwhelm the other objectives, resulting a solution ($\mathbf{x}^* = [2, 1, 80, 1, 1]$) which effectively minimizes the costs of resilience over the design and operational costs. In this solution, the drone incorporates a hexarotor architecture in addition to the battery redundancy, which results in much more weight, meaning the drone can only fly at a much higher operational height. Returning to base is still the preferred policy for both sets of faults for this design.

This exploration illustrates the system couplings present in the design problem between the upper design and operational models and the resilience model. That is, different designs and operational policies result in different optimal resilience policies and vice-versa–in this case, when the drone flies low, it cannot return to base when a rotor fails, leading to a different optimal policy in the rural scenario than the urban and congested scenario. Additionally, much of the operational cost at the design and operational level (in terms of height and architecture) has major impacts on the resulting resilience costs. Additionally, depending on the form of the design trade-space, different optimization architectures could be preferable: in a situation where design and operational costs dominate resilience costs, it may not be beneficial to use an all-in-one or bilevel framework as opposed to a two-stage approach, because the resilience model does not affect the optimal decision in that instance. Thus, the rest of the analyses continue use the urban scenario as an example, since it is the scenario where the operational and resilience costs are on similar scales.

5.5.2.2 Architecture Comparison

To understand the relative performance of optimization architectures, this section assesses each in terms of effectiveness and computational costs in the drone problem in the urban flight scenario. To perform this comparison, each optimization architecture was applied to the drone design problem using an exhaustive search at each level, recording the optimal solution, solution costs, number of evaluations in each model levels, and total computational time of the search using an 3.20 GHz

Table 5.8: Comparison of optimization effectiveness and computational cost of optimization architectures. Note that negative cost is net profit/revenue (i.e., larger negative numbers are better and larger positive numbers are worse).

Approach	$\mathbf{x}^*$	Total Cost($)	D/O Cost($)	Res. Cost($)	D/O Evals	Res. Evals	Time (s)
All-in-one	2, 0, 40, 1, 1	-14,401K	-14,485K	84K	2304	2304	2375.3
Sequential	0, 0, 30, 2, 2	-14,249K	-14,514K	264K	144	16	27.5
Bilevel	2, 0, 40, 1, 1	-14,401K	-14,485K	84K	144	1152	1062.1
Bilevel Dec.	2, 0, 40, 1, 1	-14,401K	-14,485K	84K	144	576	245.6

Intel Core i5-6500 CPU. This comparison is shown in Table 5.8, where $\mathbf{x}^*$ is the found optimal solution, Total Cost is the combined design and resilience cost, D/O Cost ($) is the design and operational cost, Res. Cost is the resilience cost, D/O Evals is the number of design and operational model evaluations, Res. Evals is the number of resilience model evaluations, and Time (s) is the computational time. As shown, each architecture finds the optimal design except for the sequential architecture, which also has the least computational cost. Unlike the sequential approach, which sacrifices design optimality for computational performance, the bilevel approaches find the same design as the all-in-one approach with less computational cost. While some of this is due to the fewer design/operational model evaluations (which are much less in the bilevel approaches), the largest computational cost reduction comes from the fewer resilience model evaluations, since each resilience model evaluation is 21 simulations (one simulation per fault scenario) compared to a single simulation at the design/operational level.

As shown in Table 5.8, the bilevel frameworks result in a very large reduction of evaluations at the upper level compared to the all-in-one. This is because in an exhaustive search, the all-in-one architecture evaluates every model at each design $\mathbf{x}$, even when many of the designs (designs with the same design/operational polices) have the same design/operational cost. The main difference between the bilevel approach and the all-in-one approach is that the bilevel approach does not unnecessarily re-evaluate the upper level model to compare designs with the same design/operation policies but different resilience policies. While this property is not relevant to this problem, since the major driver of computational cost is the resilience model, it may become more salient in a multidisciplinary design scenario in which there is a more expensive design model.

The major computational reductions of the bilevel and sequential methods used in this problem

come from the reduction in lower-level evaluations. In the bilevel approach, this occurs because designs/operational plans which are infeasible at the upper level are not searched at the lower-level, resulting in fewer wasted iterations at the lower level (while not shown here, similar reductions can occur in an all-in-one approach if model evaluation is halted when the design/operational models are infeasible). When using the bilevel approach with the lower-level decomposition, there are even fewer lower-level evaluations because of the reduction in problem space from $(4 * 4 = 16)$ discrete combinations to $4 + 4 = 8$ solutions to each sub-problem. Additionally, the bilevel approach with the lower-level decomposition achieves lower computational cost because only the modes related to a particular variable (7 for the battery and 6 for the lines, and 12 for those not related to either, which is run once for each lower-level optimization) are evaluated for the search of that variable, resulting in fewer simulations per-lower level iteration. Finally, the sequential approach reduces the number of upper-level and lower-level iterations by only performing each optimization only once.

5.5.3 Discussion

While the results in Table 5.8 present a fairly clear story about the effectiveness and computational costs of the different architectures in the optimization of resilience, the ideal choice of optimization architecture flows out of the needs of the problem. For example, the all-in-one architecture is inefficient but easy to set up, and, when using an exhaustive search, is guaranteed to find the global minimum. Thus, in smaller problems, where there are not many variables or failure scenarios, there may be no reason to use a more complicated bilevel or sequential framework, especially it would not improve solution quality.

The case for using a sequential framework in the design of resilience is much more contextual (see: [24]). While it is not particularly effective at maximizing the inherent resilience of design and operational variables, it does enable one to find a design with optimal design/operational costs and a contingency management scheme which maximizes resilience for that design. Thus, there are a few scenarios where the sequential framework may be useful:

- When the design and operational variables do not affect the inherent resilience of the design (i.e., there is no constraint connecting the upper and lower levels of the optimization); in this case one does not need a feedback loop between the upper and lower levels because the upper-level variables do not effect the lower-level costs.
- When the costs in the upper-level model dominate the lower-level resilience costs (i.e., no matter the lower-level resilience cost of a particular design, the upper-level design and operational costs have a greater magnitude). This could happen if the upper-level design and operational costs are naturally at a higher magnitude than the lower-level resilience costs, or it could happen by design by including creating a surrogate of the lower-level resilience costs and including it in the upper-level objective.
- Finally, when the computational cost of the models is high and/or the space of designs is very large, it may be impracticable to conduct a full optimization of the design. Additionally, it is sometimes helpful in the optimization process (especially with gradient-based approaches) to find a feasible "starting point" design. In cases like this, when the goal is to find an acceptable design with as little computation as possible, the two-stage approach may be a better fit to the job than a bilevel or all-in-one approach.

Finally, as discussed in Section 5.5.2.2, the main advantage of the bilevel approach (both with and without the decomposition) when using an exhaustive search is that it reduces the number of evaluations which need to be performed at either level. However, this reduction (and its salience) is dependent on the attributes of the problem–if the upper-level models are computationally cheap, most of the reduction comes from not running the lower-level optimization over infeasible points. This reduction can largely be accomplished in an all-in-one framework by not evaluating the lower-level model when the design is infeasible, though the all-in-one framework will still explore the infeasible lower-level variables. However, depending on the form and solution strategy for the upper-level problem, this effect may be large or small. If the upper-level has no constraints, it will not reduce the number of lower-level iterations, since every single upper-level design is feasible and will be optimized in the lower level. Additionally, if one is using an optimization method other than

exhaustive search, the number of iterations at each level (and in an all-in-one approach) other factors will influence the convergence of the method, rather than just size of the design space, which will change this comparison.

The use of the lower-level decomposition strategy is also contingent on the form of the problem. While there is a clear case that the lower-level decomposition strategy reduces both the space of the lower-level problem (and thus the number of iterations) and the computational cost per iteration (since only a subset of the scenarios are run for each variable), it can only be used in specific situations where the variables are independent and can be associated with specific sets of scenarios. In the drone problem, this is resolved because each variable is a contingency plan for a specific set of fault scenarios where one of the line or battery modes is present. However, if there was an interaction or constraint relating these variables (e.g., an overall crash probability requirement or if there were interactions between fault scenarios/contingency plans), one could not use this strategy and be guaranteed an optimal solution. In this situation, one would either need to optimize the lower level as one problem or use a more sophisticated multidisciplinary design optimization technique. It should additionally be noted that this decomposition strategy becomes equivalent to the two-stage approach when each each independent set of variables corresponds to a single set of fault scenarios (rather than multiple).

5.6 Conclusions

This chapter covered the optimization of resilience in an integrated design framework, with a focus on how to most effectively and efficiently structure the optimization of design, operational, and resilience costs in an engineered system. There are many different variables which one may consider to achieve more or less resilience. As demonstrated in Example 5.4, where the optimized control of the monopropellant system was able to out-perform the other design variants, the resilience of a system is dependent not just on the features a system uses, but also the control policy, which may achieve comparable value to a dedicated recovery feature when optimized. Thus, optimization can

play an important role in the early design of complex systems, by enabling the valuation of design features, operational changes, and contingency management policies in a single decision.

However, when structuring an optimization problem like this–where the design, operations and contingency management is optimized together–it is prudent to choose an architecture which effectively finds the optimum design at a satisfactory computational cost. To explore this problem, Section 5.2 introduced the integrated resilience optimization formulation of the resilience-based design problem and presented optimization architectures to leverage this framework, and Section 5.3 showed a special decomposition strategy which can be used when design features are preventative and can be associated with subsets of scenarios. These architectures were then compared on a drone design problem in Section 5.5 where one must choose a design architecture and flight plan which minimizes design and operational costs while maximizing the resilience of the contingency management of the drone over a set of fault scenarios. On this problem, it was shown that a bilevel approach finds the optimal design at comparatively low computational cost compared to an all-in-one approach, especially when using a decomposition approach in the lower level problem that decreases both simulation cost and problem complexity. Conversely, a sequential approach has even less computational cost but does not find the optimal design because it does not factor resilience costs into the design/operational problem. However, the ultimate choice of architecture depends on the nature of the problem considered, including the computational cost at each level, the interacting design constraints, and the size, scope, and form of the problem.

However, there were some limitations of the comparison in Section 5.5 which may affect how well the results generalize to new problems. First, the variables the problem optimized in this work were discrete and were searched using an exhaustive search strategy. This constitutes a limitation because the results may not generalize to a gradient-based search or combinatorial optimization, since the convergence of those methods depend on more than just the dimensionality of the problem. Second, in this work, the design and operational costs were considered as a single model because of the low computational cost of the design model (which was a lookup table), but in many problems the design model and the operational model may each have significant computational expense. Finally,

the study in this work focused on the reduction of failure costs function using recovery features to improve the recoverability of the system. However, this is only one part of the resilience cost function. Optimizing the other part, the mode rate model (e.g., as in Section 5.3) , would enable one to consider the effect of preventative strategies such as maintenance, product quality or design margin, and prognostics and health management [113]. Thus, to comprehensively consider resilience in the integrated resilience optimization framework presented here, future work should additionally investigate the optimization of preventative strategies in a single framework with recovery strategies.

Chapter 6: Validating the Design of Resilience using Uncertainty Quantification

6.1 Motivation

Taken together, Chapters 3-5 present an overall value-based framework for simulating faults and quantifying the value of resilience to optimize a system in the early design phase. However,the validity of applying this framework (or ones like it) to a given design problem cannot necessarily be presumed, since design and analysis approaches are subject to error. The main difficulty with early design processes is that models of design risk in the early design process are relatively high-level [8] and risk quantification in this context is subject to both parametric (i.e., whether estimates used for early decision-making will represent the real data) [11, 299] and structural (i.e., whether a fault model's constructions can represent all failure scenarios) [8] uncertainties [151]. This uncertainty is epistemic (as opposed to aleatory) because it is a result of the designer's lack of knowledge [144], not an underlying variable process. As a result, while much literature has asserted the value of early risk-based design processes, and there is support that early decisions have impact on project risk [273], it may not be clear that using these methods will consistently and accurately identify the optimal set of risk-reducing features to add to a system. This can be thought of as two interacting problems: the ability of using the method to effectively explore the space of designs and the uncertainty present in the model form and parameters used to justify the final decision.

Both the quantification of uncertainty in design methods and validation of design methods have been identified as significant needs within design research [66]. From a design perspective, a major challenge provided by the uncertainty in models used to inform decision-making is managing the trade-off between modelling effort and decision improvement [41]. In early design processes, it is often necessary to make assumptions about how the system will operate to make a decision.

However, the optimal decision can be highly sensitive to assumptions [73], which is important when the assumption is not easily knowable (e.g., is not necessarily based on long-established research or experience). In fact, many cost models used to make high-level decisions have been shown to be sensitive to subjective parameters [214]. Additionally, when the designer has poor judgement, it would be better for them to follow established heuristics than to construct a utility model to justify decisions [25]. Consequently, one might justifiably doubt if one can meaningfully proceed through the design process using a traditional decision-making framework [236, 19]. However, these in design cases with low information, better decisions can result when the designer considers the uncertainty due to lack of evidence (the epistemic uncertainty) [15].

Previous work concerned with risk and resilience-based design under epistemic uncertainty has focused on identifying the aspects of the model or design most affected by uncertainty so it can be accounted for in design or reduced in the model. In the field of risk assessment, Khorsandi and Aven [144] created an approach to quantify the risk of incorrect assumptions in a risk model and Bjørnsen et al. [29] used an extended value of information analysis to account for poor information fidelity in decision-making. In the design area, Owens et al. [215] provide an approach to account for epistemic uncertainty in mission design through sensitivity analysis, using a probability of sufficiency measure to determine how much design margin should be incorporated to account for uncertainty. Similarly, Stone et al. [271] used conjoint analysis to determine when to refine models under uncertainty. While these approaches enable designers to isolate and address the issue of uncertainty, they do not determine whether a given design and modelling approach is valid given that uncertainty. To address this, Malak and Paredis [175] proposed using value of information to validate models for reuse, identifying it as a path for further research. This work builds on that idea.

6.1.1 Aims, Contributions, and Organization

The aim of this chapter is to provide and demonstrate processes that support validation of early risk and resilience-based design approaches. In this work, design process validation is considered to be

determining if a given design approach provides meaningful results for the problem it is supposed to solve. It should be noted that other definitions of validation exist across related fields. Specifically, in model development, the validation of a model refers to its ability to give accurate, credible results, while the suitability of the results leading to the right decisions for a particular problem is referred to as "accreditation" [63, 275]. We refer the process performed here as supporting validation because we are interested in the validity of the combined design process, which is based not only on the suitability of an underlying model, but on how that model is used for design.

To approach this challenge, this work focuses on answering the question: "Was a risk-based or resilient design approach so sensitive to uncertain assumptions that one should not accept the results?" While one might understand the importance of reducing risk early in the design process, it may be difficult to tell if an approach gives good results when it is based on a relatively low-fidelity model of the system based primarily on assumptions, rather than measurements from a real system. The main contribution shown here is a testing framework that determines:

1. whether quantifying uncertainty is necessary within an approach,
2. whether the value of the design is significantly improved by the approach when uncertainty is considered, and
3. whether a decision is too sensitive to epistemic uncertainty to justify a particular choice given by the approach.

Based on these tests (as well as other tests, process information, and their judgement), a decision-maker could then determine whether to accept a given risk-based design process as valid.

6.2 Method

Testing the validity of early risk and resilience-based design approaches in the framework presented here follows the process illustrated in Figure 6.1. First, a variety of design variants are explored and a value model is constructed with different parameters for each based on the results of the model,

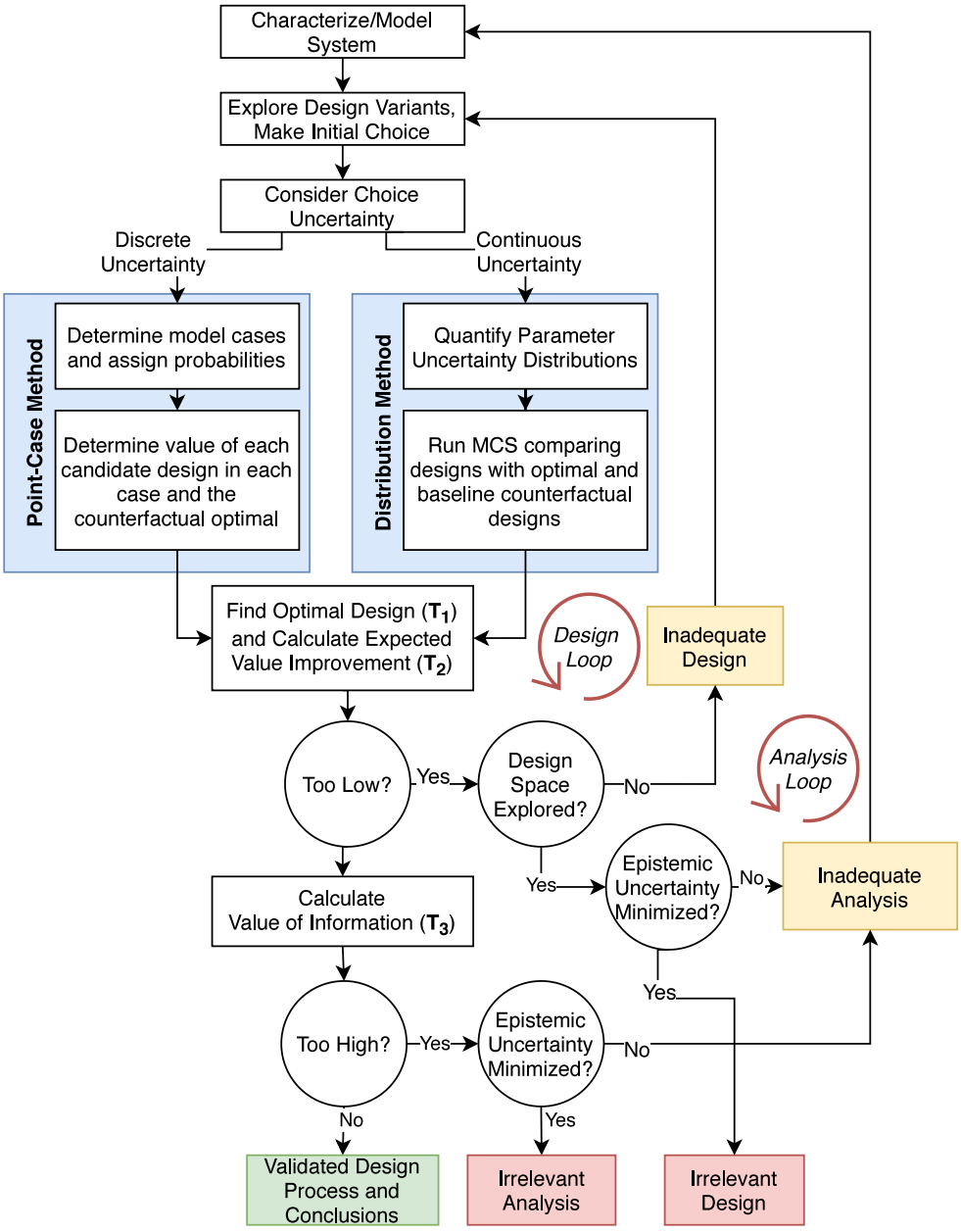


Figure 6.1: Proposed testing framework within a larger design process.

giving a chosen design, as discussed in Section 6.2.1. Next, the uncertainty in the value model with respect to the assumptions made (that is, the epistemic uncertainty resulting from the designer's lack of knowledge) is quantified in terms of continuous and discrete distributions, as applicable to the type of uncertainty (parametric, structural, etc.) and desired level of detail, as presented in Section 6.2.2. Based on this uncertainty model, tests are performed on the value model to determine whether the process was valid, as presented in Section 6.2.3. The first test, $\mathbf{T_1}$, compares the chosen design with the design that would have been chosen considering uncertainty to test the use of point estimates in design. The second test, $\mathbf{T_2}$, uses the expected value improvement in the design process to test whether the design process sufficiently explored the design. If it was sufficiently explored, one might conclude that either the there was either too much uncertainty in the analysis or that the design process was irrelevant to achieving the desired outcome. The third test, $\mathbf{T_3}$, uses the value of information to determine if the analysis used was too uncertain to be relevant enough to make the given decisions. As shown in Figure 6.1, based on the interpretation of these tests, the designer may then either revise the design process by exploring further design alternatives and gathering more information to inform analysis or declare the design or analysis process itself valid or invalid for the considered problem.

6.2.1 Design Problem Formulation

In general, an expected cost model for risk or resilience-based design balances the design, operational, and failure costs, following the form provided in Chapter 3:

$$C = C_D + C_O + \sum_{f \in F} n_f * C_f \tag{6.1}$$

where C_D is the design cost, C_O is the operational cost, and the failure cost is the sum of the expected values of each failure f in the set of failures F, which is the expected number of failures n_f times the failure cost C_f. This failure cost C_f can result from the direct effects of failures (e.g. safety and repair costs) in the case of a risk-based design process and from the dynamic operational

losses resulting from a failure in the case of a resilience-based design process. Demonstration of this value modelling process is provided for both examples in Section 6.3 and Section 6.4 (a simpler example is provided in Section 3.5).

6.2.2 Uncertainty Quantification

Uncertainty quantification is used in this framework to determine how the designer's epistemic uncertainty with respect to the cost function affects the decision-making process. This involves quantifying the single set of assumptions (e.g. chosen models, point estimate inputs, etc) used for design in terms of their statistical distributions. Unfortunately, because the primary type of uncertainty in early design is epistemic, the probability distributions must be specified from the Bayesian (or subjective) point of view–where different levels of probability represent different levels of the belief of the decision-maker– instead of the frequentist view–where probability is the considered the distribution of outcomes over a large number of experiments [46, 288]. Thus, the uncertainty must be elicited from the designer, which can be accomplished using a number of methods [75], examples of which are presented below.

6.2.2.1 Discrete Assumptions

Often, assumptions must be made about exclusive, discrete options, such as the form of a model, whether or not a condition is present in a system, or what sort of scenario the system will encounter. Additionally, when one is already using a relatively simple design method (e.g. using a few simple equations rather than detailed simulations because they take less time to evaluate), it may be more convenient to express the uncertainty as discrete assumptions with each case given a discrete probability, as shown in Figure 6.2. In this approach, the value of each design in each scenario must be determined in each case (e.g., with different inputs to the model or a different chosen form of the model), and probabilities must then be assigned for each scenario.

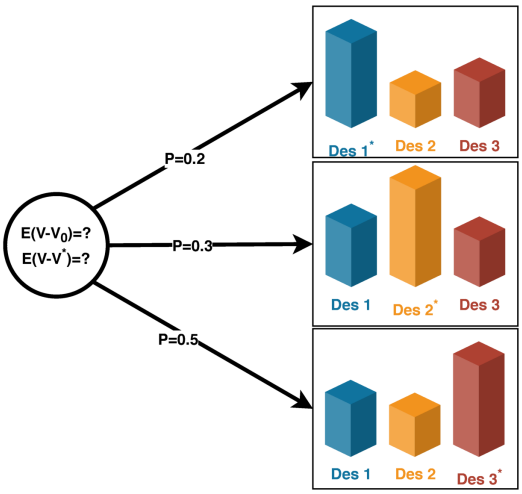


Figure 6.2: Considering a design problem under uncertain discrete assumptions may lead to differing preferred design options in each case.

6.2.2.2 Continuous Assumptions

When assumptions are made about the range of parameters, continuous distributions can be used to represent the uncertainty inherent to assuming a particular set of values for input parameters. In this process, one must quantify not just the uncertainty of a particular parameter estimate, but the possibility of multiple poor estimates interacting due to systematic errors. Since the parameter estimates used in early design are based on previous information and the intuition of the designer (which is subject to biases) these sources of error must be considered in the uncertainty model.

This work uses a multiplicative model of error sources to account for possible compounding interactions between estimation errors while maintaining a relatively simple uncertainty model. The model used here considers possible biases at each level of system representation used in the case study: system, component/function, and parameter type. This model is given in Equation 6.2, where system-wide estimation bias is S, component (or function-related) bias is C, parameter type bias is T, and individual instantiation error is ϵ, resulting in a distribution Y for the parameter that

was previously given the point-estimate y_{est}.

$$Y = y_{est} * S * C * T * \epsilon \tag{6.2}$$

In this work, the Beta-PERT distribution is used for each source of error, centered around 1 and varying between a maximum multiplier R and minimum multiplier $1/R$. The Beta-PERT distribution is used for this because it is a traditionally-used distribution for representing the uncertainty of the designer about precise parameter values [278]. In the model used here, the overall distribution for the variable Y has a most probable point at the estimate, with tails extending an order of magnitude R above and below the estimate in the direction of each considered source of error– a distribution which somewhat conservatively skews towards over-estimates–implicitly accounting for theorized optimism biases [254] in y_{est}, as proposed by Moskowitz and Bullers [200]. However, alternative uncertainty models may be used to represent the specific problem at hand if different estimation biases are considered to be affecting the estimate or if the error distribution for a bias is considered to be uniform or multimodal, for instance.

6.2.2.3 Uncertainty Propagation

To make a decision given uncertain parameter values, the uncertainty must be propagated through the value function to determine the expected value or utility of the designs. To perform the tests required by this method, two uncertainty-based quantities need to be determined: the expected value of a design and the expected value of information. The expected value of a design $\mathbb{E}\{V(\vec{x})\}$ given I discrete uncertain inputs $\vec{z}_i$ with probabilities P_i is:

$$\mathbb{E}\{V(\vec{x})\} = \sum_{i \in I} P_i * V(\vec{x}, \vec{z}_i) \tag{6.3}$$

The expected value $\mathbb{E}\{V(\vec{x})\}$ of a design given continuous uncertain variables is:

$$\mathbb{E}\{V(\vec{x})\} = \int_{-\infty}^{\infty} V(\vec{x}, z) * f(z) dz \tag{6.4}$$

where $V(\vec{x}, z)$ is the value of the design $\vec{x}$ with uncertain parameter z, and $f(z)$ is the probability density function for that parameter.

While a number of metrics for the value of information have been provided in previous work (see Section 2.1.2), the expected value of perfect information is used here because it is relatively simple to calculate (does not require specifying additional distributions), and because the uncertainty distribution quantified is epistemic, not aleatory. This metric is the difference in expected value between the design with the optimal expected value based on the quantified uncertainty, and the designs that would be optimal if a given input scenario were to come to fruition. This is called the value of perfect information because it represents how much value improvement one could get if they knew the exact values of uncertain inputs ("perfect information"), rather than a range of uncertainty. When uncertainty is discrete, the $EVPI$ is:

$$EVPI = \sum_{i \in I} P_i * (V(\vec{x_i^*}, \vec{z_i}) - V(\vec{x_c}, \vec{z_i})) \tag{6.5}$$

where I is the set of scenarios, P_i is the probability of a given scenario with inputs $\vec{z_i}$, $V(\vec{x_i^*}, \vec{z_i})$ is the value of the optimal design $\vec{x_i^*}$ for that scenario and $V(\vec{x_c}, \vec{z_i})$ is the value of the chosen design $\vec{x_c}$. For continuous parameters, this similarly takes the form:

$$EVPI = \int_{-\infty}^{\infty} (V(\vec{x^*}(z), z) - V(\vec{x_c}, z)) f(z) * dz \tag{6.6}$$

where $\vec{x^*}(z)$ is the optimal design vector at a specific value of the uncertain parameter z, $V(\vec{x_c}, z)$ is the value of the chosen design, and $f(z)$ is the probability density function for z.

In practice, calculating the expected value $\mathbb{E}\{V(\vec{x})\}$ of the design (as well as the $EVPI$) may well require a high-dimensional integration that may be difficult to derive analytically. In this work

(see Example 2), this value is calculated using Monte Carlo sampling on the inputs of the model because the underlying model and optimization process is computationally light-weight. In practice, a variety of techniques can assist with the calculation of this integral (e.g. Taylor series expansion, numerical integration, variance reduction, etc. [77]) depending on how the uncertainty is quantified. For example, if the underlying quantified uncertainty is normally distributed and the value model is linear, linear transformation rules can be used to calculate the expectation. On the other hand, if the value model calls a black box simulation, it may be necessary to approximate the result instead.

6.2.3 Process Acceptance Conditions and Recommendations

Based on the metrics defined in Section 6.2.2, this framework performs validity tests on the design process to determine whether it was appropriate for the given problem given the level of uncertainty. The three tests considered here are an uncertainty check which determines if considering uncertainty in the process was necessary, a value test which determines if the process improved the design, and a meaningfulness test which determines if the process sufficiently justifies a given choice given the uncertainty. Each are presented below.

6.2.3.1 T_1: Uncertainty Check

The uncertainty check determines if there was a significant difference between the design chosen in an uncertainty-based method and a method made using point-estimates with no uncertainty. If the design does not change significantly when uncertainty is quantified in a specific problem, its use may then be accepted because quantifying uncertainty is not necessary. On the other hand, if there is a significant improvement in the design, then consideration of uncertainty is required and the design process is not valid. For this condition to hold, the value of uncertainty quantification V_{UQ}, calculated as difference between the expected value of the uncertainty-based design $\mathbb{E}\{V(x^*_{UQ})\}$ and the expected value of the point-estimate-based design $\mathbb{E}\{V(x^*_P)\}$ (where the uncertainty has now

been quantified) must be below some desired threshold $V_{UQ,des}$, as shown:

$$V_{UQ} = \mathbb{E}\{V(x^*_{UQ})\} - \mathbb{E}\{V(x^*_P)\} < V_{UQ,des} \tag{6.7}$$

It should be noted that while it is generally expected that there will be a value difference between the uncertainty-based design and the point estimate-based design, for some discrete problems there may be no difference at all if the discrete options have a large enough difference in value compared to the uncertainty range.

6.2.3.2 T_2: Value Improvement Test

The value improvement test determines if the value of the design increased significantly from the initial design before starting the design approach and the chosen design after the design approach. If the design chosen by a given design-analysis procedure does not significantly improve the value of the design, the procedure in that instance was not valuable. Based on this condition being violated, one would conclude that either the design space has not been meaningfully explored, the analysis is too uncertain to yield a significant difference in design value, or that the procedure was irrelevant to the needs of the problem. This condition may be stated:

$$V_{DP} = \mathbb{E}\{V(x_0)\} - \mathbb{E}\{V(x^c)\} > V_{DP,des} \tag{6.8}$$

where V_{DP} is the value of the design approach, $\mathbb{E}\{V(x_0)\}$ is the expected value of the starting or input design x_0, $\mathbb{E}\{V(x^c)\}$ is the value of the design x_c chosen by the design approach, and $V_{DP,des}$ is the desired minimum value increase from the design process, which may be determined from the cost of going through the process and the value increase given by alternative established processes (that is, the opportunity cost).

6.2.3.3 T_3: Meaningfulness Test

The meaningfulness test determines if the design decisions given by the design approach are justifiable given the uncertainty that could change the optimal design. The meaningfulness test is based on the expected value of perfect information, which, as shown in Section 6.2.2 is the difference between the value of the optimal design over the range of uncertainty and the optimal design with no consideration of uncertainty in that scenario. If the expected value of information is too high, more certainty is needed to make the decision than is provided in the given analyses. Ideally, this means more data and analysis is needed before making the decision than is provided in the approach. However, there could be other problems, such as the design space having adverse inherent variability or other factors that make the problem as stated impossible to design towards. This condition is then:

$$EVPI < EVPI_{des} \tag{6.9}$$

where $EVPI$ is the expected value of perfect information of the chosen design and the $EVPI_{des}$ is the maximum allowable value of information, which is determined based on the the problem considered.

A compelling threshold for the value of perfect information in the context of justifying individual choices (rather than the entire process) is the difference in expected value between the optimal design x^* and the alternative non-optimal design $x^{\not *}$ under consideration. If the value of information is greater than this difference, the choice is not justified based on the decision model–there is too much choice-affecting uncertainty to consider this choice to be meaningful.

$$EVPI < \mathbb{E}\{V(x^*)\} - \mathbb{E}\{V(x^{\not *}\} \tag{6.10}$$

It should be noted that this definition is only appropriate for integer problems or comparisons between alternatives where there are significant value gaps between designs. When the problem is continuous, applying this test generates a zone of invalidity x_{inv} in which changes are not justified

from the model given the level of uncertainty:

$$x_{inv} := x \ \forall \mathbb{E}\{V(x^*)\} - \mathbb{E}\{V(x)\} > EVPI \tag{6.11}$$

It should be noted that while a violation of this criteria means that not enough is known to make a decision, it does not mean that no decision should be made, for example, if the uncertainty is difficult or expensive to reduce. Instead it means that the decision-making process does not sufficiently justify the choice one way or another–the decision is arbitrary given known information. While one would still expect to see the best payoff from the design with the highest expected value, the value difference between designs is too small compared to the choice-affecting uncertainty to have confidence in the results.

6.3 Example: Early Choice of Health Management Approach

The following example demonstrates how the discrete point-case uncertainty method presented in Section 6.2.2.1 can be used to perform tests on a simple value-driven design process. In this process, one considers a prognostics and health management (PHM) approach for a given system with a simple value model with different inputs and resulting values of each design resulting from the designer's expected performance of each approach. The problem here is relatively simple and generic, considering a single fault in a moderate-scale system over many years. The rest of this section provides the details of the value model, considered design options, and possible design scenarios.

6.3.1 Value Model

The value model model considers the design, operational, and failure costs of the system, following the form provided in Chapter 3. In the single-fault case, the fault costs C_F are the sum of expected failure scenario costs (6.12), mitigated fault scenario cost (6.13), and false alarm cost for the system

(6.14):

$$C_F = T * r_f * P_{fs|f} * C_{fs} \tag{6.12}$$

$$+T * r_f * P_{ms|f} * C_{ms} \tag{6.13}$$

$$+T * r_u * C_u \tag{6.14}$$

where T is the lifecycle usage time, r_f is the fault rate, $P_{fs|f}$ is the probability of the fault scenario fs occurring unmitigated (or, $1 - E$, where E is the effectiveness of the mitigating feature) with cost C_{fs}, $P_{ms|f}$ is the probability of the mitigated scenario ms given the fault occurs (E) which has cost C_{ms}, r_u is the false alarm rate, and C_u is the cost of a false alarm. Considering a model at this level of detail, then, controllable PHM benchmarks include effectiveness E, mitigated fault severities C_{ms}, and false alarm rate r_u, although other parameters may be controllable through the design of the rest of the system (lifecycle time, false alarm costs, failure costs, etc) and may occur at lower levels of a more detailed resilience-based cost model (e.g. critical prediction horizon, etc).

The operational costs C_O is the result of costs and revenues from use (6.15) as well as the cost of maintenance (6.16):

$$C_O = T * (c_o - r_o) \tag{6.15}$$

$$+(T/t_m) * C_m \tag{6.16}$$

where c_o is the per-hour cost of operations, r_o is the per-hour cost of revenue, t_m is the maintenance interval, and C_m is the cost per maintenance interval. For more detailed benchmark specification of PHM systems, the cost of performance can result in benchmarks on hardware weight, volume, and other performance-related parameters while the maintenance can be specified here using the average intervals for each required operation and the costs of each of those operations.

The design costs C_D are a function of the per-unit baseline manufacturing cost C_{mb}, feature manufacturing costs C_{mf} and the overall baseline development costs C_{db} and feature development

costs C_{df} that result from the expected time, cost, and risk of developing the technology:

$$C_D = (C_{db} + C_{df})/n + (C_{mb} + C_{mf}) \tag{6.17}$$

where n is the total number of systems manufactured. The implementability of the system are related as much to the project as to the technology, however a more detailed consideration could result in benchmark plan for hardware and software costs, as well as the product schedule and development resource use.

The overall value considered in this design uses the net present value (NPV) formula to balance the future value of design and operational costs against the immediate value of design costs:

$$V = -NPV(C_F, C_O, C_D, i, T) \tag{6.18}$$

where i is the yearly discount factor.

The baseline values and demonstration of the model is shown in Table 6.1. In this situation the system is expected to run consistently at a moderate scale ($n = 200$ systems) over a long life ($T = 25000$ hours over a 17 year life) at high reliability ($r_f = 5 * 10^{-6}$ faults/hr) by an established company ($i = 5\%$). However, if the system fails there will be a large cost due to safety effects ($16M dollars). This is typical of the design situation with safety impacts, such as low-scale aviation.

6.3.2 Design Process and Results

A variety of different potential functions are considered in the assessment in Table 6.1 to the system to increase system resilience, including:

1. **Baseline Design:** The baseline design in which no risk-mitigation strategy is provided.
2. **Increased Inspection:** The baseline design in which risk is reduced through frequent inspection. This results in considerable operational costs due to the heavy maintenance schedule but is given a low effectiveness ($E = 0.5$) since not all faults will be apparent to inspectors in the

time immediately before the event occurs.

3. **PHM System (CBM):** A design with a PHM system implementing a condition-based maintenance strategy. This approach is given moderate effectiveness ($E = 0.95$) because while the underlying model may characterize and track degradation of the system well, there are still "random" errors that will occur unpredictably. This also has a minor effect on operational costs due to weight and sensor maintenance.

4. **Hot Redundancy:** A design with a hot redundancy that is constantly running and managed to activate immediately when a fault occurs in the active component. This has high effectiveness (assuming total independence of faults) but results in higher operational costs since parts must be replaced twice as often and there is a cost of weight. However, there is less up-front design cost since redundancy circuits are well-developed technology.

5. **PHM System (Recovery):** A design with a diagnostic-based recovery system leveraging flexibility or reconfigurability in the system. The implementation of such a system is contingent on the rest of the system having functionality that can be reconfigured. In this demonstration it is considered that a high effectiveness can be achieved using this approach with less maintenance and performance cost than a redundancy. However, the chance of partial recovery makes the mitigated fault cost higher than it would be otherwise, and there is a high up-front design cost to develop the system.

A comparison of the various systems using some example numbers is shown in Table 6.1. As shown, in this design situation, the total NPV is greatly improved by the implementation of any fault-mitigating feature due to the high failure cost, and the health management functions (CBM System, Hot Redundancy, and Recovery System) each have a comparable overall NPV, with the recovery system having the highest.

Table 6.1: Comparison of Design Options Using Cost Model for Resilient Features in Baseline Scenario

Fault Costs	**Baseline Design**	**Incr. Inspection**	**PHM (CBM)**	**Hot Redundancy**	**PHM (Recovery)**
New Fault Rate	5.00E-06	5.00E-06	5.00E-06	*1.00E-05*	5.00E-06
Effectiveness		0.5	0.95	*0.999995*	0.9999
False Alarm Rate		1.00E-05	1.00E-05	*2.00E-05*	1.00E-05
Unmitigated Failure Cost	$16,000,000.00	$16,000,000.00	$16,000,000.00	$16,000,000.00	$16,000,000.00
Mitigated Fault Cost	$0.00	$300.00	$400.00	$900.00	$100,000.00
False Alarm Cost		$300.00	$400.00	$900.00	$100,000.00
Unmitigated Failure Rate	*5.00E-06*	*2.50E-06*	*2.50E-07*	*2.50E-11*	*5.00E-10*
Mitigated Fault Rate		*2.50E-06*	*4.75E-06*	*1.00E-05*	*5.00E-06*
Lifecycle Unmitigated	*1.25E-01*	*6.25E-02*	*6.25E-03*	*6.25E-07*	*1.25E-05*
Lifecycle Mitigated	*0*	*6.25E-02*	*1.19E-01*	*2.50E-01*	*1.25E-01*
Lifecycle False Alarms	*0*	*2.50E-01*	*2.50E-01*	*5.00E-01*	*2.50E-01*
Unmitigated Failure Costs	*$2,000,000.00*	*$1,000,000.00*	*$100,000.00*	*$10.00*	*$200.00*
Mitigated Fault Costs	*$0.00*	*$18.75*	*$47.50*	*$225.00*	*$12,498.75*
False Alarm Costs	*$0.00*	*$75.00*	*$100.00*	*$450.00*	*$25,000.00*
Total Fault Costs	*$2,000,000.00*	*$1,000,093.75*	*$100,147.50*	*$685.00*	*$37,698.75*
Total Fault Costs (NPV)	**$1,322,923.19**	**$661,523.61**	**$66,243.72**	**$453.10**	**$24,936.28**
Operational Costs					
Maintenance Int. (hrs)		100	1000	400	500
Per-interval cost		$100.00	$120.00	$200.00	$200.00
Usage Cost ($/hr)	$400.00	$400.00	$400.01	$402.00	$400.01
Usage Revenue ($/hr)	$500.00	$500.00	$500.00	$499.00	$500.00
Usage Profig ($/hr)	*$100.00*	*$100.00*	*$99.99*	*$97.00*	*$99.99*
Total Operations		*250.00*	*25.00*	*62.50*	*50.00*
Total Maintenance Cost	*$0.00*	*$25,000.00*	*$3,000.00*	*$12,500.00*	*$10,000.00*
Total Usage Profit	*$2,500,000.00*	*$2,500,000.00*	*$2,499,750.00*	*$2,425,000.00*	*$2,499,750.00*
Total Operational Profit	*$2,500,000.00*	*$2,475,000.00*	*$2,496,750.00*	*$2,412,500.00*	*$2,489,750.00*
NPV Operational Profit	**$1,653,653.99**	**$1,637,117.45**	**$1,651,504.24**	**$1,595,776.10**	**$1,646,874.00**
Des. & Manu. Costs					
Baseline Dev. Costs	$100,000,000.00	$100,000,000.00	$100,000,000.00	$100,000,000.00	$100,000,000.00
Feature Dev. Costs	$0.00	$10,000.00	$3,000,000.00	$500,000.00	$3,000,000.00
Baseline Manu. Costs	$80,000.00	$80,000.00	$80,000.00	$80,000.00	$80,000.00
Feature Manu. Costs	$0.00	$0.00	$800.00	$4,000.00	$800.00
Single-Sys. Dev. Costs	*$500,000.00*	*$500,050.00*	*$515,000.00*	*$502,500.00*	*$515,000.00*
Tot. Manu. Costs	*$80,000.00*	*$80,000.00*	*$80,800.00*	*$84,000.00*	*$80,800.00*
Tot. Des. & Manu.Costs	*$580,000.00*	*$580,050.00*	*$595,800.00*	*$586,500.00*	*$595,800.00*
Totals					
Tot. Value	-$80,000.00	$894,856.25	$1,800,802.50	$1,825,315.00	$1,856,251.25
Tot. NPV	**-$249,269.20**	**$395,543.84**	**$989,460.51**	**$1,008,823.00**	**$1,026,137.73**
KEY:	Input	*Response*	**NPV**		

6.3.3 Uncertainty Quantification

To quantify the uncertainty effecting this process, a number of discrete scenarios are considered and assigned probabilities based on the likelihood that they will occur. In this example the scenarios are considered to be complete and mutually exclusive, meaning probabilities of each scenario sum to one. These probabilities P for each scenario are provided below, along with the parameter values, explanation, and discussion. The effect of these new parameter values on the optimal design is shown in Figure 3 in the main paper for each scenario.

- **Situation 1: Nominal Scenario**

($P = 0.35$)

In the baseline case, high failure costs and moderate performance costs make the combination recovery system preferable due to the low effect on performance and moderate effectiveness compared to the redundancy scheme.

- **Situation 2: High False Alarm Rate**

  ($r_u = 5 * 10^{-4}$, $P = 0.2$)

  When the rate of false alarms given by a system is high, the introduced cost has a significant effect on the preferability of a recovery system, since the cost of accidental recovery is higher than the cost of accidental prevention.

- **Situation 3: High Effectiveness, Common Mode Errors**

  (CBM $E = 0.99$, Red. $E = 0.9$, $P = 0.2$)

  In the baseline case, the preferability of a redundancy is a result of higher effectiveness due to assumptions about independence and prognostic effectiveness. However, when the fault rate for each redundancy is not independent and a high effectiveness can be achieved, the prognostic system becomes preferable.

- **Situation 4: Low Mitigatability**

  (Red., Recovery $E = 0.8$, $P = 0.1$)

  When a fault is difficult to recover from, the effectiveness of fault masking and recovery options is low. As a result, a prognostic approach becomes preferred.

- **Situation 5: Low Cost of Recovery**

  (Recovery $C_u, C_{ms} = 1000$, $P = 0.1$)

  The assumption for the recovery system in the baseline case is that there will be some additional cost taken on by entering the recovery state (e.g. by triggering a safety system). When this cost is low, the recovery system becomes preferred.

- **Situation 6: High Fault Rate**

  ($r_f = 1 * 10^{-4}$, $P = 0.05$)

  In the high rate situation, expected failure costs become even more dominant, making less effective features even less preferable. This reverses in low-rate situations, where maintenance, performance, and manufacturing costs are more likely to be a consideration.

Table 6.2: Validity tests for the health management design problem considering a variety of different design scenarios.

	Baseline Design	**Increased Inspection**	**PHM System (CBM)**	**Hot Redundancy**	**PHM System (Recovery)**	**Scen. Prob.**	$C^* - C_{point}$	$C^* - C_{unc}$
1 Nominal Scenario	-$249K	$396K	$989K	$1009K	**$1026K**	0.35	$0K	-$17K
2 High False Alarm Rate	-$249K	$393K	$986K	$994K	$216K	0.20	-$778K	$0K
3 High Eff, Common Modes	-$249K	$396K	$1042K	$877K	$1026K	0.20	-$16K	-$166K
4 Low Mitigatability	-$249K	$396K	$989K	$744K	$763K	0.10	-$226K	-$245K
5 Low Cost of Recovery	-$249K	$396K	$989K	$1009K	$1051K	0.10	$0K	-$42K
6 High Fault Rate	-$25385K	-$12172K	-$268K	$1003K	$867K	0.05	-$137K	$0K
Expected Value	-$1506K	-$233K	$937K	**$953K**	$832K	1.00	-$188K	-$68K

Uncertainty Check:	**$120K**	
	Point-design (recov.)	**Uncert. design (red.)**
Value Improvement:	$2338K	$2459K
EVPI:	$188K	$68K
Threshold – Baseline	$2338K	$2459K
Threshold – Inspection	$1066K	$1186K
Threshold – PHM System (CBM)	**-$104K**	**$16K**
Threshold – Hot Redundancy	**-$120K**	X
Threshold – PHM System (Recovery)	X	$120K

6.3.4 Validity Test

The uncertainty check $\mathbf{T_1}$ fails because the chosen design is significantly different (both the design and the value of the design) when this uncertainty is considered, as shown in Figure 6.3. The

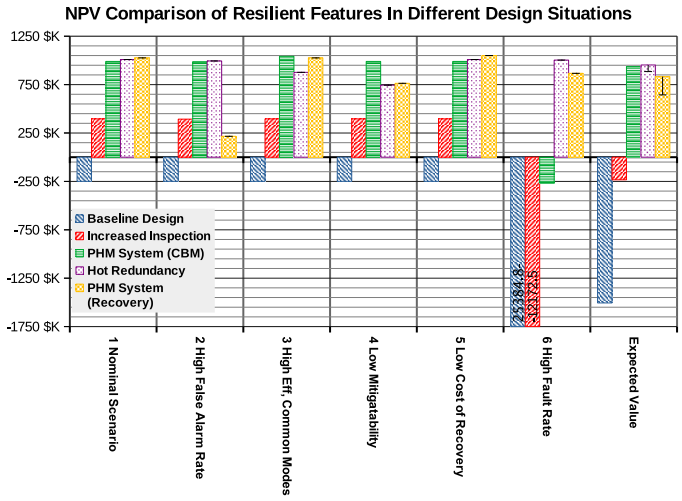


Figure 6.3: Comparing the value of fault mitigation features under different design situations. The first column is the design results generated using point-estimates while the last column is the design results generated when the uncertainty of being in each situation is considered.

difference in value between the point-case design in the nominal scenario considered in the design scenario and the optimal is $V_{UQ} = \$120K$, which makes the point case method invalid considering a reasonable value threshold $V_{UQ,des} = \$50K$. While the recovery system was chosen as the best in the nominal scenario, it becomes a much less favored option when the expected costs of the uncertain scenarios are considered, with the hot redundancy becoming the favored option. Using point estimates for this problem was therefore not valid because of the impact of uncertainties on choice.

The value test $\mathbf{T_2}$ succeeds because the chosen design has a much higher value than the initial design. Using the threshold of $V_{DP,des} = \$250K$, the design process passes, with an increase of $V_{DP} = \$2338K$ for the point-method design and $V_{DP} = \$2459K$ for the design chosen under uncertainty. The interpretation of this is that while the process would have been improved had it involved uncertainty, it still significantly increased the value of the design. While there are no counterfactual design processes to compare against to determine if this process was optimal, it still delivers on the promise of increasing design value early by mitigating risk due to faults.

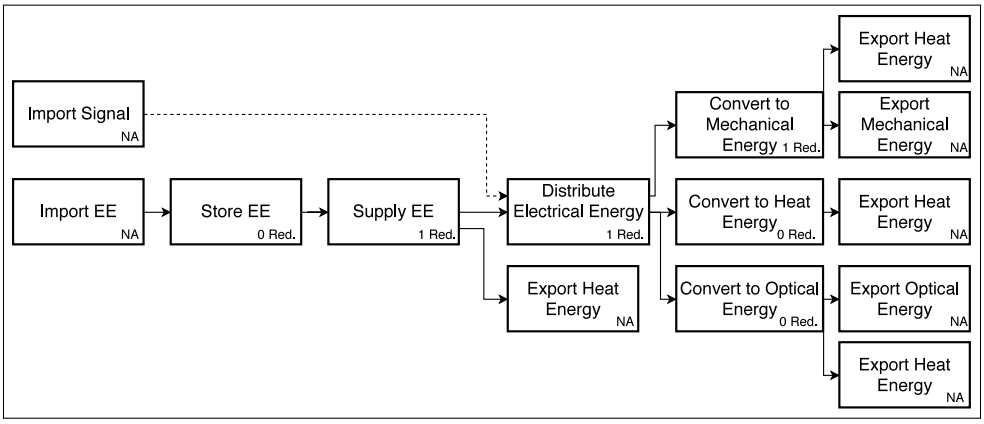


Figure 6.4: Functional model of electrical power distribution system with allocated redundancies noted.

The meaningfulness test $\mathbf{T_3}$ fails for both the point-estimate method and uncertainty-based design. The point-case method fails because the value of perfect information of the chosen design is $EVPI = \$188K$, which is larger than the difference between it and the other two best designs ($\$104K$ and $\$-120K$), which is expected since it failed the uncertainty check (making the differences negative). However, even if uncertainty was considered, the test would still fail because the $EVPI = \$68K$ is larger than the difference between the top two designs ($EVPI = \$16K$). As a result, while the design process can meaningfully justify using hot redundancy or a PHM system over a recovery system, increasing inspections, or using the baseline design, it cannot meaningfully justify the choice between a hot redundancy and PHM system. Given the magnitude of the difference in value for each of these designs is still high at a large scale (and the uncertainty at this stage is mostly reducible), further investigation of both options should be performed before making the decision.

To summarize, in this problem the choice of health management approach based on point-estimate information is found to be invalid because of the high level of uncertainty in input estimates.

6.4 Example: EPS System Redundancy Allocation

To demonstrate the continuous parametric uncertainty quantification and validation techniques, this section considers a simple example of optimization of redundancy within an electric power system using the decomposition strategy outlined in Section 5.3. This power system has been considered in previous work studying function failure approaches [158, 130, 185] and draws inspiration from the ADAPT power distribution system used to test diagnostic algorithms [232, 233, 155]. As shown in the functional model in Figure 6.4, the purpose of this system is to store and supply electrical energy to mechanical, heat, and optical loads using provided electrical energy and signal input. For the purpose of this example, the value in the system is derived from the quality of these output flows.

6.4.1 Value Model

The value model for this system is based on the design cost of each redundancy, the operational costs of replacing components, and the failure costs of a mode occurring in use. Failure costs in the system are a result from modes occurring that result in degraded or lost output flows of mechanical, heat, or optical energy in the `Export Mechanical Energy`, `Export Heat Energy`, and `Export Optical Energy` functions. The failure cost C_F over the whole system, considering all functions and modes is then:

$$C_F = \sum_{n \in N} \sum_{m \in M_n} T * -ln(1 - (e^{\lambda_m t})^{(x_n+1)}) * C_{s \sim m} \tag{6.19}$$

where n is a given function in the set of functions N, m is a mode in the set M_n of modes for function n, T is the life-cycle time (which was set to 43800 hours or 5 years in this example), λ_m is the rate of failure over unit time t, x_n is the number of redundancies for the function, and C_m is the cost of the mode. Each function, mode, effect, rate, and cost is provided in Table 6.3. It should be noted that this equation assumes that the failed redundant component will be replaced promptly, since over the time the component is not replaced the probability of failure is the failure probability of the remaining components.

Operational costs result from replacing failed components over the life-cycle of the product. This cost increases with redundancy, since more components are in use at any one time (assuming a hot standby configuration). The resulting expected maintenance cost for an individual mode m in function n is:

$$C_O = \sum_{n \in N} \sum_{m \in M_n} T * -ln(1 - (e^{\lambda_m t})) * (x_n + 1) * C_n \tag{6.20}$$

where the notation is consistent with Equation 6.20 and C_n is the cost of each redundancy provided in Table 6.3

Finally, the design cost from redundancies comes from the initial installation of the component for each function:

$$C_D = \sum_{n \in N} (x_n + 1) * C_n \tag{6.21}$$

where notation is consistent with Equation 6.20 and 6.19.

6.4.2 Design Process

Since several decisions are made in this design problem (about each redundancy), finding the best design requires an optimization approach. Since the cost of each function increases linearly with the number of redundancies and the failure probability decreases (approximately) with the power of the redundancies, when increasing the number of redundancies from zero, the overall cost associated with each function is expected to either start at the minimum or decay exponentially before increasing linearly. Following this design approach, to optimize the score x_n is increased from zero until $C(x_n)$ stops increasing.

The results of this design approach are shown in Table 6.3, in the column labeled "Red. Added." As can be seen, redundancies are added in the functions where the associated design and operational cost is less than the failure cost. In those cases, the failure cost is reduced dramatically due to the exponential decrease in probability from moving to single-part failures to joint-part failures due to

Table 6.3: Design problem parameters and results

(a) Problem Parameters

Function	**Mode**	**Effect**	**Rate (occ/hr)**	**Effect Cost**
Store Electrical Energy	PartialStorage	Degraded Mech. Energy, Optical Energy, Heat rate, effort	5.00E-06	1500
Red. Cost: 1750	NoStorage	No Mech. Energy, Optical Energy, Heat rate, effort	5.00E-06	2700
Supply Electrical Energy	AdverseResistance	Low Mech. Energy, Optical Energy, Heat rate, effort	2.00E-06	1500
Red. Cost: 400	MinorOverloading	No Optical Energyrate, effort, high Mech. Energy rate, effort	1.00E-05	1400
	MajorOverloading	No Mech. Energy, Optical Energy, Heat rate, effort	3.00E-06	2700
	ShortCircuit	No Mech. Energy, Optical Energy, Heat rate, effort	1.00E-07	2700
	OpenCircuit	No Mech. Energy, Optical Energy, Heat rate, effort	5.00E-08	2700
Distribute Electrical Energy	AdverseResistance	Low Mech. Energy, Optical Energy, Heat rate, effort	1.00E-05	1500
Red. Cost: 1000	PoorAllocation	Degraded Mech. Energy, Optical Energy, Heat rate, effort	2.00E-05	2700
	ShortCircuit	No Mech. Energy, Optical Energy, Heat rate, effort	2.00E-05	2700
	OpenCircuit	No Mech. Energy, Optical Energy, Heat rate, effort	3.00E-05	2700
Convert Electrical Energy to Mechanical	HighTorque	High Mech. Energy rate, effort	1.00E-04	500
Red. Cost: 200	LowTorque	Low Mech. Energy rate, effort	5.00E-05	500
	TooHighTorque	High Mech. Energy rate, effort, no Optical Energy or Heat rate, effort	5.00E-05	2900
	OpenCircuit	No Mech. Energy rate, effort	5.00E-05	900
	ShortCircuit	No Mech. Energy, Optical Energy, Heat rate, effort	5.00E-05	2700
Convert Electrical Energy to Optical	Adverse Optical Resistance	Low Optical Energy rate, effort	5.00E-07	500
Red. Cost: 200	NoConversion	No Optical Energy rate, effort	2.00E-06	900
Convert Electrical Energy to Heat	NotEnoughHeat	low heat rate, effort	2.00E-06	500
Red. Cost: 200	Hot	High Heat rate, effort	1.00E-07	500
	TooHot	Highest Heat rate, effort	5.00E-07	1100
	OpenCircuit	No Heat rate, effort	1.00E-07	900

(b) Design Results and Associated Costs

Function	**Pre-Design Failure Cost**	**Red. Added**	**Post-Design Failure Cost**	**Design Cost**	**Operational Cost**	**Total Cost**
Store Electrical Energy	919.80	0	919.80	0	0.00	919.80
Sup Electrical Energy	1,117.12	1	0.01	400	265.43	665.44
Distribute Electrical Energy	8,935.30	1	0.21	1000	3,504.04	4,504.25
Convert Electrical Energy to Mechanical Energy	17,520.49	1	0.99	200	2,628.09	2,529.07
Convert Electrical Energy to Optical Energy	89.79	0	89.79	0	0.00	89.79
Convert Electrical Energy to Heat Energy	74.02	0	74.02	0	0.00	74.02
Total	28,656.53					9,082.37

the independence of part failure probabilities (an assumption of the design problem). The results in this example show a $20,000 decrease in overall costs due to failure from redundancy allocation–a reduction of roughly 70%.

6.4.3 Uncertainty Quantification

Because this design process is performed early, each of the numbers provided in Table 6.3 result from estimates which are prone to various biases. To model the effect of these biases, the uncertainty model form presented in Section 6.2.2 is adapted to consider the uncertainty in design costs, failure mode rates, and failure mode costs (other uncertainties, such as independence of redundancy failure rates are not considered, since this is a demonstration of the approach). In this example, the Monte Carlo method is used, which determines the distributions of output parameters by randomly picking a large number of values from the input parameter distributions and propagating them through the model. The formula used for individual design cost response $c_{n,i}$ for function n in sample point i is:

$$c_{n,i} = c_{n,est} * s_{cd,i} * \epsilon_{n,cd,i} \tag{6.22}$$

where $c_{n,est}$ is the estimate, $s_{cd,i}$ is the sample point drawn from the system-wide design cost bias distribution, and $\epsilon_{n,cd,i}$ is drawn from the error distribution of the cost of design for the function n.

The formula used for individual mode rate response $\lambda_{m,i}$ for mode m is:

$$\lambda_{m,i} = \lambda_{m,est} * s_{\lambda,i} * f_{n,\lambda,i} * \epsilon_{m,\lambda,i} \tag{6.23}$$

where $\lambda_{m,est}$ is the estimate, $s_{\lambda,i}$ is drawn from the system-wide rate bias distribution, $f_{n,\lambda,i}$ is drawn from the rate bias distribution for function n, and $\epsilon_{m,\lambda,i}$ is drawn from the error distribution of the individual mode rate.

Table 6.4: Expected costs of redundancies in each function (and the optimal design) considering uncertainty in parameter values.

Functions	**0 Red.**	**1 Red.**	**2 Red.**	**Opt. Design**	**VoD**	**EVPI**	$V^* - V^{\not*}$
Store Electrical Energy	1382.27	3207.55	6415.08	0 Red.	0	28.64	1826.28
Supply Electrical Energy	1683.62	870.96	1781.88	1 Red.	812.66	30.14	812.66
Distribute Electrical Energy	13268.84	6396.83	12792.72	1 Red.	6872.01	108.17	6395.89
Convert Electrical Energy to Mechanical Energy	26432.77	4226.13	8447.65	1 Red.	22206.64	0.0	4221.52
Convert Electrical Energy to Optical Energy	134.04	268.36	536.71	0 Red.	0	7.3	134.32
Convert Electrical Energy to Heat Energy	110.89	269.99	539.98	0 Red.	0	2.88	159.10
Total:	43011.43	15239.82	30514.02	13120.10	29891.31	177.13	134.34

The formula used for mode cost uncertainty $c_{m,i}$ for response i from mode m is:

$$c_{m,i} = c_{m,est} * s_{cm,i} * \epsilon_{m,cm,i} \tag{6.24}$$

where $c_{m,est}$ is the estimate, $s_{cm,i}$ is drawn from the system-wide mode cost bias distribution and $\epsilon_{m,cm,i}$ is drawn from the error distribution of the individual mode cost. Each of the distributions used in this example followed the Beta-PERT distribution using the Crystal Ball software package [210], and were set to vary with a range parameter of $R = 2$ for the maximum and minimum values and the default peakedness.

6.4.4 Validity Test

The validity of this process is tested by quantifying uncertain distributions and using the Monte Carlo technique to calculate the expected values of each test metric. Here a Beta-PERT distribution is used to quantify the uncertainty in the system, function, parameter, and instantiation errors in Eq. 6.2 for failure rates and design and operational costs according to $PERT(a = 1/R, 1, R)$, where the range R was set to 2 for all parameters. By simulating the value model using these uncertain distributions as inputs, the uncertainty check, the increase in value test, and meaningfulness test are performed below.

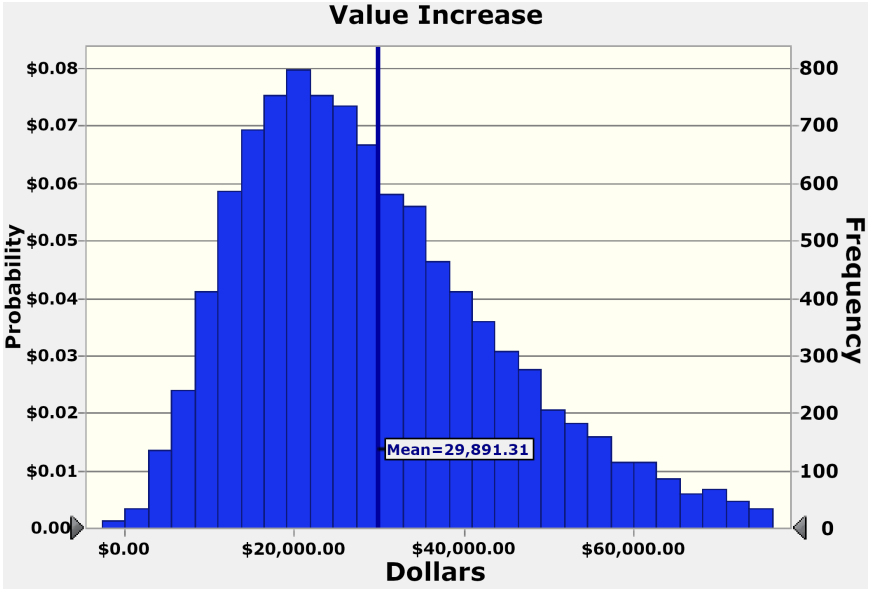


Figure 6.5: The distribution of value from choosing the point-estimate design compared to the baseline no-redundancy design.

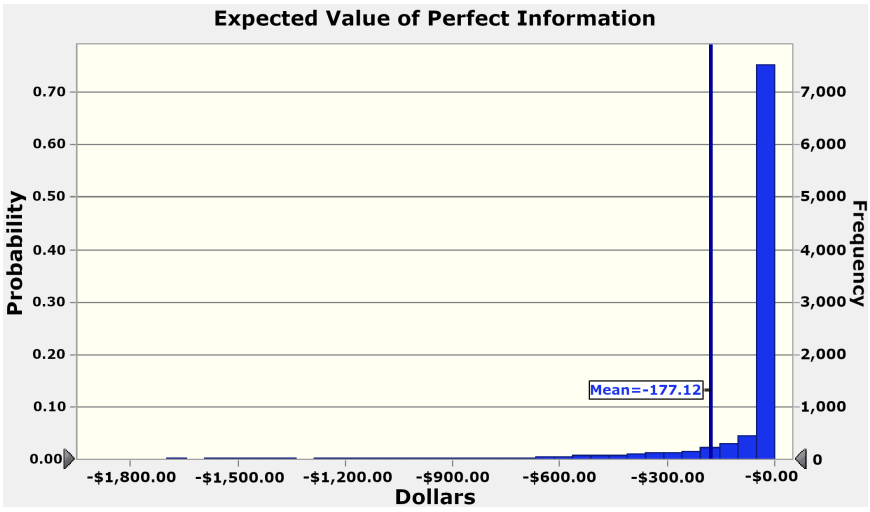


Figure 6.6: Distribution of loss of value from choosing the point-estimate design over a design generated with perfect information.

The design process passes the uncertainty check $\mathbf{T_1}$ because even though the predicted values of the design are different, the same optimal design is chosen when considering uncertainty as when using the estimates, as shown in Table 6.4. Interestingly, the increase in value of the design using this approach is larger than would be predicted from point-estimates, which likely is a result of the skew caused by the multiplier distributions being centered around 1 and varying between $1/R$ and R, which leads to a greater magnitude of error from under-estimation than over-estimation.

The design process passes the value test $\mathbf{T_2}$ because the value increase of the optimal design is very large. Using the threshold of $V_{DP,des} = 1000$, the process passes, since the expected value increase is $V_{DP} = 29891$, as shown in Figure 6.5. However, this may no longer hold if, for example, there is a more established process that can achieve better value. Looking at the value of individual design decisions shown in Table 6.4, it can be seen that the process was not necessary for the `Store Electrical Energy`, `Convert Electrical Energy to Optical Energy`, and `Convert Electrical Energy to Heat Energy` functions since the baseline design was kept. This does not invalidate the overall process, since other variables contributed a large enough value to compensate, but instead shows these variables were not worth exploring.

Finally, the design process passes the meaningfulness test $\mathbf{T_3}$ because the cost of uncertainty is much less than the threshold set. Using the conservative threshold of $EVPI_{des} = 1000$, the process still passes with $EVPI = 177.13$, as shown in Figure 6.6. This result continues to hold when considering each individual decision, as shown in the right two columns of Table 6.4, in which the value of information for each decision is much lower than the value difference between it and the next best decision. However, when viewed as a whole, the total value of information is larger than the next-best designs in which there are redundancies in the functions `Convert Electrical Energy to Optical Energy` and `Convert Electrical Energy to Heat Energy`, respectively. This means that if the value information was not able to be calculated on a single decision-basis, those decisions would be considered arbitrary compared to the level of uncertainty present system-wide.

Thus, the process of choosing redundancies in early design is validated on this problem based on the uncertainty check, value test, and meaningfulness test, since the decisions had a high impact

compared to the uncertainty present in estimates.

6.5 Conclusions

This chapter defined three concerns that must be addressed by a resilience-based design process to be accepted as valid: whether a design selection approach required uncertainty quantification, how effective the design approach was at increasing design value, and whether the design results are certain enough to justify a given choice. Three tests were provided in Section 6.2 to quantify each of these concerns: the uncertainty check (whether the result changes when uncertainty is quantified), the value improvement test (how much the expected value of the design increased), and the meaningfulness test (whether the value of the decision is higher than the cost of information). As shown in the provided example problems, these tests can then be used to determine the validity of a given design approach when the effect of epistemic uncertainty is quantified. In a simple example of choosing a health management approach for a system in Section 6.3, the testing approach showed that the early design approach should have incorporated uncertainty to account for the different potential design scenarios and shows that the choice between the top two designs is not sufficiently justified. In the second example in Section 6.4 considering redundancy allocation in early design, the testing approach showed that considering uncertainty is not required to make optimal choices and that the choices made have a high enough impact on value compared to the choice-affecting uncertainty to be certain. These examples demonstrate how uncertainty quantification can be used as a framework to understand the validity of early risk-based and resilient design approaches.

However, the primary limitation of this method is that it relies on a subjective uncertainty assessment to validate a subjective decision-making process. Thus, if one doubts the usefulness of an early design decision-making framework (e.g., per arguments in Ref. [294]) one might justifiably also doubt the results of this validity testing framework, especially when the designer realizes the same biases in the validity test as they did in the decision-making process (e.g., motivated reasoning). Nevertheless, the value of this approach when diligently carried out is that it finds cases where

decisions were not sufficiently justified given the uncertainty–an important issue in decision-making [69]–so that those design processes can be improved in the future. Thus, future work should show show how to use previous data and psychological studies regarding estimation biases to inform the assessment. For example, while a multiplicative Beta-PERT uncertainty model is provided in Section 6.2.2.2, other models such as an additive model or a model without biases but in which errors correlate with each other could be used for the same approach. An important question for future research, then, is how best to represent this underlying parameter uncertainty, given error can result from a number of different interacting sources and different model forms may affect the results of the validity tests presented here. While some work has been performed to link designer biases in decision-making with real-world accidents [141], and quantified distributions for estimation error [69, 302, 250], future work needs to develop a model of this in the resilient design process and investigate how different uncertainty models affect the resulting determination of validity.

Second, it should be noted that the process shown here enables validation of a method over a *specific instance* of a problem. While this enables the designer to answer questions about whether a specific method was applicable to a specific problem, it does not provide a general assessment of where a design process will apply. While it could be argued that the demonstrations here are archetypal of most general design cases, and, thus, the results shown here should hold for most problems, they do not constitute proof of general validity. To approach this problem, future work needs to explore the allowable ranges of parameters for certain specific values given specific levels of uncertainty. Using an approach like this could additionally show the ranges where the use of heuristics is valid, which is an important question for design theory.

Chapter 7: Conclusions

7.1 Summary

The preceding chapters presented methods and examples of how to include resilience in design decision making (Chapter 3), how to model resilience for use in early design (Chapter 4), how to optimize resilience in an integrated framework with design and operational costs (Chapter 5), and how to understand the impact of uncertainty on the validity of the design process (Chapter 6. While these methods stand on their own, they were developed to integrate with each other to comprise a design framework for early-stage resilience-based design. As demonstrated in the provided examples, (and noted extensively in Table 1.2), while it is not strictly necessary to use the methods together, the methods naturally integrate to enable a coherent resilience-based design process.

In addition to outlining and demonstrating these methods, there was also some study about how they work and when/how to use each. Section 3.6 studies the use of different quadratures when using fault sampling approaches in resilience quantification, with some advice and discussion on how to balance simulation cost and fidelity in design. Section 5.5 compares optimization architectures which can be be used in integrated resilience optimization on the basis of effectiveness and computational cost, and discusses potential use-cases for each. Finally, Section 6 provides examples of when a design process might be valid or invalid based on the amount of uncertainty present in model used to make the decisions. However, aside from these explorations of the particular details of the methods, there has been less overall discussion of this work to a general design scenario.

Thus, to conclude this work, the next sections first assess and discuss how and where these methods may be applied (Section 7.2) and gaps which may be filled in future work (Section 7.3).

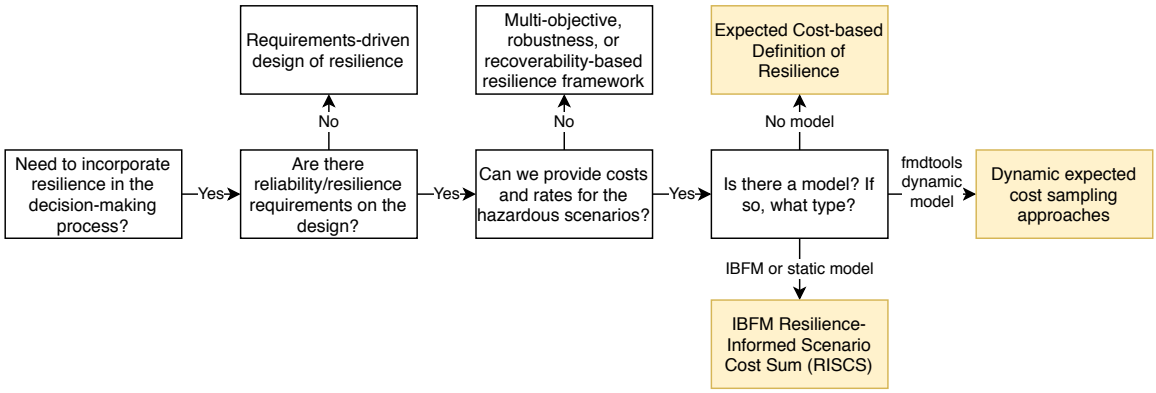


Figure 7.1: Context and purpose of cost-based resilience objectives in Chapter 3.

7.2 Assessment of Methods

Much of the overall design framework presented here relies on the expected cost-based definition of resilience presented in Chapter 3. It forms both the objectives used in the optimization problems formulated in Chapter 5 and the basis of the validity testing framework in Chapter 6. However, as shown in Figure 7.1, using a cost-based objective is not always appropriate because of the formulation of the design problem. Indeed, many risk, reliability, and resilience frameworks formulate the problem as an optimization of costs subject to risk constraints because these are given to the designer as requirements. While there is a strong case for using an inclusive cost function (i.e., one which includes safety) as the primary driver in design (see Section 2.1.2), when hazards in the system have impact on safety, stakeholders are often more comfortable setting a requirement for hazard probability than directly trading it against design and operational costs. As a result, when in this design situation, designers would be better suited to using a more traditional requirements-driven design approach (reliability [166] or safety engineering [136]) than the value-driven approach presented in this work. Additionally, another assumption underlying this work is that rates and cost models hazardous scenarios can be developed with any sort of certainty. While this work additionally provided a validity testing framework understand and control the effect of uncertainty in the design process (Chapter 6), this validity testing framework can only work if uncertainties are manageable enough to be quantified. In this situation, when the designer still wishes to make the

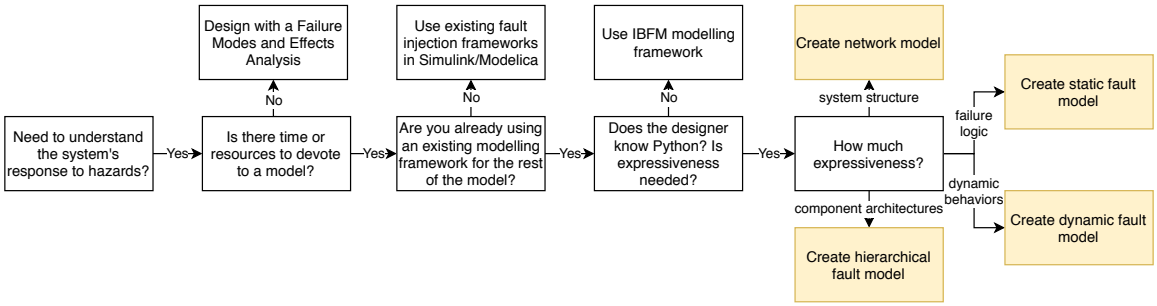


Figure 7.2: Context and purpose of the modelling framework in Chapter 4 in an overall design process.

system resilient to hazards, they may instead consider using a multi-objective [265], robustness [292] or recoverability-based [38] resilience framework that is less dependent on explicitly modelling the hazards to take into account but instead on building generic properties into the system which ensure the system will be resilient to a large number of scenarios.

This work additionally presented a modelling, simulation and analysis framework and toolkit for resilient design in Chapter 4. While this framework has a number of advantages (e.g., expressiveness, extend-ability, etc.) that make it suitable for resilience research, there are many design cases where one might use a different framework instead. Thus, the decision tree for using fmdtools or some other framework is shown in Figure 7.2. Typically in the design process, expert-driven approaches such as failure modes and effects analysis [217, Sec. 10.4] or fault-tree analysis [283] are used–while using a model-based can improve on this by enabling an iterative design process, using models in design is still contingent on having the design resources to devote to model-building. When a model is used, it may also be helpful to use an existing modelling framework (e.g., simulink or modelica) that is already commonly used in the model-based engineering of the system [136]. Finally, the usage of fmdtools is highly contingent on designer Python knowledge and thus it may be easier to use other frameworks (e.g. IBFM [185], which is also used in this work) to model fault scenarios. Nevertheless, fmdtools has applicability when these conditions are met, and is an ideal tool for conducting resilience *research*, where it is more important to enable the modeller the ability to express complex behaviors in code and extend the modelling paradigm to meet research needs.

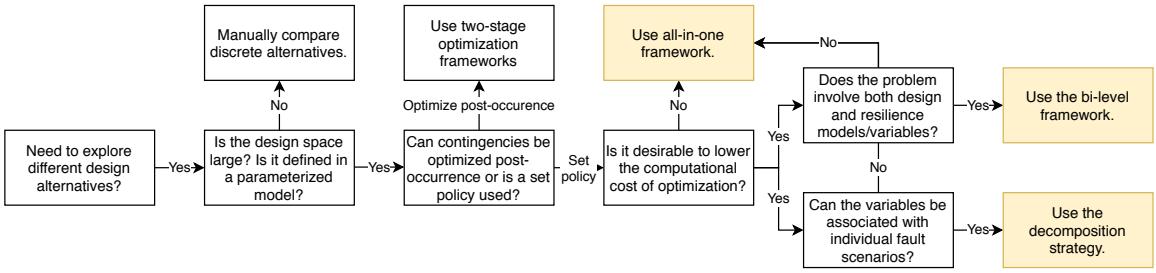


Figure 7.3: Context and purpose of the optimization frameworks in Chapter 5 in a design process.

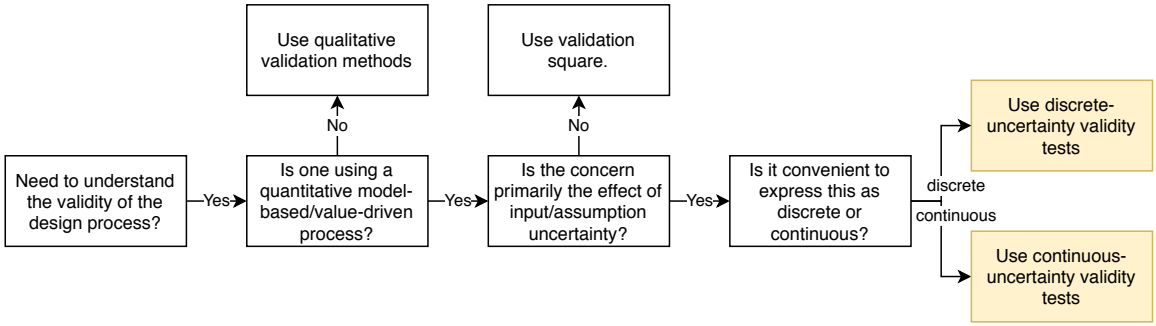


Figure 7.4: Context and purpose of the validation framework in Chapter 6 in a design process.

To enable the automated design of resilience in large design spaces, this work presents an integrated resilience optimization framework and optimization architectures for efficiently and effectively structuring the problem in Chapter 5. However, optimization is not the only method of exploring the design space and is not always applicable in the design process. If the design variants are highly different and cannot be developed from the same model, the design space is low, or it is difficult to parameterize the model(s) it may be preferable to use a simple concept selection process or simply iteratively improve the design instead of setting up a formal optimization process. Nevertheless, when optimization of resilience is applicable to the design process the methods in Chapter 5 will be useful for understanding and structuring the problem.

Finally, Chapter 6 provided a validity testing framework for understanding how uncertainty effects the design process. However, this validity testing framework is only applicable for considering a specific question of validity–the effect of uncertainty on the underlying design decisions in a value-based framework. However, validity in general is a much broader concern than just uncertainty

in design inputs, but also the underlying model structures and thought processes used to make decisions [209]. Thus, when thinking of applying Chapter 6 to validate a design process, one should first go though the process shown in Figure 7.4. If one is not using a value-driven design process or has broader concerns than the validity of design under input uncertainty, it may be better to use existing methods, such as the validation square [251].

7.3 Future work

The unified framework for design, modelling, optimization, and validation of resilience presented here is in no way an exhaustive study of every detail of each individual topic. Indeed, taking this framework as a baseline formulation of the resilient design problem opens a number of research questions that may be addressed in future work. To close this work, here are some possible research questions to be addressed in future work, based on limitations of this baseline framework:

- *How can one approach high uncertainty in fault scenario rates and probabilities in early design? What probability model is appropriate for approaching deep uncertainties about hazard scenarios and addressing second-order safety concerns?*

  While Chapter 3 presented a general definition of resilience as an expected cost of hazards, there are a number of limitations regarding the methods used to quantify this expectation. Specifically, one needs to be able to identify the set of scenarios (e.g. single-fault, joint-fault, etc.) to take into account in the expectation and develop a probability model for those scenarios. However, there are a number of difficulties in estimating these probabilities, especially when predicting low-probability unprecedented events–a rate may not be available and any rate taken from previous systems may not apply to the new system, and the designer may not have a good understanding of what hazards may occur [55, 225]. Thus, future work needs to investigate means of identifying (automatically from a model or discursively) high-consequence failures (e.g., [150]) and determine how to estimate/specify a probability model when hazard scenario information is sparse.

- *What is the relationship between resilience frameworks? Can sub-metric optimization(s) be performed while still optimizing expected cost?*

  As discussed in background Section 2.1, a number of reliability and resilience frameworks have been presented in previous work which optimize different parts of the hazard response–preventing the hazard from occurring or recovering after it happens. The restatement of the resilience-based design problem in this work as an optimization of expected cost is much more comprehensive than these other definitions (e.g., [167, 38]), however, there are downsides to this approach, especially when the design becomes more detailed, and it may be unnecessary to create a value model when there are not significant trade-offs to consider (e.g., if one is not allocating features with trade-offs but optimizing the function of already-specified features). Thus, future works should identify how these perspectives on resilience can align and where it is, from the comprehensive resilience optimization point of view presented in this work, practicable to use more narrowly-defined resilience optimization frameworks, as has previously been used to validate design heuristics [25].

- *How does making a model more detailed change ones consideration of resilience? Are there late-stage modelling considerations that need to be accounted for in early design models?*

  Chapter 4 presents the fmdtools modelling framework, which can be used to represent systems at different levels of fidelity with network, static, dynamic, and hierarchical modelling paradigms. This movement through the design process from a low-detail sketch to a fully specified design is typical as one proceeds from conceptual design through the embodiment design process [217]. However, the elaboration of the design in this way also results in new behaviors, fault modes, and mitigating strategies. However, in general, the design decision-making processes presented here in the examples and in Chapter 5 are only consider design decision-making at a single moment in the design/modelling process given the information given by the model. Thus, there may be issues, for example, if one optimizes resilience early but then new hazards are introduced or identified as the design is elaborated. On the other hand, if one waits to consider resilience late in the design process, high-level decisions made

in the early design stage may lead one to be stuck with a too-fragile design when resilience is finally considered. As a result, the resilience-based design process needs to be studied, both to help designers predict and account for hazards which will be introduced by successive design work and to manage resilience throughout the design process (e.g., [276]).

- *How can one represent the complexities of complex engineered systems (human interactions, dynamic environment, distributed interactions, etc.) in dynamic resilience models?*

  In the modelling framework presented in Chapter 4, a generic modelling framework was developed for the resilience of engineered systems. However, real complex engineered systems have a number of specialized modelling concerns which need to be taken into account in design, especially the interactions between the human system and the machine cyber/automated interfaces. Previous work has developed methods for integrating the design of human tasks and interfaces with the functions of the system in early design [125, 264] and on quantifying expected costs of human-function interactions [126]. However, this work has not been integrated with the fmdtools modelling framework, which should be addressed in future work.

- *How can stochastic input and model parameters be taken into account in fault-injection-based resilience modelling?*

  One of the major limitations of the fmdtools modelling framework in Chapter 4 is that it is a deterministic framework, while many engineered systems behaviors (especially in hazardous scenarios) are uncertain or probabilistic. Thus, when modelling the different hazard scenarios, fmdtools can only consider the set of input fault scenarios, rather than the full set of resulting possible contingencies. Methods have been put forward to do this in safety engineering [86, 301], but not in the design of resilience. While this project was able to model some uncertainty through input parameters in the fire response model example in Section 4.4, it is currently unable to model uncertain behavioral assumptions. Thus, future work needs to extend the fmdtools framework to systematically incorporate both uncertain input assumptions and uncertain behavioral assumptions.

- *How can one best balance simulation cost and accuracy when using fault scenarios to represent expected costs in integrated resilience optimization?*

  Chapter 5 presented the idea of integrating the optimization of resilience with the rest of the system design and introduced a number of architectures to structure this optimization efficiently. However, for this optimization to be in any way efficient, it needs to run over a reduced set of faults, since the space of potential scenarios is very large. Thus, in the example in Section 5.5, only the single-fault scenarios were considered at a single time. However, choosing reduced subsets of fault scenarios has the opportunity to bias the optimization or resilience toward those scenarios instead of the full set. Future work needs to how to best choose this representative set of scenarios to best reduce simulation cost without biasing the resilience optimization. Additionally, there is some opportunity to explore more sophisticated optimization architectures to enable a more computationally-efficient optimization process. For example, the use of surrogate resilience models could enable consideration of resilience in a (more efficient) sequential approach. Furthermore, it may be possible to limit the number of lower-level optimizations in the bilevel formulation by re-using solutions from each optimization. Thus, future work should explore additionally strategies to reduce computational cost in resilience optimization.

- *How does one test the validity of more complicated design/optimization processes? What about processes where the form of the model changes over time?*

  Chapter 5 presented methods for testing the validity of a design process. However, the example design processes presented in the chapter were quite simple, looking at either a simple optimization problem or a single design evaluation. Real design problems often are not that simple–having higher dimensionality and more heterogeneous input uncertainty. For example, the optimization example in Chapter 5 had a large number of variables and a complex optimization strategy. Thus, in this situation, calculating the validity tests may be much more difficult than presented. Future work should study how to perform these tests in more complicated optimization problems, such as those presented in Chapter 5.

These are just a few questions which may be answered in future work. The framework presented here is a baseline formulation of a framework which can be used for the design of resilience. In this way, the work here sets the stage for future contributions to the field.

Bibliography

[1] Galileo end of mission press kit. Technical report, National Aeronautics and Space Administration, 2003.

[2] Milton Abramowitz and Irene A Stegun. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, volume 55. US Government printing office, 1948.

[3] Masakazu Adachi, Yiannis Papadopoulos, Septavera Sharvia, David Parker, and Tetsuya Tohdo. An approach to optimization of fault tolerant architectures using hip-hops. *Software: Practice and Experience*, 41(11):1303–1327, 2011.

[4] Jose Ignacio Aizpurua and Eñaut Muxika. Model-based design of dependable systems: limitations and evolution of analysis and verification approaches. *International Journal on Advances in Security*, 6(1), 2013.

[5] Arden Albee, Steven Battel, Richard Brace, Garry Burdick, John Casani, Jeffrey Lavell, Charles Leising, Duncan MacPherson, Peter Burr, and Duane Dipprey. Report on the loss of the mars polar lander and deep space 2 missions. 2000.

[6] Tunc Aldemir. Computer-assisted markov failure modeling of process control systems. *IEEE Transactions on reliability*, 36(1):133–144, 1987.

[7] Tunc Aldemir. A survey of dynamic methodologies for probabilistic safety assessment of nuclear power plants. *Annals of Nuclear Energy*, 52:113–124, 2013.

[8] Karen Allenby and Tim Kelly. Deriving safety requirements using scenarios. In *Proceedings Fifth IEEE International Symposium on Requirements Engineering*, pages 228–235. IEEE, 2001.

[9] J Allison. Complex system optimization: A review of analytical target cascading, collaborative optimization, and other formulations, 2004.

[10] James T Allison and Daniel R Herber. Special section on multidisciplinary design optimization: multidisciplinary design optimization of dynamic engineering systems. *AIAA journal*, 52(4):691–710, 2014.

[11] Louis Anthony (Tony) Cox Jr. What's wrong with risk matrices? *Risk Analysis: An International Journal*, 28(2):497–512, 2008.

[12] SAE ARP. 4761. *Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment*, 2, 1996.

[13] Victor Arribas, Svetla Nikova, and Vincent Rijmen. Vermi: Verification tool for masked implementations. In *ICECS*, pages 381–384. IEEE, 2018.

[14] J Ash and D Newth. Optimizing complex networks for resilience against cascading failure. *Physica A: Statistical Mechanics and its Applications*, 380:673–683, 2007.

[15] Jason Matthew Aughenbaugh and Christiaan J Paredis. The value of using imprecise probabilities in engineering design. *Journal of Mechanical Design*, 128(4):969–979, 2006.

[16] Terje Aven. On how to define, understand and describe risk. *Reliability Engineering & System Safety*, 95(6):623–631, 2010.

[17] Terje Aven. Risk assessment and risk management: Review of recent advances on their foundation. *European Journal of Operational Research*, 253(1):1–13, 2016.

[18] Terje Aven. How some types of risk assessments can support resilience analysis and management. *Reliability Engineering & System Safety*, 167:536–543, 2017.

[19] Terje Aven and Roger Flage. Use of decision criteria based on expected values to support decision-making in a production assurance and safety setting. *Reliability Engineering & System Safety*, 94(9):1491–1498, 2009.

[20] Sweewarman Balachandran and Ella Atkins. Markov decision process framework for flight safety assessment and management. *Journal of Guidance, Control, and Dynamics*, 40(4):817–830, 2017.

[21] Albert-László Barabási. Scale-free networks: A decade and beyond. *Science*, 325(5939):412–413, 2009.

[22] Evelyn ML Beale. On minimizing a convex function subject to linear inequalities. *Journal of the Royal Statistical Society: Series B (Methodological)*, 17(2):173–184, 1955.

[23] M Belhadj and T Aldemir. The cell to cell mapping technique and chapman-kolmogorov representation of system dynamics. *Journal of sound and vibration*, 181(4):687–707, 1995.

[24] Dimitri Bettebghor, Stéphane Grihon, and Joseph Morlier. Bilevel optimization of large composite structures based on lamination parameters and post-optimal sensitivities. part 1: Theoretical aspects. *Article Manuscript, May*, 2018.

[25] William R Binder and Christiaan JJ Paredis. Optimization under uncertainty versus algebraic heuristics: A research method for comparing computational design methods. In *ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages V02BT03A057–V02BT03A057. American Society of Mechanical Engineers, 2017.

[26] Arpan Biswas, Yong Chen, Nathan Gibson, and Christopher Hoyle. Bilevel flexible-robust optimization for energy allocation problems. *ASCE-ASME J Risk and Uncert in Engrg Sys Part B Mech Engrg*, 6(3), 2020.

[27] Arpan Biswas, Yong Chen, and Christopher Hoyle. A bi-level optimization approach for energy allocation problems. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 51760, page V02BT03A041. American Society of Mechanical Engineers, 2018.

[28] Arpan Biswas and Christopher Hoyle. A literature review: Solving constrained non-linear bi-level optimization problems with classical methods. In *ASME 2019 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers Digital Collection.

[29] Kjartan Bjørnsen, Jon Tømmerås Selvik, and Terje Aven. A semi-quantitative assessment process for improved use of the expected value of information measure in safety management. *Reliability Engineering & System Safety*, 188:494–502, 2019.

[30] Mark Blackburn, Robert Busser, Aaron Nauman, Robert Knickerbocker, and Richard Kasuda. Mars polar lander fault identification using model-based testing. In *Proceedings 26th Annual NASA Goddard Software Engineering Workshop*, pages 128–135. IEEE, 2001.

[31] Mishap Investigation Board. Mars climate orbiter mishap investigation board phase i report november 10, 1999, 1999.

[32] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang. Complex networks: Structure and dynamics. *Physics Reports*, 424(4):175 – 308, 2006.

[33] J. Bracken and J.T. McGill. Defense applications of mathematical programs with optimization problems in the constraints. *Oper. Res.*, 22(5):1086–1096, 1974.

[34] SR Bradley and Alice M Agogino. An intelligent real time design methodology for component selection: an approach to managing uncertainty. *Journal of Mechanical Design*, 116(4):980–988, 1994.

[35] Robert Braun, Peter Gage, Ilan Kroo, and Ian Sobieski. Implementation and performance issues in collaborative optimization. In *6th Symposium on Multidisciplinary Analysis and Optimization*, page 4017, 1996.

[36] Michel Bruneau, Stephanie E Chang, Ronald T Eguchi, George C Lee, Thomas D O'Rourke, Andrei M Reinhorn, Masanobu Shinozuka, Kathleen Tierney, William A Wallace, and Detlof Von Winterfeldt. A framework to quantitatively assess and enhance the seismic resilience of communities. *Earthquake spectra*, 19(4):733–752, 2003.

[37] Peter Bunus, Olle Isaksson, Beate Frey, and Burkhard Münker. Rodon-a model-based diagnosis approach for the dx diagnostic competition. *Proc. DX'09*, pages 423–430, 2009.

[38] Beatrice Cassottana, Lijuan Shen, and Loon Ching Tang. Modeling the recovery process: A key dimension of resilience. *Reliability Engineering & System Safety*, 190:106528, 2019.

[39] Peter Chemweno, Liliane Pintelon, Peter Nganga Muchiri, and Adriaan Van Horenbeek. Risk assessment methodologies in maintenance decision making: A review of dependability modelling approaches. *Reliability Engineering & System Safety*, 173:64–77, 2018.

[40] A. Chen and K. Subprasom. Analysis of regulation and policy of private toll roads in a build-operate-transfer scheme under demand uncertainty. *Transp. Res. Part Policy Pract*, 41(6):537–558, July 2007.

[41] Wei Chen, George Kesidis, Tina Morrison, J. Tinsley Oden, Jitesh H. Panchal, Christiaan Paredis, Michael Pennock, Sez Atamturktur, Gabriel Terejanu, and Michael Yukish. *Uncertainty in Modeling and Simulation*, pages 75–86. Springer International Publishing, Cham, 2017.

[42] Xi Chen, Zhimin Xi, and Pengyuan Jing. A unified framework for evaluating supply chain reliability and resilience. *IEEE Transactions on Reliability*, 66(4):1144–1156, 2017.

[43] Abraham Cherfi, Michel Leeman, Florent Meurville, and Antoine Rauzy. Modeling automotive safety mechanisms: A markovian approach. *Reliability Engineering & System Safety*, 130:42–49, 2014.

[44] Ferdinando Chiacchio, Jose Ignacio Aizpurua, Lucio Compagno, and Diego D'Urso. Shyftoo, an object-oriented monte carlo simulation library for the modeling of stochastic hybrid fault tree automaton. *Expert Systems with Applications*, 146:113139, 2020.

[45] Ferdinando Chiacchio, Jose Ignacio Aizpurua, Lucio Compagno, Soheyl Moheb Khodayee, and Diego D'Urso. Modelling and resolution of dynamic reliability problems by the coupling of simulink and the stochastic hybrid fault tree object oriented (shyftoo) library. *Information*, 10(9):283, 2019.

[46] Stephen E Chick. Subjective probability and bayesian methodology. *Handbooks in Operations Research and Management Science*, 13:225–257, 2006.

[47] Pier Davide Ciampa and Björn Nagel. Agile paradigm: the next generation collaborative mdo for the development of aeronautical systems. *Progress in Aerospace Sciences*, 119:100643, 2020.

[48] Eric Coatanéa, Sarayut Nonsiri, Tuomas Ritola, Irem Y Tumer, and David C Jensen. A framework for building dimensionless behavioral models to aid in function-based failure propagation analysis. *Journal of Mechanical Design*, 133(12):121001, 2011.

[49] Reuven Cohen, Keren Erez, Daniel Ben-Avraham, and Shlomo Havlin. Resilience of the internet to random breakdowns. *Physical review letters*, 85(21):4626, 2000.

[50] Paul D Collopy and Peter M Hollingsworth. Value-driven design. *Journal of aircraft*, 48(3):749–759, 2011.

[51] Benoît Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel optimization. *Annals of operations research*, 153(1):235–256, 2007.

[52] Benoit Combemale, Xavier Crégut, Jean-Pierre Giacometti, Pierre Michel, and Marc Pantel. Introducing simulation and model animation in the mde topcased toolkit. 2008.

[53] B Cottam, E Specking, C Small, E Pohl, Gregory S Parnell, and Randy K Buchanan. Defining resilience for engineered systems. *Engineering Management Research*, 8(2):11–29, 2019.

[54] Bobby J Cottam, Eric A. Specking, Colin A. Small, Edward A. Pohl, Gregory S. Parnell, and Randy K. Buchanan. Defining Resilience for Engineered Systems. *Engineering Management Research*, 8(2):11, August 2019.

[55] Louis Anthony Cox Jr. Confronting deep uncertainties in risk analysis. *Risk Analysis: An International Journal*, 32(10):1607–1629, 2012.

[56] Fellipe Sartori da Silva and José Alexandre Matelli. Development of metrics for resilience quantification in energy systems. In *Proceedings of the Annual Conference of the PHM Society*, volume 11, 2019.

[57] George B Dantzig. Linear programming under uncertainty. *Management science*, 1(3-4):197–206, 1955.

[58] John C Day, Michel D Ingham, Richard M Murray, Leonard J Reder, and Brian C Williams. Engineering resilient space systems. *Insight*, 18(1):23–25, 2015.

[59] Paulo Victor Rodrigues De Carvalho. The use of functional resonance analysis method (fram) in a mid-air collision to understand some characteristics of the air traffic management system resilience. *Reliability Engineering & System Safety*, 96(11):1482–1498, 2011.

[60] William Denson, Greg Chandler, William Crowell, and Rick Wanner. Nonelectronic parts reliability data 1991. Technical report, Reliability Analysis Cennter, Griffiss AFB NY, 1991.

[61] Nicolas Dulac and Nancy Leveson. An approach to design for safety in complex systems. In *Int. Symposium on Systems Engineering (INCOSE)*, 2004.

[62] Simon Eckermann, Jon Karnon, and Andrew R Willan. The value of value of information. *Pharmacoeconomics*, 28(9):699–709, 2010.

[63] James N Elele, David H Hall, Allie R Farid, David J Turner, Mark E Davis, and David R Keyser. Lessons learned from independent verification and validation of models and simulations (work in progress). In *Proceedings of the Symposium on Theory of Modeling & Simulation-DEVS Integrative*, page 17. Society for Computer Simulation International, 2014.

[64] Mustafa Suphi Erden, Hitoshi Komoto, Thom J van Beek, Valentina D'Amelio, Erika Echavarria, and Tetsuo Tomiyama. A review of function modeling: Approaches and applications. *Ai Edam*, 22(2):147–169, 2008.

[65] E Euler. The failures of the mars climate orbiter and mars polar lander- a perspective from the people involved. In *Guidance and control 2001*, pages 635–655, 2001.

[66] Georges Fadel, Joshua Summers, Gregory Mocko, and ChristiaanParedis. Nsf sponsored engineering design and systems engineering foundations workshop final report. Technical report, Clemson University, 2015.

[67] Reza Faturechi, Eyal Levenberg, and Elise Miller-Hooks. Evaluating and optimizing resilience of airport pavement networks. *Computers & Operations Research*, 43:335–348, 2014.

[68] Kiri Feldman and Peter Sandborn. Analyzing the return on investment associated with prognostics and health management of electronic products. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 43277, pages 1401–1409, 2008.

[69] Baruch Fischhoff and Don MacGregor. Subjective confidence in forecasts. *Journal of Forecasting*, 1(2):155–172, 1982.

[70] Serghei Floricel and Roger Miller. Strategizing for anticipated risks and turbulence in large-scale engineering projects. *International Journal of project management*, 19(8):445–455, 2001.

[71] Paolo Franchin and Francesco Cavalieri. Probabilistic assessment of civil infrastructure resilience to earthquakes. *Computer-Aided Civil and Infrastructure Engineering*, 30(7):583–600, 2015.

[72] Stuart Fraser, Alarma Simpson, Ariel Núñez, Vivien Deparday, Simone Balog, Brenden Jongman, Richard Murnane, Nicolas Taillefer, Nicolas Chauvin, Olivier Morel, et al. Thinkhazard!—delivering natural hazard information for decision making. In *2016 3rd International Conference on Information and Communication Technologies for Disaster Management (ICT-DM)*, pages 1–6. IEEE, 2016.

[73] Bart D Frischknecht, Katie Whitefoot, and Panos Y Papalambros. On the suitability of econometric demand models in design for market systems. *Journal of Mechanical Design*, 132(12):121007, 2010.

[74] Alessio Gambi, Marc Müller, and Gordon Fraser. Asfault: Testing self-driving car software using search-based procedural content generation. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 27–30. IEEE, 2019.

[75] Paul H Garthwaite, Joseph B Kadane, and Anthony O'Hagan. Statistical methods for eliciting probability distributions. *Journal of the American Statistical Association*, 100(470):680–701, 2005.

[76] Giorgis Georgakoudis, Ignacio Laguna, Hans Vandierendonck, Dimitrios S Nikolopoulos, and Martin Schulz. Safire: Scalable and accurate fault injection for parallel multithreaded applications. In *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 890–899. IEEE, 2019.

[77] Paolo Giudici, Geof H Givens, and Bani K Mallick. *Wiley Series in Computational Statistics*. Wiley Online Library, 2013.

[78] Brunno Goldstein, Sudarshan Srinivasan, Naveen K Mellempudi, Dipankar Das, Leandro Santiago, Victor C. Ferreira, N. Solon, Sandip Kundu, and Felipe M. G. França. Reliability evaluation of compressed deep learning models. In *2020 IEEE 11th Latin American Symposium on Circuits Systems (LASCAS)*, 2020.

[79] Justin S Gray, John T Hwang, Joaquim RRA Martins, Kenneth T Moore, and Bret A Naylor. Openmdao: An open-source framework for multidisciplinary design, analysis, and optimization. *Structural and Multidisciplinary Optimization*, 59(4):1075–1104, 2019.

[80] Florian Grigoleit, Sebastian Holei, Andreas Pleuß, Robert Reiser, Julian Rhein, Peter Struss, and Jana v Wedel. The qsafe project–developing a model-based tool for current practice in functional safety analysis. 2016.

[81] Xiaoyu Gu, John E Renaud, Leah M Ashe, Stephen M Batill, Amrjit S Budhiraja, and Lee J Krajewski. Decision-based collaborative optimization. *Journal of Mechanical Design*, 124(1):1–13, 2002.

[82] Julia Gundermann, Artem Kolesnikov, Morgan Cameron, and Torsten Blochwitz. The fault library-a new modelica library allows for the systematic simulation of non-nominal system behavior. In *Proceedings of the 2nd Japanese Modelica Conference, Tokyo, Japan, May 17-18, 2018*, number 148, pages 161–168. Linköping University Electronic Press, 2019.

[83] Hengdao Guo, Ciyan Zheng, Herbert Ho-Ching Iu, and Tyrone Fernando. A critical review of cascading failure analysis and modeling of power system. *Renewable and Sustainable Energy Reviews*, 80:9–22, 2017.

[84] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[85] Yacov Y Haimes. On the definition of resilience in systems. *Risk Analysis*, 29(4):498–501, 2009.

[86] Aram P Hakobyan. *Severe accident analysis using dynamic accident progression event trees.* PhD thesis, The Ohio State University, 2006.

[87] Tyler Raymond Halbert. *An improved algorithm for sequential information-gathering decisions in design under uncertainty.* PhD thesis, 2015.

[88] Brandon Haley, Andy Dong, and Irem Y. Tumer. A comparison of network-based metrics of behavioral degradation in complex engineered systems. *Journal of Mechanical Design*, 138(12), 2016.

[89] Brandon Haley, Andy Dong, and Irem Y. Tumer. A comparison of network-based metrics of behavioral degradation in complex engineered systems. *Journal of Mechanical Design*, 138(12), 2016.

[90] George A Hazelrigg. A framework for decision-based engineering design. *Journal of Mechanical Design*, 120(4):653–658, 1998.

[91] George A Hazelrigg. An axiomatic framework for engineering design. *Journal of Mechanical Design*, 121(3):342–347, 1999.

[92] Ping He, Charles A Mader, Joaquim RRA Martins, and Kevin J Maki. Dafoam: An open-source adjoint framework for multidisciplinary design optimization with openfoam. *AIAA Journal*, 58(3):1304–1319, 2020.

[93] Mohammad Hejase, Arda Kurt, Tunc Aldemir, Ümit Özgüner, Sergio B Guarro, Michael K Yau, and Matt D Knudson. Quantitative and risk-based framework for unmanned aircraft control system assurance. *Journal of Aerospace Information Systems*, pages 57–71, 2018.

[94] Bergen Helms, Kristina Shea, and Frank Hoisl. A framework for computational design synthesis based on graph-grammars and function-behavior-structure. In *ASME 2009 international design engineering technical conferences and computers and information in engineering conference*, pages 841–851. American Society of Mechanical Engineers, 2009.

[95] Devanandham Henry and Jose Emmanuel Ramirez-Marquez. Generic metrics and quantitative approaches for system resilience as a function of time. *Reliability Engineering & System Safety*, 99:114–122, 2012.

[96] Daniel R Herber and James T Allison. Nested and simultaneous solution strategies for general combined plant and control design problems. *Journal of Mechanical Design*, 141(1), 2019.

[97] Daniel Ronald Herber. *Advances in combined architecture, plant, and control design*. PhD thesis, University of Illinois at Urbana-Champaign, 2017.

[98] C. S. Holling. Resilience and Stability of Ecological Systems. *Annual Review of Ecology and Systematics*, 4:1–23, 1973.

[99] Crawford S Holling. Resilience and stability of ecological systems. *Annual review of ecology and systematics*, 4(1):1–23, 1973.

[100] Crawford Stanley Holling. Engineering resilience versus ecological resilience. *Engineering within ecological constraints*, 31(1996):32, 1996.

[101] Crawford Stanley Holling. Engineering resilience versus ecological resilience. *Engineering within ecological constraints*, 31(1996):32, 1996.

[102] Erik Hollnagel. *FRAM: the functional resonance analysis method: modelling complex socio-technical systems*. CRC Press, 2017.

[103] Nico B Holzel, Thomas Schilling, and Volker Gollnick. An aircraft lifecycle approach for the cost-benefit analysis of prognostics and condition-based maintenance-based on discrete-event simulation. Technical report, DLR-German Aerospace Center Hamburg Germany, 2014.

[104] Philipp Hönig, Rüdiger Lunde, and Florian Holzapfel. Model based safety analysis with smartiflow. *Information*, 8(1):7, 2017.

[105] Seyedmohsen Hosseini, Kash Barker, and Jose E. Ramirez-Marquez. A review of definitions and measures of system resilience. *Reliability Engineering & System Safety*, 145:47–61, January 2016.

[106] Seyedmohsen Hosseini, Kash Barker, and Jose E Ramirez-Marquez. A review of definitions and measures of system resilience. *Reliability Engineering & System Safety*, 145:47–61, 2016.

[107] Thomas J Howard, Stephen J Culley, and Elies Dekoninck. Describing the creative design process by the integration of engineering design and cognitive psychology literature. *Design studies*, 29(2):160–180, 2008.

[108] Christopher Hoyle, Irem Y Tumer, Alexander F Mehr, and Wei Chen. Health management allocation during conceptual system design. *Journal of Computing and Information Science in Engineering*, 9(2):021002, 2009.

[109] Chuck Hsiao, Richard Malak, et al. Considering risk attitude in a value of information problem. In *DS 80-3 Proceedings of the 20th International Conference on Engineering Design (ICED 15) Vol 3: Organisation and Management, Milan, Italy, 27-30.07. 15*, pages 093–102, 2015.

[110] Yunwei Hu. *A guided simulation methodology for dynamic probabilistic risk assessment of complex systems*. PhD thesis, 2005.

[111] Daniel Hulse, Sequoia Andrade, Eleni Spirakis, Hannah Walsh, and Misty Davies. Smart-stereo: Preliminary model description. Technical Report 20205007481, NASA Ames Research Center, September 2020. https://ntrs.nasa.gov/citations/20205007481.

[112] Daniel Hulse, Arpan Biswas, Christopher Hoyle, Irem Tumer, Chetan Kulkarni, and Kai Goebel. Exploring architectures for integrated resilience optimization.

[113] Daniel Hulse, Christopher Hoyle, Kai Goebel, and Irem Tumer. Using value assessment to drive phm system development in early design. In *Proceedings of the Annual Conference of the PHM Society*, volume 11, 2019.

[114] Daniel Hulse, Christopher Hoyle, Kai Goebel, and Irem Y Tumer. Optimizing function-based fault propagation model resilience using expected cost scoring. In *ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages V02AT03A052–V02AT03A052. American Society of Mechanical Engineers, 2018.

[115] Daniel Hulse, Christopher Hoyle, Kai Goebel, and Irem Y Tumer. Quantifying the resilience-informed scenario cost sum: A value-driven design approach for functional hazard assessment. *Journal of Mechanical Design*, 141(2), 2019.

[116] Daniel Hulse, Christopher Hoyle, Irem Tumer, Chetan Kulkarni, and Kai Goebel. Temporal fault injection considerations in resilience quantification. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers, 2020. IDETC2020-19287.

[117] Daniel Hulse, Christopher Hoyle, Irem Y Tumer, and Kai Goebel. Decomposing incentives for early resilient design: Method and validation. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 59193, page V02BT03A015. American Society of Mechanical Engineers, 2019.

[118] Daniel Hulse, Christopher Hoyle, Irem Y Tumer, and Kai Goebel. How uncertain is too uncertain? Validity tests for early resilient and risk-based design processes. *Journal of Mechanical Design*, pages 1–23, 2020.

[119] Daniel Hulse, Kagan Tumer, Christopher Hoyle, and Irem Tumer. Modeling multidisciplinary design with multiagent learning. *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*, 2018.

[120] Daniel Hulse, Hannah Walsh, Andy Dong, Christopher Hoyle, Irem Tumer, Chetan Kulkarni, and Kai Goebel. fmdtools: A fault propagation toolkit for resilience assessment in early design. 2020.

[121] Daniel Hulse, Hannah Walsh, and Hongyang Zhang. Designengrlab/fmdtools: v0.5.3-revision2, May 2020.

[122] Daniel Hulse, Hongyang Zhang, Arpan Biswas, and Hannah Walsh. DesignEngrLab/fmdtools: v0.5.4-optimization demonstration, October 2020.

[123] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[124] Ryan S Hutcheson and Katie Grantham. Does access to expert knowledge allow students to better assess risk? In *ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 145–149. American Society of Mechanical Engineers, 2012.

[125] Lukman Irshad, Salman Ahmed, H Onan Demirel, and Irem Y Tumer. Computational functional failure analysis to identify human errors during early design stages. *Journal of Computing and Information Science in Engineering*, 19(3), 2019.

[126] Lukman Irshad, Daniel Hulse, H. Onan Demirel, Irem Y. Tumer, and David C. Jensen. Introducing likelihood of occurrence and expected cost to human error and functional failure reasoning framework. In *ASME 2020 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. ASME, 2020.

[127] Takuto Ishimatsu, Nancy G Leveson, John P Thomas, Cody H Fleming, Masafumi Katahira, Yuko Miyamoto, Ryo Ujiie, Haruka Nakao, and Nobuyuki Hoshino. Hazard analysis of complex spacecraft using systems-theoretic process analysis. *Journal of Spacecraft and Rockets*, 51(2):509–522, 2014.

[128] J Jänsch and H Birkhofer. The development of the guideline vdi 2221-the change of direction. In *DS 36: Proceedings DESIGN 2006, the 9th International Design Conference, Dubrovnik, Croatia*, 2006.

[129] PA Jansma. Open! open! open! galileo high gain antenna anomaly workarounds. In *2011 Aerospace Conference*, pages 1–21. IEEE, 2011.

[130] D Jensen, Irem Y Tumer, and Tolga Kurtoglu. Design of an electrical power system using a functional failure and flow state logic reasoning methodology. *Prognostics and Health Management Society*, 2009.

[131] David Jensen, Irem Y Tumer, and Tolga Kurtoglu. Flow state logic (fsl) for analysis of failure propagation in early design. In *ASME 2009 International design engineering technical conferences and computers and information in engineering conference*, pages 1033–1043. American Society of Mechanical Engineers Digital Collection, 2009.

[132] Saurabh Jha, Subho S Banerjee, James Cyriac, Zbigniew T Kalbarczyk, and Ravishankar K Iyer. Avfi: Fault injection for autonomous vehicles. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 55–56. IEEE, 2018.

[133] Saurabh Jha, Timothy Tsai, Siva Hari, Michael Sullivan, Zbigniew Kalbarczyk, Stephen W Keckler, and Ravishankar K Iyer. Kayotee: A fault injection-based system to assess the safety and reliability of autonomous vehicles to faults and errors. *arXiv preprint arXiv:1907.01024*, 2019.

[134] Anjali Joshi and Mats PE Heimdahl. Model-based safety analysis of simulink models using scade design verifier. In *International Conference on Computer Safety, Reliability, and Security*, pages 122–135. Springer, 2005.

[135] Anjali Joshi and Mats PE Heimdahl. Behavioral fault modeling for model-based safety analysis. In *10th IEEE High Assurance Systems Engineering Symposium (HASE'07)*, pages 199–208. IEEE, 2007.

[136] Anjali Joshi, Mats PE Heimdahl, Steven P Miller, and Mike W Whalen. Model-based safety analysis. 2006.

[137] Sohag Kabir and Yiannis Papadopoulos. Applications of bayesian networks and petri nets in safety, reliability, and risk assessments: A review. *Safety science*, 115:154–175, 2019.

[138] Gregory J Kacprzynski, Michael J Roemer, and Andrew J Hess. Health management system design: Development, simulation and cost/benefit optimization. In *Proceedings, IEEE Aerospace Conference*, volume 6, pages 6–6. IEEE, 2002.

[139] Peter Kall, Stein W Wallace, and Peter Kall. *Stochastic programming*. Springer, 1994.

[140] Hanumanthrao Kannan, Bryan L Mesmer, and Christina L Bloebaum. Increased system consistency through incorporation of coupling in value-based systems engineering. *Systems Engineering*, 20(1):21–44, 2017.

[141] Vikranth Kattakuri. *FAILURES IN SPACECRAFT SYSTEMS: AN ANALYSIS FROM THE PERSPECTIVE OF DECISION MAKING*. PhD thesis, Purdue University Graduate School, 2019.

[142] Elham Keshavarzi. Resilient design for complex engineered systems in the early design phase. 2018.

[143] Elham Keshavarzi, Matthew McIntire, Kai Goebel, Irem Y. Tumer, and Christopher Hoyle. 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*, page V02AT03A043, August 2017.

[144] Jahon Khorsandi and Terje Aven. Incorporating assumption deviation risk in quantitative risk assessments: A semi-quantitative approach. *Reliability Engineering & System Safety*, 163:22–32, 2017.

[145] Nam-Ho Kim, Dawn An, and Joo-Ho Choi. *Prognostics and Health Management of Engineering Systems*. Springer International Publishing, Cham, 2017.

[146] Yusoon Kim, Yi-Su Chen, and Kevin Linderman. Supply network disruption and resilience: A network structural perspective. *Journal of operations Management*, 33:43–59, 2015.

[147] Steven Kmenta and Koshuke Ishii. Scenario-based failure modes and effects analysis using expected cost. *Journal of Mechanical Design*, 126(6):1027–1035, 2004.

[148] Steven Kmenta and Kosuke Ishii. Scenario-based fmea: a life cycle cost perspective. In *Proc. ASME Design Engineering Technical Conf. Baltimore, MD*, 2000.

[149] Martina Kollárová. Fault injection testing of openstack. *Ph. D. dissertation*, 2014.

[150] Mark Koren, Saud Alsaif, Ritchie Lee, and Mykel J Kochenderfer. Adaptive stress testing for autonomous vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1–7. IEEE, 2018.

[151] Melanie E Kreye, Yee Mey Goh, and Linda B Newnes. Manifestation of uncertainty-a classification. In *DS 68-6: Proceedings of the 18th International Conference on Engineering Design (ICED 11), Impacting Society through Engineering Design, Vol. 6: Design Information and Knowledge, Lyngby/Copenhagen, Denmark, 15.-19.08. 2011*, 2011.

[152] Daniel Krus and Katie Grantham Lough. Function-based failure propagation for conceptual design. *AI EDAM*, 23(4):409–426, 2009.

[153] Benjamin Kruse, Torsten Gilz, Kristina Shea, and Martin Eigner. Systematic comparison of functional models in sysml for design library evaluation. *Procedia CIRP*, 21:34–39, 2014.

[154] James Kurien and Maria Dolores R-Moreno. Costs and benefits of model-based diagnosis. In *2008 ieee aerospace conference*, pages 1–14. IEEE, 2008.

[155] Tolga Kurtoglu, Sriram Narasimhan, Scott Poll, David Garcia, Lukas Kuhn, Johan de Kleer, Arjan van Gemund, and Alexander Feldman. First international diagnosis competition-dxc'09. *Proc. DX'09*, pages 383–396, 2009.

[156] Tolga Kurtoglu and Irem Y Tumer. A graph-based fault identification and propagation framework for functional design of complex systems. *Journal of Mechanical Design*, 130(5):051401, 2008.

[157] Tolga Kurtoglu 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.

[158] Tolga Kurtoglu, Irem Y Tumer, and David C Jensen. A functional failure reasoning methodology for evaluation of conceptual system architectures. *Research in Engineering Design*, 21(4):209–234, 2010.

[159] M. Labbé, P. Marcotte, and G. Savard. A bilevel model of taxation and its application to optimal highway pricing. *Manag. Sci.*, 44(12):1608–1622, December 1998.

[160] Pierre-Etienne Labeau, Carol Smidts, and S Swaminathan. Dynamic reliability: towards an integrated platform for probabilistic risk assessment. *Reliability engineering & system safety*, 68(3):219–254, 2000.

[161] Andrew B Lambe and Joaquim RRA Martins. Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes. *Structural and Multidisciplinary Optimization*, 46(2):273–284, 2012.

[162] Joseph R Laracy and Nancy G Leveson. Apply stamp to critical infrastructure protection. In *Technologies for Homeland Security, 2007 IEEE Conference on*, pages 215–220. IEEE, 2007.

[163] Zsolt Lattmann, Adrian Pop, Johan De Kleer, Peter Fritzson, Bill Janssen, Sandeep Neema, Ted Bapty, Xenofon Koutsoukos, Matthew Klenk, Daniel Bobrow, et al. Verification and design exploration through meta tool integration with openmodelica. In *Proceedings of the 10 th International Modelica Conference; March 10-12; 2014; Lund; Sweden*, number 096, pages 353–362. Linköping University Electronic Press, 2014.

[164] Kuan Waey Lee, William Moase, Andrew Ooi, Chris Manzie, and Eric C Kerrigan. Optimization framework for codesign of controlled aerodynamic systems. *AIAA Journal*, 54(10):3149–3159, 2016.

[165] Cynthia A Lengnick-Hall, Tammy E Beck, and Mark L Lengnick-Hall. Developing a capacity for organizational resilience through strategic human resource management. *Human Resource Management Review*, 21(3):243–255, 2011.

[166] Elmer Eugene Lewis. *Introduction to reliability engineering*. Wiley, 1987.

[167] Junxuan Li and Zhimin Xi. Engineering recoverability: A new indicator of design for engineering resilience. In *ASME 2014 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages V02AT03A044–V02AT03A044. American Society of Mechanical Engineers, 2014.

[168] Zhaojun Steven Li and Mohammad S Mobin. System reliability assessment incorporating interface and function failure. In *Reliability and Maintainability Symposium (RAMS), 2015 Annual*, pages 1–8. IEEE, 2015.

[169] Igor Linkov, Todd Bridges, Felix Creutzig, Jennifer Decker, Cate Fox-Lent, Wolfgang Kröger, James H Lambert, Anders Levermann, Benoit Montreuil, Jatin Nathwani, et al. Changing the resilience paradigm. *Nature Climate Change*, 4(6):407, 2014.

[170] Katie Grantham Lough, Robert Stone, and Irem Y Tumer. The risk in early design method. *Journal of Engineering Design*, 20(2):155–173, 2009.

[171] Katie Grantham Lough, Robert B. Stone, and Irem Tumer. 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*, 2006.

[172] Karin Lunde, Rüdiger Lunde, and Burkhard Münker. Model-based failure analysis with rodon. In *Proceedings of the 2006 conference on ECAI 2006: 17th European Conference on Artificial Intelligence August 29–September 1, 2006, Riva del Garda, Italy*, pages 647–651. IOS Press, 2006.

[173] Suniya S Luthar, Dante Cicchetti, and Bronwyn Becker. The construct of resilience: A critical evaluation and guidelines for future work. *Child development*, 71(3):543–562, 2000.

[174] Cameron A MacKenzie and Chao Hu. Decision making under uncertainty for design of resilient engineered systems. *Reliability Engineering & System Safety*, 192:106171, 2019.

[175] Richard J Malak and Christiaan JJ Paredis. Validating behavioral models for reuse. *Research in Engineering Design*, 18(3):111–128, 2007.

[176] Joao Pedro Malere. Application of linear programming to optimize the cost-benefit of an ivhm system. Technical report, SAE Technical Paper, 2017.

[177] Joshua T Margolis, Kelly M Sullivan, Scott J Mason, and Mariah Magagnotti. A multi-objective optimization model for designing resilient supply chain networks. *International Journal of Production Economics*, 204:174–185, 2018.

[178] Joaquim RRA Martins and Andrew B Lambe. Multidisciplinary design optimization: a survey of architectures. *AIAA journal*, 51(9):2049–2075, 2013.

[179] Rolando Martins, Rajeev Gandhi, Priya Narasimhan, Soila Pertet, António Casimiro, Diego Kreutz, and Paulo Veríssimo. Experiences with fault-injection in a byzantine fault-tolerant protocol. In *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*, pages 41–61. Springer, 2013.

[180] José Alexandre Matelli and Kai Goebel. Conceptual design of cogeneration plants under a resilient design perspective: Resilience metrics and case study. *Applied Energy*, 215:736–750, 2018.

[181] Norm Matloff. Introduction to discrete-event simulation and the simpy language. *Davis, CA. Dept of Computer Science. University of California at Davis. Retrieved on August*, 2(2009):1–33, 2008.

[182] William A Maul, George Kopasakis, Louis M Santi, Thomas S Sowers, and Amy Chicatelli. Sensor selection and optimization for health assessment of aerospace systems. *Journal of Aerospace Computing, Information, and Communication*, 5(1):16–34, 2008.

[183] David May and Walter Stechele. An fpga-based probability-aware fault simulator. In *2012 International Conference on Embedded Computer Systems (SAMOS)*, pages 302–309. IEEE, 2012.

[184] Mohammadreza Mazidi, Navid Rezaei, Fatemeh Jahanbani Ardakani, Maryam Mohiti, and Josep M Guerrero. A hierarchical energy management system for islanded multi-microgrid clusters considering frequency security constraints. *International Journal of Electrical Power & Energy Systems*, 121:106134, 2020.

[185] Matthew G McIntire, Elham Keshavarzi, Irem Y Tumer, and Christopher Hoyle. 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 Digital Collection, 2016.

[186] Frank McKenna. Opensees: a framework for earthquake engineering simulation. *Computing in Science & Engineering*, 13(4):58–66, 2011.

[187] Ali Farhang Mehr, Irem Tumer, and Eric Barszcz. Optimal design of integrated systems health management (ishm) for improving the safety of nasas exploration missions: A multidisciplinary design approach. In *Sixth World Congress on Structural and Multidisciplinary Optimization, Rio de Janeiro, May*, 2005.

[188] Ali Farhang Mehr and Irem Y Tumer. Risk-based decision-making for managing resources during the design of complex space exploration systems. *Journal of Mechanical Design*, 128(4):1014–1022, 2006.

[189] Hoda Mehrpouyan, Brandon Haley, Andy Dong, Irem Y. Tumer, and Christopher Hoyle. Resilient design of complex engineered systems against cascading failure. In *ASME 2013 International Mechanical Engineering Congress & Exposition*, volume 12: Systems and Design, page V012T13A063, San Diego, 2013. ASME.

[190] Leila Meshkat, Steve Jenkins, Sanda Mandutianu, and Vance Heron. Automated generation of risk and failure models during early phase design. In *Aerospace Conference, 2008 IEEE*, pages 1–12. IEEE, 2008.

[191] M Messer, JH Panchal, JK Allen, F Mistree, V Krishnamurthy, B Klein, and PD Yoder. Designing embodiment design processes using a value-of-information-based approach with applications for integrated product and materials design. In *ASME 2008 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 823–840. American Society of Mechanical Engineers, 2008.

[192] Matthias Messer, Jitesh H Panchal, Janet K Allen, and Farrokh Mistree. Model refinement decisions using the process performance indicator. *Engineering Optimization*, 43(7):741–762, 2011.

[193] Matthias Messer, Jitesh H Panchal, Vivek Krishnamurthy, Benjamin Klein, P Douglas Yoder, Janet K Allen, and Farrokh Mistree. Model selection under limited information using a value-of-information-based indicator. *Journal of Mechanical Design*, 132(12):121008, 2010.

[194] Scott B Miles. Participatory disaster recovery simulation modeling for community resilience planning. *International Journal of Disaster Risk Science*, 9(4):519–529, 2018.

[195] Elise Miller-Hooks, Xiaodong Zhang, and Reza Faturechi. Measuring and maximizing resilience of freight transportation networks. *Computers & Operations Research*, 39(7):1633–1643, 2012.

[196] Raj Minhas, Johan De Kleer, Ion Matei, Bhaskar Saha, Bill Janssen, Daniel G Bobrow, and Tolga Kurtoglu. Using fault augmented modelica models for diagnostics. In *Proceedings of the 10 th International Modelica Conference; March 10-12; 2014; Lund; Sweden*, number 96, pages 437–445. Linköping University Electronic Press, 2014.

[197] Failure Mode. Failure mode / mechanism distributions, fmd-97. Technical report, Reliability Information Analysis Center (RIAC), Rome (NY, USA), 1998.

[198] Andrey Morozov, Kai Ding, Mikael Steurer, and Klaus Janschek. Openerrorpro: A new tool for stochastic model-based reliability and resilience analysis. In *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*, pages 303–312. IEEE, 2019.

[199] Andrey Morozov, Thomas Mutzke, Boqi Ren, and Klaus Janschek. Aadl-based stochastic error propagation analysis for reliable system design of a medical patient table. In *2018 Annual Reliability and Maintainability Symposium (RAMS)*, pages 1–7. IEEE, 2018.

[200] Herbert Moskowitz and William I Bullers. Modified pert versus fractile assessment of subjective probability distributions. *Organizational Behavior and Human Performance*, 24(2):167–194, 1979.

[201] Salim Moslehi and T Agami Reddy. Sustainability of integrated energy systems: A performance-based resilience assessment methodology. *Applied energy*, 228:487–498, 2018.

[202] Sascha Müller, Andreas Gerndt, and Thomas Noll. Synthesizing failure detection, isolation, and recovery strategies from nondeterministic dynamic fault trees. *Journal of Aerospace Information Systems*, 16(2):52–60, 2019.

[203] Haruka Nakao, Masa Katahira, Yuko Miyamoto, and Nancy Leveson. Safety guided design of crew return vehicle in concept design phase using stamp/stpa. In *Proc. of the 5: th IAASS Conference*, pages 497–501. Citeseer, 2011.

[204] M. E. J. Newman. *Networks*. Oxford University Press, New York, New York, 2010.

[205] Thomas M Niermann, Wu-Tung Cheng, and Janak H Patel. Proofs: A fast, memory-efficient sequential circuit fault simulator. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(2):198–207, 1992.

[206] Gang Niu and Junjie Jiang. Prognostic control-enhanced maintenance optimization for multi-component systems. *Reliability Engineering & System Safety*, 168:218–226, 2017.

[207] Kyoung-Won Noh, Hong-Bae Jun, Jae-Hyun Lee, Gyu-Bong Lee, and Hyo-Won Suh. Module-based failure propagation (mfp) model for fmea. *The International Journal of Advanced Manufacturing Technology*, 55(5-8):581–600, 2011.

[208] Younju Oh, Junbeom Yoo, Sungdeok Cha, and Han Seong Son. Software safety analysis of function block diagrams using fault trees. *Reliability Engineering & System Safety*, 88(3):215–228, 2005.

[209] Andrew T Olewnik and Kemper Lewis. On validating engineering design decision support tools. *Concurrent Engineering*, 13(2):111–122, 2005.

[210] EIDT Oracle. Crystal ball user's guide, 2012.

[211] Kate H Orwin and David A Wardle. New indices for quantifying the resistance and resilience of soil biota to exogenous disturbances. *Soil Biology and Biochemistry*, 36(11):1907–1912, 2004.

[212] Min Ouyang, Leonardo Dueñas-Osorio, and Xing Min. A three-stage resilience analysis framework for urban infrastructure systems. *Structural safety*, 36:23–31, 2012.

[213] Min Ouyang, Chuang Liu, and Min Xu. Value of resilience-based solutions on critical infrastructure protection: Comparing with robustness-based solutions. *Reliability Engineering & System Safety*, 190:106506, 2019.

[214] Andrew Owens and Olivier de Weck. Sensitivity analysis of the advanced missions cost maodel. 46th International Conference on Environmental Systems, 2016.

[215] Andrew Owens, Olivier de Weck, Chel Stromgren, Kandyce Goodliff, and William Cirillo. Accounting for epistemic uncertainty in mission supportability assessment: A necessary step in understanding risk and logistics requirements. 47th International Conference on Environmental Systems, 2017.

[216] Bryan M O'Halloran, Christopher Hoyle, Irem Y Tumer, and Robert B Stone. The early design reliability prediction method. *Research in Engineering Design*, 30(4):489–508, 2019.

[217] Gerhard Pahl and Wolfgang Beitz. *Engineering design: a systematic approach*. Springer Science & Business Media, 2013.

[218] Richard F Paige, Louis M Rose, Xiaocheng Ge, Dimitrios S Kolovos, and Phillip J Brooke. Fptc: automated safety analysis for domain-specific languages. In *International Conference on Model Driven Engineering Languages and Systems*, pages 229–242. Springer, 2008.

[219] The pandas development team. pandas-dev/pandas: Pandas, February 2020.

[220] Yiannis Papadopoulos and John A McDermid. Hierarchically performed hazard origin and propagation studies. In *International Conference on Computer Safety, Reliability, and Security*, pages 139–152. Springer, 1999.

[221] Yiannis Papadopoulos, Martin Walker, David Parker, Erich Rüde, Rainer Hamann, Andreas Uhlig, Uwe Grätz, and Rune Lien. Engineering failure analysis and design optimisation with hip-hops. *Engineering Failure Analysis*, 18(2):590–608, 2011.

[222] Athanasios Papageorgiou, Johan Ölvander, Kristian Amadori, and Christopher Jouannet. Multidisciplinary and multifidelity framework for evaluating system-of-systems capabilities of unmanned aircraft. *Journal of Aircraft*, 57(2):317–332, 2020.

[223] Nikolaos Papakonstantinou, Seppo Sierla, David C Jensen, and Irem Y Tumer. 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*, pages 1045–1054. American Society of Mechanical Engineers, 2011.

[224] Jeryang Park, Thomas P Seager, Palakurth Suresh Chandra Rao, Matteo Convertino, and Igor Linkov. Integrating risk and resilience approaches to catastrophe management in engineering systems. *Risk Analysis*, 33(3):356–367, 2013.

[225] Elisabeth Paté-Cornell. On "black swans" and "perfect storms": risk analysis and management when statistics are not enough. *Risk Analysis: An International Journal*, 32(11):1823–1833, 2012.

[226] Edoardo Patelli and Matteo Broggi. Uncertainty management and resilient design of safety critical systems. In *NAFEMS World Congress 2015*, 06 2015.

[227] Edoardo Patelli, Silvia Tolo, Hindolo George-Williams, Jonathan Sadeghi, Roberto Rocchetta, Marco de Angelis, and Matteo Broggi. Opencossan 2.0: an efficient computational toolbox for risk, reliability and resilience analysis. In *Proceedings of the joint ICVRAM ISUMA UNCERTAINTIES conference*, volume 2018, 2018.

[228] Charles Perrings. Resilience and sustainable development. *Environment and Development Economics*, 11(4):417–427, 2006.

[229] Joseph R Piacenza, Kenneth John Faller, Mir Abbas Bozorgirad, Eduardo Cotilla-Sanchez, Christopher Hoyle, and Irem Y Tumer. Understanding the impact of decision making on robustness during complex system design: More resilient power systems. *ASCE-ASME J Risk and Uncert in Engrg Sys Part B Mech Engrg*, 6(2), 2020.

[230] Stuart L. Pimm. The complexity and stability of ecosystems. *Nature*, 307(5949):321, January 1984.

[231] P Piperni, A DeBlois, and R Henderson. Development of a multilevel multidisciplinary-optimization capability for an industrial environment. *AIAA journal*, 51(10):2335–2352, 2013.

[232] Scott Poll, Johan de Kleer, R Abreau, Matthew Daigle, Alexander Feldman, David Garcia, and A Sweet. Third international diagnostics competition–dxc'11. In *Proc. of the 22nd international workshop on principles of diagnosis*, pages 267–278, 2011.

[233] Scott Poll, Johan de Kleer, Alexander Feldman, David Garcia, Tolga Kurtoglu, and Sriram Narasimhan. Second international diagnostics competition–dxc'10. In *Proceedings of the 21st International Workshop on Principles of Diagnosis*, 2010.

[234] Serhiy Y Ponomarov and Mary C Holcomb. Understanding the concept of supply chain resilience. *The international journal of logistics management*, 20(1):124–143, 2009.

[235] Pavel Popela, Jan Novotnỳ, Jan Roupec, Dušan Hrabec, and Asmund Olstad. Two-stage stochastic programming for engineering problems. *Eng. Mech*, 21(5):335–353, 2014.

[236] Frank A Prince. Why nasa's management doesn't believe the cost estimate. *Engineering Management Journal*, 14(1):7–12, 2002.

[237] Giuliano Punzo, Anurag Tewari, Eugene Butans, Massimiliano Vasile, Alan Purvis, Martin Mayfield, and Liz Varga. Engineering Resilient Complex Systems: The Necessary Shift Toward Complexity Science. *IEEE Systems Journal*, pages 1–10, 2020.

[238] Birgitte Rasmussen, Kristian Borch, and Katharina DC Stärk. Functional modelling as basis for studying individual and organisational factors–application to risk analysis of salmonella in pork. *Food Control*, 12(3):157–164, 2001.

[239] Birgitte Rasmussen and Cris Whetton. Hazard identification based on plant functional modelling. *Reliability Engineering & System Safety*, 55(2):77–84, 1997.

[240] Randal T Rausch, Kai F Goebel, Neil H Eklund, and Brent J Brunell. Integrated in-flight fault detection and accommodation: A model-based study. *Journal of Engineering for Gas Turbines and Power*, 129(4):962–969, 2007.

[241] Antoine Rauzy. Notes on computational uncertainties in probabilistic risk/safety assessment. *Entropy*, 20(3):162, 2018.

[242] Tzvi Raz, Aaron J Shenhar, and Dov Dvir. Risk management, project success, and technological uncertainty. *R&D Management*, 32(2):101–109, 2002.

[243] Sean Rismiller, Jonathan Cagan, and Christopher McComb. Stochastic stackelberg games for agend-driven robust design. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers, 2020. DETC2020-22153.

[244] E. Roghanian, S.J Sadjadi, and M.B Aryanezhad. A probabilistic bi-level linear multi-objective programming problem to supply chain planning. *Appl. Math. Comput*, 188(1):786–800, May 2007.

[245] Adam Rose. Economic resilience to natural and man-made disasters: Multidisciplinary origins and contextual dimensions. *Environmental Hazards*, 7(4):383–398, 2007.

[246] Horst Schirmeier, Martin Hoffmann, Christian Dietrich, Michael Lenz, Daniel Lohmann, and Olaf Spinczyk. Fail*: An open and versatile fault-injection framework for the assessment of software-implemented hardware fault tolerance. In *2015 11th European Dependable Computing Conference (EDCC)*, pages 245–255. IEEE, 2015.

[247] Nico Schlömer. quadpy: Numerical integration (quadrature, cubature) in Python, December 2019. from https://doi.org/10.5281/zenodo.1173132.

[248] Jeffrey Schlosser and Christiaan JJ Paredis. Managing multiple sources of epistemic uncertainty in engineering decision making. Technical report, SAE Technical Paper, 2007.

[249] Nico Schlömer, Nick R. Papior, Matthieu Ancellin, and Darius Arnold. nschloe/quadpy v0.14.7, April 2020.

[250] David A Seaver, Detlof Von Winterfeldt, and Ward Edwards. Eliciting subjective probability distributions on continuous variables. *Organizational Behavior and Human Performance*, 21(3):379–391, 1978.

[251] Carolyn C Seepersad, Kjartan Pedersen, Jan Emblemsvåg, Reid Bailey, Janet K Allen, and Farrokh Mistree. The validation square: how does one verify and validate a design method. *Decision making in engineering design*, pages 303–314, 2006.

[252] Jayashree Shankar, Raphael T Haftka, and Layne T Watson. Computational study of a nonhierarchical decomposition algorithm. *Computational Optimization and Applications*, 2(3):273–293, 1993.

[253] Alexander Shapiro and Andy Philpott. A tutorial on stochastic programming. *Manuscript. Available at www2. isye. gatech. edu/ashapiro/publications. html*, 17, 2007.

[254] Tali Sharot. The optimism bias. *Current biology*, 21(23):R941–R945, 2011.

[255] Lijuan Shen, Beatrice Cassottana, Hans Rudolf Heinimann, and Loon Ching Tang. Large-scale systems resilience: A survey and unifying framework. *Quality and Reliability Engineering International*, 36(4):1386–1401, 2020.

[256] Ada-Rhodes Short, Ann D Lai, and Douglas L Van Bossuyt. Conceptual design of sacrificial sub-systems: failure flow decision functions. *Research in Engineering Design*, pages 1–16, 2017.

[257] AR Short. *Design of autonomous systems for survivability through conceptual object-based risk analysis*. PhD thesis, Colorado School of Mines. Arthur Lakes Library, 2016.

[258] Seppo Sierla, Irem Tumer, Nikolaos Papakonstantinou, Kari Koskinen, and David Jensen. Early integration of safety to the mechatronic system design process by the functional failure identification and propagation framework. *Mechatronics*, 22(2):137–151, 2012.

[259] Timothé M Sissoko, Marija Jankovic, Christiaan JJ Paredis, and Eric Landel. An empirical study of a decision-making process supported by simulation in the automotive industry. In *ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers Digital Collection, 2018.

[260] Colin Small, Gregory Parnell, Edward Pohl, Simon R Goerger, Bobby Cottam, Eric Specking, and Zephan Wade. Engineered resilient systems with value focused thinking. In *INCOSE international symposium*, volume 27, pages 1371–1385. Wiley Online Library, 2017.

[261] Jaroslaw Sobieszczanski-Sobieski. Optimization by decomposition: a step from hierarchic to non-hierarchic systems. 1988.

[262] Alvise Sommariva. Fast construction of fejér and clenshaw–curtis rules for general weight functions. *Computers & Mathematics with Applications*, 65(4):682–693, 2013.

[263] Josh Sorenson. Quadcopter flying on the sky, Apr 2018. Available under CC0 (Public Domain).

[264] Nicolás F Soria Zurita, Robert B Stone, H Onan Demirel, and Irem Y Tumer. Identification of human–system interaction errors during early design stages using a functional basis framework. *ASCE-ASME J Risk and Uncert in Engrg Sys Part B Mech Engrg*, 6(1), 2020.

[265] Eric Specking, Bobby Cottam, Gregory Parnell, Edward Pohl, Matthew Cilli, Randy Buchanan, Zephan Wade, and Colin Small. Assessing engineering resilience for systems with multiple performance measures. *Risk Analysis*, 39(9):1899–1912, 2019.

[266] Prasanna Sridharan and Matthew I Campbell. A grammar for function structures. In *ASME 2004 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 41–55. American Society of Mechanical Engineers, 2004.

[267] Michael Stamatelatos, William Vesely, Joanne Dugan, Joseph Fragola, Joseph Minarick, and Jan Railsback. Fault tree handbook with aerospace applications. Technical report, NASA, 2002.

[268] Danielle Stewart, Jing Janet Liu, Michael W Whalen, Darren Cofer, and Michael Peterson. Safety annex for the architecture analysis and design language. 2018.

[269] Robert B. Stone, Irem Y. Tumer, and Michael Van Wie. The Function-Failure Design Method. *Journal of Mechanical Design*, 127(3):397–407, July 2004.

[270] Robert B Stone and Kristin L Wood. Development of a functional basis for design. *Journal of Mechanical design*, 122(4):359–370, 2000.

[271] Thomas Stone, Seung-Kyum Choi, and Hemanth Amarchinta. Structural model refinement under uncertainty using decision-maker preferences. *Journal of Engineering Design*, 24(9):640–661, 2013.

[272] Shun Takai and Kosuke Ishii. The maximum resource allocation for uncertainty reduction in a decision-analytic concept selection. *Concurrent Engineering*, 18(1):19–29, 2010.

[273] James JY Tan, Kevin N Otto, and Kristin L Wood. Relative impact of early versus late design decisions in systems development. *Design Science*, 3, 2017.

[274] Nathan Tedford and JRRA Martins. *Comparison of MDO architectures within a universal framework*. Citeseer, 2007.

[275] Ben H Thacker, Scott W Doebling, Francois M Hemez, Mark C Anderson, Jason E Pepin, and Edward A Rodriguez. Concepts of model verification and validation. Technical report, Los Alamos National Lab., 2004.

[276] Stephanie C Thompson and Christiaan JJ Paredis. A process-centric problem formulation for decision-based design. In *ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 753–762. American Society of Mechanical Engineers, 2009.

[277] Stephanie C Thompson and Christiaan JJ Paredis. An investigation into the decision analysis of design process decisions. *Journal of Mechanical Design*, 132(12):121009, 2010.

[278] Deborah L Thurston. Utility function fundamentals. In *Decision making in engineering design*. ASME Press, 2006.

[279] Lloyd N Trefethen. Is gauss quadrature better than clenshaw–curtis? *SIAM review*, 50(1):67–87, 2008.

[280] David Ullman. *The mechanical design process*. McGraw-Hill Science/Engineering/Math, 2009.

[281] Steven Ulrich, Karl Tand Eppinger. *Product design and development*. McGraw-Hill Education, 2012.

[282] Franciscus LJ van der Linden. General fault triggering architecture to trigger model faults in modelica using a standardized blockset. In *Proceedings of the 10th International Modelica Conference-Lund, Sweden-Mar 10-12, 2014*, number 96, pages 427–436. LiU Electronic Press, 2014.

[283] William Vesely, Joseph Fragola, Joseph Minarick, and JAn Railsback. Fault Tree Handbook With Aerospace Applications. Technical report, NASA Office of Safety and Mission Assurance, Washington, DC, August 2002.

[284] Matheus Palhares Viana, Esther Tanck, Marcelo Emilio Beletti, and Luciano da Fontoura Costa. Modularity and robustness of bone networks. *Molecular BioSystems*, 5(3):255–261, 2009.

[285] John Von Neumann and Oskar Morgenstern. *Theory of games and economic behavior*. Princeton university press, 2007.

[286] Heinrich Von Stackelberg. *Market structure and equilibrium*. Springer Science & Business Media, 1934.

[287] Yatin Wadhawan and Clifford Neuman. Bags: A tool to quantify smart grid resilience. In *FedCSIS Communication Papers*, pages 323–332, 2017.

[288] Eric-Jan Wagenmakers, Michael Lee, Tom Lodewyckx, and Geoffrey J Iverson. Bayesian versus frequentist inference. In *Bayesian evaluation of informative hypotheses*, pages 181–207. Springer, 2008.

[289] Jörg Waldvogel. Fast construction of the fejér and clenshaw–curtis quadrature rules. *BIT Numerical Mathematics*, 46(1):195–202, 2006.

[290] Hannah S. Walsh, Andy Dong, and Irem Y. Tumer. The role of bridging nodes in behavioral network models of complex engineered systems. *Design Science*, 4(e8), 2018.

[291] Hannah S. Walsh, Andy Dong, and Irem Y. Tumer. An analysis of modularity as a design rule using network theory. *Journal of Mechanical Design*, 141(3):031102, 2019.

[292] Hannah S Walsh, Andy Dong, and Irem Y Tumer. An analysis of modularity as a design rule using network theory. *Journal of Mechanical Design*, 141(3), 2019.

[293] Pingfeng Wang, Byeng D Youn, Chao Hu, Jong Moon Ha, and Byungchul Jeon. A probabilistic detectability-based sensor network design method for system health monitoring and prognostics. *Journal of Intelligent Material Systems and Structures*, 26(9):1079–1090, 2015.

[294] Yan Wang. On rational decision with subjective probability in design. 2012.

[295] Zili Wang, Yiqian Cui, and Junyou Shi. A framework of discrete-event simulation modeling for prognostics and health management (phm) in airline industry. *IEEE Systems Journal*, 11(4):2227–2238, 2015.

[296] Stefan Winter, Thorsten Piper, Oliver Schwahn, Roberto Natella, Neeraj Suri, and Domenico Cotroneo. Grinder: On reusability of fault injection tools. In *2015 IEEE/ACM 10th International Workshop on Automation of Software Test*, pages 75–79. IEEE, 2015.

[297] Kristen L Wood, RJ B Stone, DRJ Mcadams, J Hirtz, and Simon Szykman. A functional basis for engineering design: Reconciling and evolving previous efforts. Technical report, National Institute of Standards and Technology, 2002.

[298] William H Wood and Alice M Agogino. Decision-based conceptual design: modeling and navigating heterogeneous design spaces. *Journal of Mechanical Design*, 127(1):2–11, 2005.

[299] M Keith Wright, Lynne Stokes, and James S Dyer. Reliability and coherence of causal, diagnostic, and joint subjective probabilities. *Decision Sciences*, 25(5-6):691–709, 1994.

[300] Jiaxin Wu and Pingfeng Wang. Risk-averse optimization for resilience enhancement of complex engineering systems under uncertainties. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers, 2020. DETC2020-22226.

[301] Jun Yang and Tunc Aldemir. An algorithm for the computationally efficient deductive implementation of the markov/cell-to-cell-mapping technique for risk significant scenario identification. *Reliability Engineering & System Safety*, 145:1–8, 2016.

[302] Ilan Yaniv and Dean P Foster. Precision and accuracy of judgmental estimation. *Journal of behavioral decision making*, 10(1):21–32, 1997.

[303] Anil Yildiz, M Ugur Akcal, Batuhan Hostas, and N Kemal Ure. Switching control architecture with parametric optimization for aircraft upset recovery. *Journal of Guidance, Control, and Dynamics*, 42(9):2055–2068, 2019.

[304] Nita Yodo and Pingfeng Wang. Engineering Resilience Quantification and System Design Implications: A Literature Survey. *Journal of Mechanical Design*, 138(11):111408–111408–13, September 2016.

[305] Nita Yodo and Pingfeng Wang. Resilience allocation for early stage design of complex engineered systems. *Journal of Mechanical Design*, 138(9), 2016.

[306] Byeng D Youn, Chao Hu, and Pingfeng Wang. Resilience-driven system design of complex engineered systems. *Journal of Mechanical Design*, 133(10), 2011.

[307] Bo Yang Yu, Tomonori Honda, Syed Zubair, Mostafa H Sharqawy, and Maria C Yang. A framework for system design optimization based on maintenance scheduling with prognostics and health management. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 55881, page V03AT03A035. American Society of Mechanical Engineers, 2013.

[308] Xiaoge Zhang, Zhen Hu, and Sankaran Mahadevan. Bilevel optimization model for resilient configuration of logistics service centers. *IEEE Transactions on Reliability*, 2020.

[309] Enrico Zio. Integrated deterministic and probabilistic safety assessment: concepts, challenges, research directions. *Nuclear Engineering and Design*, 280:413–419, 2014.

[310] Enrico Zio. The future of risk assessment. *Reliability Engineering & System Safety*, 177:176–190, 2018.