AN ABSTRACT OF THE DISSERTATION OF

Hannah S. Walsh for the degree of Doctor of Philosophy in Mechanical Engineering presented on October 30, 2020.

Title: The Structural Characteristics of Robustness in Large-Scale Complex Engineered Systems

Abstract approved: _____

Irem Y. Tumer, Andy Dong

Modern engineered systems are increasingly integrated into society, decentralized, and interconnected. However, scale and complexity create unique challenges in system design, such as emergent behavior and unintended consequences. This work investigates the structural characteristics of robustness in complex engineered systems by identifying patterns of under-recognized vulnerabilities at various levels of hierarchy. The work begins at the highest level of hierarchy, taking a system of systems view in order to investigate unintended consequences. Then, this work investigates the system architectures that make them prone to certain types of failures using different network modeling perspectives. Finally, this work uses network modeling to identify system elements that are key to controlling system robustness from a system behavior perspective. Altogether, this research provides a novel perspective on robustness of complex systems and presents empirical findings on general patterns of system vulnerabilities that can be used in future system-level architecture design. Applications of the work are focused on aerospace systems but can be applied in a diverse range of domains, including energy and transportation.

The Structural Characteristics of Robustness in Large-Scale Complex Engineered Systems

by

Hannah S. Walsh

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented October 30, 2020
Commencement June 2021

Doctor of Philosophy dissertation of Hannah S. Walsh presented on October 30, 2020.

APPROVED:

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.

Hannah S. Walsh, Author

ACKNOWLEDGEMENTS

I would like to begin by thanking my advisors Dr. Irem Tumer and Dr. Andy Dong. I am truly lucky to have had the opportunity to work with and learn from you both. Irem, your mentorship has been pivotal in my personal and professional development over the past four years. Andy, I am so grateful for your patience and endless supply of ideas and inspiration. I appreciate the time and input of my committee members Dr. Chris Hoyle, Dr. Bryony DuPont, and Dr. Kathy Mullet, who have all been instrumental in making my dissertation the best it can be. I would like to thank the Robust Software Engineering group at NASA Ames Research Center, especially Dr. Guillaume Brat. Guillaume convinced me to go to graduate school in the first place and correctly predicted that I would indeed not only get a Masters, as I had originally planned, but also a PhD. I would like to thank the students of the Design Engineering Lab for helping to create the wonderful environment that we have for learning and growth. In particular I would like to thank recent graduates of our program Dr. Pegah Keshavarzi and Dr. Nico Soria, who have become dear friends and role models for me as researchers and as human beings. I would also like to thank my personal friends for their welcome distraction from both the routine and the existential stresses of graduate school. Of course, none of this would have been possible without the lifelong support of my parents. And to my amazing husband, thank you for reminding me of what is important in life and for always believing in me.

CONTRIBUTION OF AUTHORS

All research conducted in completing this dissertation was completed with the guidance of Irem Tumer and Andy Dong. Additionally, in Chapter 3, Guillaume Brat provided guidance on machine learning approaches, and Andy Dong participated in the validation of the proposed human encoding scheme. In Section 5.3, Mohammad Hejase and Daniel Hulse contributed to the development of the Structural Consequence Analysis approach. Mohammad Hejase additionally produced several of the figures in Section 5.3. Co-authors of each publication are acknowledged in the corresponding chapters.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

Page

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

Figure Page

LIST OF FIGURES (Continued)

Figure Page

LIST OF TABLES

LIST OF TABLES (Continued)

To my grandfather, Bill Walsh, who inspired me to fly high.

Chapter 1: Introduction

Understanding how systems respond to internal and external perturbations is a key part of designing systems that operate safely and perform adequately. This idea is the driver behind robustness analysis. Robustness in the context of this research is a property of systems that tend to retain acceptable, if reduced, performance even when faced with internal faults, external perturbations, and unexpected interactions with other systems. Many methods exist for improving system robustness, perhaps the most salient of which is the Taguchi method [2]. Such methods tend to focus on finding the optimal parameter settings given knowledge of noise factors such as manufacturing variability and variable environmental conditions. These approaches have enjoyed much success in improving product quality.

Many of these methods, however, break down when confronted with complexity. Complexity stems from structural or dynamic characteristics [3] or from socio-technical integration [4, 5]. Complexity presents unique design challenges. Specifically, complex systems may exhibit nonlinear behavior and emergent characteristics and may be difficult to understand, with unclear cause and effect. Additionally, coupling in complex systems may also lead to small perturbations causing large, system level effects, known as the butterfly effect [6]. These characteristics lead to challenges in accurately predicting behavior in such a way as is necessary for robustness analysis methods such as the Taguchi method. Beyond behavior specification, there are logistical challenges in designing such systems. Large-scale complex systems require the cooperation of a large number of professionals, complicating the systems engineering and design processes. Such challenges are evident in the budget overruns and delays that are notorious in large scale complex engineering projects [7]. For instance, data from 2018 indicates the National Aeronautics and Space Administration (NASA) operating at a cost growth of 27.6% over its baseline, with a launch delay averaging approximately thirteen months [8]. Even more

drastic results have been reported in large-scale construction projects [9]. Worse, many large-scale engineering projects fail and are never completed [10]. Delayed and failed projects can result in enormous costs to organizations and to the public.

In response, increasingly sophisticated approaches to systems engineering and, in particular, complexity management have been implemented. There are two common approaches: modularity, which involves dividing a system into more manageable parts [11], and abstraction, which involves representing only the essential elements of the system for analysis [10]. Such approaches have been implemented successfully in many complex engineering projects and are intrinsically tied to the standard systems engineering process. The systems engineering process utilizes both approaches as it progresses from models and specification of the system at a high level of abstraction to lower-level, more detailed models and specification of subsystems. NASA's Team X is another example of a modularized (by disciplinary expertise) design process [12]. More recently, researchers have introduced novel approaches to complex engineered system design that are built on these ideas [13, 14, 15]. Both modularity and abstraction make possible the otherwise intractable problem of analyzing large-scale complex systems, and are generally useful approaches.

However, analyses that utilize such approaches are prone to overlooking or oversimplifying two important system properties: emergent behavior and interconnectedness. Emergent behaviors occur when the behavior of a system is indiscernible from the sum of its parts, and are difficult to discern taking a reductionist approach to system analysis [16]. For example, a component with many connections in the completed system (a "hub", from a network perspective) may become overloaded and fail or perform poorly. In other words, the parts of a system alone may be well understood, but their integrated behavior may be more difficult to predict. Thus, an integrated (rather than modular) approach is needed in order to understand the behavior of the system as a whole. Because of the importance of interactions between system parts, and between the system and its environment, it is desirable to study complex systems holistically and in context [17]. Addi-

tionally, abstractions can inhibit analysts' understanding of the interconnections within a system, unintentionally abstracting away seemingly minor interactions which, under certain conditions, may produce important behaviors. Neglecting these essential properties of complex engineered systems can lead to unanticipated failure modes and unintended consequences, or side effects of actions taken during the design process. These effects reduce the robustness of the designed system – that is, the system may not operate as expected under the influence of internal faults and external perturbations. With full specification being infeasible for complex systems, however, there are currently few options available for avoiding these pitfalls.

This research addresses these vulnerabilities that are under-recognized in conventional robustness analysis in three stages. Specifically, this research investigates three types of under-recognized vulnerabilities at three different levels of hierarchy in systems: unintended consequences at the system-of-systems level, architectures that help or hinder robustness at the system level, and system elements that are key to controlling robustness at the module level. Each of these three under-recognized vulnerabilities is addressed in Chapters 3–5. There are two main approaches leveraged: a machine learning approach, in Chapter 3, and a network theoretic approach, in Chapters 4–5.

Chapter 3 leverages machine learning in order to detect leading design stage indicators and archetypes of unintended consequences from a large data set of adverse events. Unintended consequences in this context can be described as unexpected behaviors occurring as a result of the interaction between systems that can be traced to a specific design decision. Particularly in highly complex, large scale systems that interact heavily with other systems and their environment, model abstraction is inevitable. All models are by definition abstractions of reality, but for complex systems, models are likely to be more abstracted than for simpler, smaller systems. By leveraging machine learning to link leading design indicators to unintended consequences, the system is effectively treated as a black box, enabling prediction of likely adverse outcomes without the drawbacks of abstraction.

Chapters 4–5 use a type of abstracted white box model – a network model. Network-based modeling and analysis of complex engineered systems enables analysis of emergent behavior that can be difficult in conventional models. Using network models, the topological or structural features of complex engineered systems are explored at two levels of hierarchy in Chapters 4 and 5. In Chapter 4, the ability of a system's architecture, in particular its modularity, to help or hinder robustness is investigated. In Chapter 5, variables with specific topological roles, specifically nodes that connect modules, are identified as key to controlling a system's robustness.

1.1 Objective and Research Questions

The overall objective of this work is to understand the robustness of large-scale complex engineered systems to failure in a way that focuses on capturing underrecognized vulnerabilities. This research identifies general patterns of vulnerability and robustness in complex engineered systems at three levels of hierarchy: the system-of-systems level, systems level, and module (subsystem) level. Each hierarchical level is addressed as one of three primary research questions in Chapters 3–5:

1. *Which sets of system characteristics tend to produce failures that can be described as unintended consequences?* At the system-of-systems level, unintended consequences occur when a system or subsystem has side effects on another system or subsystem. By clarifying the theory behind unintended consequences and by applying machine learning techniques, it is possible to identify design-stage leading indicators of the occurrence of unintended consequences. These indicators, or risk factors, can be used to predict likely archetypes of unintended consequences, or specific behavioral patterns that produce adverse outcomes.

2. *Which system topologies are most vulnerable to failure?* System level topology, or architecture, has significant bearing on multiple system characteris-

tics, such as maintainability and robustness. One descriptor of architecture is modularity, or the property of systems that have tightly coupled modules that are more loosely connected to one another. Modularity has many benefits in design, but some evidence has suggested that it may reduce robustness. It is possible to use network modeling and analysis to study the relationship between modularity and robustness in complex engineered systems, thereby gaining an understanding of system architectures that are resistant to faults. This information can guide decision-making regarding modularization and protection measures for intermodule connections.

3. *What is the topological role of the variables that are key to controlling a system's robustness to failure?* Approaches such as sensitivity analysis are often used to identify parameters that increase the fragility of systems. Such approaches, however, rely on detailed and accurate system models which, in complex systems, may neglect emergent properties and key interfaces between systems. Finding vulnerable system elements is important to planning redundancy and health management systems. By representing systems as networks, it is possible to use network metrics to identify vulnerabilities in complex systems in such a way as to capture such emergent properties.

1.2 Contributions

The contributions of the research are summarized in this section. The empirical findings of this work, relating to Chapters 3–5, respectively, are listed below.

1. *New archetypes of unintended consequences are identified from historical data.* These archetypes are more granular and rare than existing archetypes. This result provides descriptive models of ways in which unintended consequences can occur, even for relatively rare events, and enables the prediction of undesirable outcomes using risk factors.

2. *More modular complex engineered systems are shown to be less robust to*

random failure, from architectural, functional, and behavioral perspectives. This result identifies the trade-off between the benefits of modularity and its potential risks.

3. *Bridging nodes are identified as vulnerable nodes from the perspective of system robustness.* This result enables the *a priori* prediction of system parameters that contribute to fragility, while accounting for system-level, emergent behaviors.

Second, conceptual and methodological advances, relating to Chapters 3–5, respectively, are listed below. These contributions are both necessary for the empirical findings and useful in their own right in enabling early design assessment of robustness of complex systems.

1. *A conceptual foundation is proposed for describing unintended consequences in the context of engineering design.* In Section 3.3, a conceptual framework for unintended consequences is proposed, laying the groundwork for studying unintended consequences in the design of complex engineered systems.

2. *A methodology is proposed for modeling components, functions, and parameters in various combinations as bipartite networks.* In Chapter 4, these models are utilized to study modularity and robustness from different modeling perspectives.

3. *A methodology is proposed for modeling logical and embedded behavior in network-based behavioral models of complex engineered systems.* In Chapter 5, existing network modeling methods [18] are extended to new types of system behaviors.

1.3 Limitations

This research is intended as a complement rather than a replacement to conventional failure analysis. This research provides a different perspective that is

useful for designing complex systems with unexpected behaviors that are difficult to capture using conventional methods. However, this approach does not provide quantitative assessment of risk, in terms of probability of failure, as in other methods. This approach enhances risk assessment of complex systems by capturing under-recognized vulnerabilities, and can be used earlier in the design process than existing approaches – although it should be followed by conventional approaches.

1.4 Definitions

For ease of reference, technical definitions used in this dissertation are summarized in Table 1.1. Importantly, many of these definitions have specialized meanings in the context of this research. Many are detailed in later chapters.

1.5 Chapter Summary

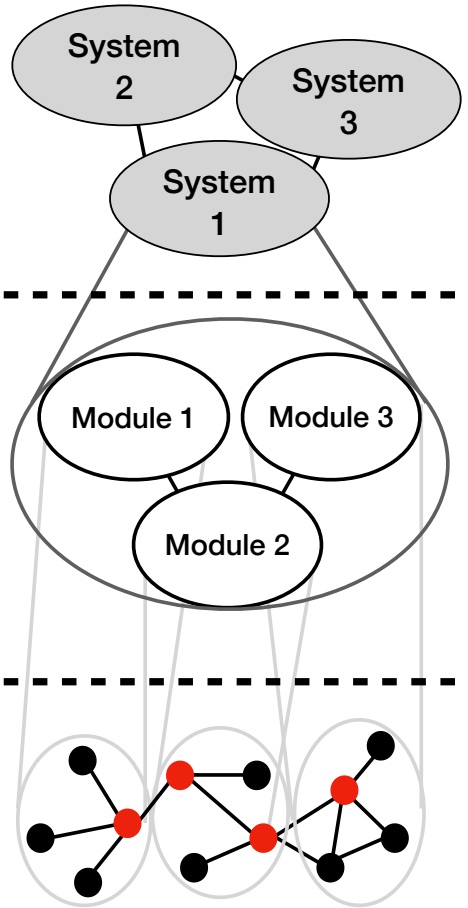
The objective of this work is to understand the robustness of large-scale complex engineered systems in a way that focuses on capturing under-recognized vulnerabilities. Robustness in this sense refers to systems that demonstrate acceptable performance even under the influence of internal or external faults or perturbations. The research questions investigate the robustness of complex engineered systems at three levels of hierarchy: the system-of-systems level, the system level, and the subsystem level. Specifically, at the system-of-systems level, this research investigates the unintended consequences of decisions made in one system on another system. At the system level, this research investigates which system architectures help or hinder robustness. Finally, at the subsystem or module level, this research investigates the topological role of variables that increase fragility. The objective, research questions, and contributions of this research are summarized in Fig. 1.1. Chapters 3–5 each address one of the research questions of this work, investigating robustness at progressively lower levels of hierarchy.

Table 1.1: Key definitions used in this research.

Terminology	Definition
Complexity	A property of systems that cannot be fully understood by a single human observer; typically characterized as highly interconnected and interdependent with nonlinear and emergent behavior, multiple levels of hierarchy, and adaptation
Emergence	A phenomenon in which a number of elements interact in such a way as to produce a higher-level behavior that is non-evident from the sum of the individual elements
Failure	A loss of system-level performance
Fault	An off-nominal condition in a system element
Fragility	The opposite of robustness; a property of systems whose performance is easily upset by small perturbations or faults
Risk	An adverse outcome, its likelihood of occurring, and its consequence
Risk Factors	Characteristics that can be identified during the design phase that are associated with certain risks; effectively, design stage leading indicators of certain types of failures
Robustness	The ability of a system to retain acceptable performance even when faced with faults, perturbations, and/or unintended consequences
System	A collection of interrelated and interdependent elements forming a whole
Under-Recognized Vulnerabilities	Sources of fragility in complex systems that are difficult to capture in conventional risk assessment; examples include topological characteristics and unintended consequences
Unintended Consequence	A side effect of one system on another system

Research Framework for Understanding the Robustness of Complex Engineered Systems

The objective of this work is to understand the robustness of large-scale complex engineered systems in a way that focuses on capturing under-recognized vulnerabilities.



Research Questions	Contributions
Chapter 3: Identification of Unintended Consequences	
Which sets of system characteristics tend to produce failures that can be described as unintended consequences?	***Empirical*** • Sixty-six archetypes of unintended consequences are identified from historical data.
	Conceptual • A conceptual foundation is proposed for describing unintended consequences in the context of engineering design.
Chapter 4: Identification of Robust System Architectures	
Which system topologies are most vulnerable to failure?	***Empirical*** • More modular systems are found to be less robust to random failure from architectural, functional, and behavioral perspectives.
	Methodological • A methodology is proposed for modeling components, functions, and parameters in various combinations as bipartite networks.
Chapter 5: Identification of Vulnerable System Elements	
What is the topological role of the variables that are key to controlling a system's robustness to failure?	***Empirical*** • Bridging nodes are identified as vulnerable nodes from the perspective of system robustness.
	Methodological • Network-based behavioral modeling of complex engineered systems are extended to logical and embedded behavior.

Figure 1.1: Overall framework for addressing the main research objective. The research questions assess robustness at three levels of hierarchy: system-of-systems, system, and module.

Chapter 2: Background

Robustness refers to a system property that relates to the system's tolerance to faults, perturbations, and/or unintended consequences. A fault refers to an off-nominal condition in a component. Effectively, if there is a fault in a component, the performance of robust systems will change less significantly than the performance of less robust systems. Chapter 3 begins with a slightly broader view of this understanding and considers how *unanticipated external perturbations or interactions* may affect system performance. Chapters 4–5 investigate how faults injected in components with varying structural roles affects system performance, enabling an understanding of the system's robustness to different topological characteristics.

Robustness is related to risk and failure analysis more generally. This chapter will provide an overview of the state-of-the-art in failure analysis techniques as well as describe the particular challenges associated with complex engineered systems. Many complex engineered systems, such as aerospace systems, require sophisticated failure analysis techniques for certification and to prevent adverse events that could jeopardize mission success or result in loss of property or life. Failure, in the context of this research and as summarized in Table 1.1, refers to a loss of performance at the system-level, and could have any of these consequences. Many approaches to failure analysis emphasize the estimation of failure probabilities of components and their downstream effects. These methods can be dynamic, meaning based on the system's behavior over time, or static, meaning based on the connectivity between system elements. System reliability is measured through, for example, Mean Time Between Failures (MTBF). Other approaches consider various undesirable system states and possible scenarios that could result in these states. This research differs from existing approaches in that, rather than focusing on quantifying risk, this research investigates various structural characteristics at different hierarchical levels that contribute to a system's robustness.

2.1 Failure Analysis and Prevention

Failure analysis and prevention are incredibly important aspects of the systems engineering process, particularly for safety critical systems. In certifying aerospace systems, typically a safety case is constructed. A safety case is used to certify that a system is safe to operate within a certain set of bounds. Safety cases are typically supplied to an appropriate regulatory agency for approval [19]. Generally, a safety case is produced towards the end of the system development process [19]. However, if safety issues are identified late in the system development process, it is often expensive to rework the system. Because subsystems are often designed by separate design teams or companies, vulnerabilities are not known until the full system is integrated and tested, leading to costly and time consuming design changes. For this reason, it is desirable to identify safety issues early in the design process and account for mitigation strategies such as redundancy earlier. While conventional methods for failure analysis are typically carried out late in the design process, there are developing methods available for assessing failures earlier in the design process. However, there are particular challenges associated with large-scale complex systems which make early design failure analysis particularly difficult. This section highlights particular shortcomings and challenges of existing failure analysis and prevention approaches which lead to under-recognized vulnerabilities being overlooked during system design.

2.1.1 Late Design Stage Failure Analysis

A well known failure analysis approach is Failure Modes and Effects Analysis (FMEA) [20]. FMEA is used to identify failure modes in a systematic manner. In FMEA, component failure modes and their effects are listed, generally using expert analysis [21]. Contributing factors to each risk are also usually assessed. The consequences, or effects, of each failure mode area also considered. Some methods also consider the severity of the effects of the failure mode, usually using a numeric rating indicating whether the effect is negligible (a lower number), catastrophic (a

higher number), or somewhere in between. The probability of the failure mode is also assessed, again using a numeric rating. Finally, the detectability of the failure mode is also considered and given a numeric rating. This rating refers to the difficulty in detecting the failure mode during operation, which may prevent mitigation strategies from being taken in time to prevent or lessen the consequences of failure. The probability, severity, and detectability ratings can be multiplied together to attain a Risk Priority Number (RPN). Failure modes with higher RPNs are considered more critical. Based on this information, designers can prioritize developing appropriate countermeasures for failure modes with high RPNs.

Fault Tree Analysis (FTA) utilizes a state-based tree structure model, which shows paths leading to an undesirable system state, or top event [22]. Fault trees are constructed using expert analysis and historical data. Combinations of events that cause the top event are called minimal cut sets. These cut sets can be used to compute the top event probability, which is compared to risk requirements to verify that a design meets risk requirements. Redundancy is one method of decreasing the top event probability. Redundancy is typically added to components of high importance within the fault tree. There are multiple metrics available for determining a component's importance. Common ones are statistical importance, or Fussell-Vesely importance, and Birnbaum importance. The statistical importance of a component measures the contribution of cut sets that contain the component to the top event probability. The equation for the statistical importance of a component is given in Eq. 2.1, where F is the Fussell-Vesely importance, j is an index for individual cut sets, m is the number of cut sets, R is the reliability function, c is a vector of all cut sets, i is an index for individual components, and c_j^i is a cut set j containing component i.

$$F(i) = \frac{\sum_j^m [1 - R(c_j^i)]}{\sum_j^m [1 - R(c_j)]} \tag{2.1}$$

Birnbaum importance has two forms: reliability importance and structural importance [23, 24]. Reliability importance is based on failure rates of components,

while structural importance assumes equal failure rates for all components. The equation for the reliability importance of a component is given in Eq. 2.2, where $B(i|p)$ is the reliability importance [23] and p is the vector of failure rate information for all components within an architecture. The equation for the structural importance of a component is given in Eq. 2.3, where n is the total number of components within an architecture [23].

$$B(i|p) = \frac{\partial R(p)}{\partial p_i} \tag{2.2}$$

$$B(i) = \left.\frac{\partial R}{\partial p_i}\right|_{p_1=\ldots=p_n=0.5} \tag{2.3}$$

There have been numerous proposed extensions to FTA in the literature. For example, fuzzy set theory has been used to consider imprecision in the probabilities used to calculate the top event probability [25]. Other work has improved the computations of top event probabilities when the number of minimal cut sets is large [26]. There have been numerous advances used to make fault trees consider time-dependent information, called dynamic fault trees [27]. This is typically done using Markov chains [28, 29]. Other extensions include Monte Carlo analysis [30], Petri nets [31], binary decision diagrams [32], dependent events [33], and repairable trees [34]. Similar to FTA, Event Tree Analysis (ETA) uses a tree structure, but it considers events rather than states and begins with a single initiating event rather than a top event [35].

Bow tie diagrams are often used in safety cases to assess the risk of a certain undesirable event. They provide a more comprehensive understanding of hazardous events [36] and provide a useful visualization of safeguards in place to prevent the occurrence of adverse events, enabling practitioners to discover shortcomings in their risk management plans. They can be used retrospectively, after an accident has occurred, as well as proactively, during an initial hazard analysis. Bow tie diagrams provide an overview of possible initiating events, on the left side of the diagram, as well as the consequences of the top event, on the right, with the top

event in the middle. This forms a bow tie shape from which the method gets its name. Safeguards can be visualized on each side of the diagram, depending on whether they are prevention or mitigation strategies, respectively.

Probabilistic Risk Assessment (PRA) is a rigorous approach to enhance safety. Risk, in the context of risk assessment, has three parts: the adverse outcome, its likelihood, and its consequence [37]. In PRA, certain undesirable outcomes are first identified. PRA identifies various paths and contributors to these undesirable outcomes. For this purpose, it generally utilizes special types of fault trees called master logic diagrams and event trees that model accident scenarios. This approach is followed by uncertainty and sensitivity analyses, typically using Monte Carlo simulation. Sensitivity analysis enables an understanding of which parameters are capable of influencing the outputs the more than others. Bayesian statistical methods are often used when data is difficult to obtain [38]. Once PRA is complete, vulnerabilities are often ranked by importance. PRA has been extended to include organizational factors that contribute to accidents [39]. Dynamic PRA, or DPRA, considers both aleatory and epistemic uncertainty whereas PRA primarily considers aleatory uncertainty [40].

Reliability Block Diagram (RBD) is a graphical tool using the system architecture and the failure probabilities of components [41, 42] that aids in the identification of areas that hinder system reliability. Based on the findings of an RBD, decisions can be made to improve reliability of or add redundancy to certain components in order to improve system reliability. In this approach, blocks are arranged in series or in parallel depending on whether components are redundant, enabling the computation of system failure rates. The entire diagram can be simplified into a series architecture using simplification rules. There have been a number of extensions to RBD, including dynamic RBD [43].

2.1.2 Early Design Stage Failure Analysis

While less precise than many of the late design methods, early design methods enable risk assessment to be carried out earlier in the design process. Early in

design, critical design decisions are still relatively flexible, meaning designers have maximum freedom to select a design with improved robustness and reliability. Conversely, precisely because fewer design decisions have been made, detailed models for reliability analysis are unavailable. Since detailed models are unavailable, early stage methods often rely on highly structured processes [21]. Examples include the Taguchi method [2] and Design for Six Sigma [44], which are intended to reduce variance and increase reliability [21]. Other, less established methods are also under development for understanding possible system failure modes and their impact. Other methods take a qualitative approach [21, 45, 46, 47]. For instance, Function Failure Identification and Propagation (FFIP) provides an understanding of failure behavior by identifying fault propagation paths with behavioral simulation. These paths can be related to the system's components and functions [47]. Later extensions to this work include Failure Flow Decision Functions (FFDF), which utilizes FFIP as well as flow state logic to identify highly critical subsystems during design [48]. Other approaches rely heavily on functional models, such as Function Failure Design Method (FFDM). In functional models, failure is understood as loss or degradation of a function rather than failure of a component [49]. Using this definition, FMEA-like failure analysis is able to be performed in early design [49]. An extension of FFDM is Risk in Early Design (RED), which enables further quantification of risk [50]. More recently, researchers have turned to Bayesian Network (BN)-based analysis of system risk and resilience early in design [51, 52] and physics-based approaches [53]. Other authors have considered risk modeling in a collaborative environment [54]. In sum, these methods enable analysis of failure and risk in early design to guide key design decisions and avoid costly redesign later in the design process.

Both early and late stage design methods are useful in assessing risk and failure modes during the design of engineered systems. However, many of these methods struggle to capture under-recognized vulnerabilities in complex systems due to inherent challenges in modeling and analyzing complex systems. This research takes an alternative approach in considering the trends and patterns of vulnerability

across complex systems, enabling *a priori* prediction of such vulnerabilities that can be used in conjunction with conventional approaches to risk assessment. In the next section, a more in-depth review of complex systems will be presented in order to provide the theoretical context for the challenges of complex systems design.

2.2 Complexity in Engineering Design

Complexity is often attributed to systems or scenarios that are challenging to understand, typically with many interconnected parts. The terms "complex" and "complicated" are often used interchangeably in everyday language; however, in the systems sciences, even very complicated systems may not be considered complex. In this research, what differentiates the two is that complex systems are difficult to model and understand thoroughly for a human observer due to intricate coupling, non-linearity, and scale. A common example of a complex system is the National Airspace System (NAS). The NAS includes airplanes, the airspace itself, rules and procedures, personnel, passengers, airports, and other elements. In engineering design and systems engineering, complex engineered systems typically refer to systems that require the integration of multiple disciplinary subsystems in order to perform their intended function.

Complex systems display emergent properties, that is, they have properties that their parts by themselves do not have [55]. These are often considered higher level behaviors or characteristics. For example, the NAS has emergent properties such as hubs and traffic patterns, which are not determinable from looking at the elements of the system alone [56]. The airplanes themselves are capable of performing functions, namely, powered flight, that are impossible for its parts alone. The design process can introduce emergent behaviors due to the interactions between various design team members [57]. Emergence is necessary for the functionality of complex engineered systems; however, unintended emergent behaviors can also cause unexpected failure modes or reduced performance.

Other characteristics that are often attributed to complex systems include autonomous components, self-organization, and adaptation. Self-organization refers

to situations in which order arises from parts that are originally disordered. An adaptive system is capable of changing in response to environmental factors, usually resulting in an increase in complexity [56]. In the NAS, procedures and demand adapt after terrorist attacks, pandemics, and due to market forces. These characteristics of complex systems are not always seen in complex engineered systems themselves, but may become apparent when viewing complex engineered systems as a product of the systems engineering process or as being a part of a system of systems. For example, autonomous system elements exist in the design process of an airplane as well as the maintenance and operation schedule. After 9/11, locking doors were retrofitted into existing aircraft between the cockpit and passenger area – an adaptation to a changing environment.

Some authors measure the complexity of a system according to the amount of information needed in order to describe it [17], often based on axiomatic design [58]. Complexity is also relatable to hierarchy [55], that indicating measuring emergent behavior is another method of quantifying complexity. Other methods emphasize structural complexity [59, 60]. Others treat complexity as a property that is either present or absent, with no "degree" of complexity.

Complex systems are by nature difficult to understand by human designers and therefore create challenges during design. The concept of bounded rationality helps explain why it is so difficult to predict the consequences of actions taken to influence complex systems, and why they are therefore difficult to design. According to bounded rationality, humans make generally rational decisions, but are limited by time and resources available to them [61]. Because of bounded rationality, in practical applications, heuristics are often used in decision-making [62, 63]. However, heuristics inherently do not always produce the optimal solution. Additionally, they may be influenced by cognitive biases [64, 65]. Fundamentally, any decision made by human designers – and even decisions made using models made by humans – will inherently be working with limited information in the context of complex systems. This fundamental assumption about complexity explains the occurrence of the under-recognized vulnerabilities explored in this research.

These complexity-related challenges complicate efforts to perform robustness assessment of engineered systems. Increasingly complex and evolving systems and systems-of-systems that are also safety critical, such as advanced air mobility, necessitate such methods. The means and methods proposed in this research are designed to handle such challenges. Network-theoretic approaches enable modeling and analysis of large-scale systems without the need for partitioning into smaller parts. System dynamics modeling, leveraged our the analysis of unintended consequences, enables consideration of organizational factors in a way that is often difficult to integrate into computational models typically used in design. Machine learning treats highly complex systems as a "black box", enabling the identification of patterns and correlations without a fully specified white box model. In these ways, this research provides metrics and methods for assessing robustness of complex engineered systems in early design.

Chapter 3: Detecting Indicators and Archetypes of Unintended Consequences in Engineered Systems

Starting at the highest level of hierarchy within Fig. 1.1, this chapter addresses the first research question: which sets of system characteristics tend to produce failures that can be described as unintended consequences? In the context of this research, unintended consequences are side effects of one system on another system, and are often perceived as unknown unknowns, unanticipated behavior, or irrational behavior. Unintended consequences occur either within a system of systems context or, in highly complex projects, between subsystems of an engineered system. In order to address this research question, further theoretical clarification on unintended consequences is necessary. This initial research was published in the Proceedings of the 2019 International Conference on Engineering Design and was cowritten by Hannah S. Walsh, Andy Dong, and Irem Y. Tumer [66]. The research directly addressing the first research question was published in the Proceedings of the 2020 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference and was cowritten by Hannah S. Walsh, Andy Dong, Irem Y. Tumer, and Guillaume P. Brat [67].

3.1 Motivation

Complex engineered systems are often prone to unanticipated behaviors. This is especially true in systems utilizing novel technologies or with high coupling. This chapter focuses on unintended consequences, which are side effects of one system on another system. Unintended consequences may cause system failures, but they can occur even when no part of the system fails *per se*. As a simple conceptual example, a part of one system may interact with another system in an unexpected manner, reducing the performance of the supersystem. Further, they occur due

to organizational factors or human-machine interaction within the broader system that interacts with subsystems, creating unexpected behaviors. These can relate to system safety [68] and can create system failures in systems of systems [69]. The causes of unintended consequences may often be clear retrospectively, but can be challenging to predict proactively. This is because many unintended consequences have little precedent or are relatively rare events, and occur due to complex relationships between systems or subsystems that can be difficult to model and analyze completely.

This research identifies leading indicators, or risk factors, that predict certain archetypes of unintended consequences by learning from historical data. This approach ventures that learning about the trends and patterns in the occurrence of unintended consequences can help prevent future unintended consequences. Archetypes in this context are exemplars of unintended consequences that contain certain sets of risk factors and can be modeled using system dynamics models. There are existing archetypes, some of which describe unintended consequences, modeled using system dynamics [68, 70]. These archetypes describe recurring patterns of behavior within systems and/or organizations. However, these archetypes are relatively high level and are not sufficiently granular to provide actionable information about potentially hazardous scenarios in the context of engineering design. Instead, in this research, a machine learning approach is used to discover more granular archetypes based on patterns of leading indicators in a large data set. Since the machine learning approach processes a large number of cases, it is possible to detect not only commonly occurring archetypes, but also less common archetypes. By detecting archetypes that are both more granular and less common, this research aims to improve existing descriptive and predictive models of unintended consequences in the context of complex engineered systems.

Using a publicly available data set of lessons learned at NASA [1], we detect a total of sixty six archetypes of unintended consequences. Lessons are first tagged according to whether they contain unintended consequences. For the lessons containing unintended consequences, certain risk factors are identified. Then, a self-

organizing map is used to learn archetypes containing similar occurrences of unintended consequences from the data set. A sample of the archetypes is then modeled using system dynamics in order to provide descriptive models and to validate the archetypes by comparing them with existing, high-level archetypes.

3.2 Background

The relevant literature is divided into two sections. First, literature on unintended consequences and related concepts is explored. The term is clearly defined and disentangled from related work. Second, existing work on archetypes is reviewed, with an emphasis on archetypes in the systems thinking context.

3.2.1 Unintended Consequences

Unintended consequences are increasingly being recognized as important, under-recognized vulnerabilities in engineered systems. In this section, unintended consequences are related to more commonly recognized concepts such as risk and failure. Additionally, existing work on unintended consequences in the context of engineering design is reviewed.

3.2.1.1 Related Concepts

A *risk* generally refers to the probability of occurrence of an uncertain adverse event. The characteristics and existence of the event are known; what is not known is the likelihood of the event happening. Risk quantification and management is a well-researched area [71, 72, 73]. Unintended consequences are, in this context, *under-recognized risks* to engineered systems. They are under-recognized because it is difficult for designers to model and analyze complex engineered systems with modularized architectures and design processes, especially the interfaces between modules. Whether these risks come from the interfaces between subsystems or between systems, many can be described as unintended consequences.

A *failure* is typically defined as a loss of ability to achieve an objective, in other words, performance outside specifications. The accepted understanding of failure within engineering has been that loss of performance in parts of a system causes loss of performance of the system. More recent work has acknowledged that failures can occur even when the individual parts perform within specification [74]. In this research, the understanding of unintended consequences is that they can lead to failure. In other words, studying unintended consequences is one way to identify under-recognized vulnerabilities that could lead to failures.

From a systems perspective, *emergence* refers to properties of a system that are not determinable by considering the constituent parts alone [16, 75, 76]. However, there are varying definitions and understandings of emergence [16]. The concept is sometimes used to mean undesirable behavior that is not explicitly designed-in [77]. Others attribute the concept of emergent behavior to a lack of understanding of a system [78, 79]. Despite the polysemous meanings, emergence typically refers to an effect or property that is produced by a bottom-up process, in other words, from the individual parts to the larger system within which the parts operate. Emergence as a concept takes a nuanced view on causality in the sense that a particular outcome can arise due to one or more sources, but explains that those outcomes occur due to a bottom-up process rather than purposeful actions taken at the system level and then propagated "downward" to the constituent parts, which is the view taken in studying unintended consequences.

3.2.1.2 Prior studies on unintended consequences

There have been prior studies seeking to address unintended consequences. For instance, Bahill defines unintended consequences as "*future* effects on *other* systems that might be caused by the new system being designed" and proposes a systematic approach for identifying unintended consequences during the systems engineering process [80]. Similarly, Watz and Hallstedt conceptualize unintended consequences as trade-offs related to sustainability and performance considerations in product requirements [81]. Van Boussuyt et al. proposed a method for identifying "irra-

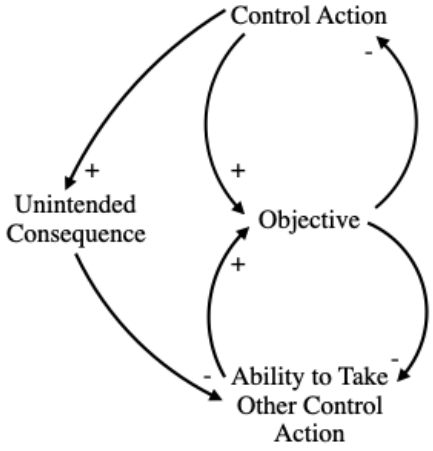
tional system behaviors", which are effectively similar to unintended consequences, using the functional basis and functional modeling [82]. In their research, irrational behaviors are failure flows from one system that affect another system within a system-of-systems context and that are unexpected and/or difficult to predict or understand [82]. This research is complementary to such approaches, but takes an alternative approach in using machine learning to understand trends in the occurrence of unintended consequences.

3.2.2 Archetypes

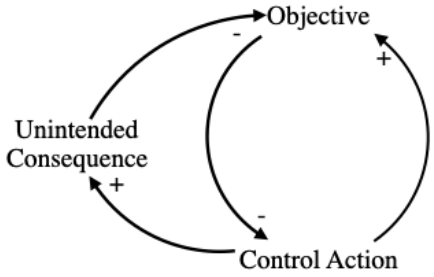
Archetypes refer to recurring patterns, typically of behavior or sets of characteristics [70, 83]. In systems thinking, they are represented using system dynamics modeling [84]. System dynamics models represent visually and mathematically the relationships between variables [85]. Archetype identification involves a trade-off between generality and specificity. All archetypes are, by their very nature, generalizable at least to some extent. However, the general systems archetypes [70] are very high level, whereas, for example, the archetypes of system safety proposed by Marais et al [68] are more specialized and therefore more descriptive.

Not all of the known system archetypes describe unintended consequences. Using the definition that unintended consequences are side effects of control actions, there are two high level system archetypes that describe unintended consequences. One is shifting the burden, which describes scenarios in which a design decision indirectly reduces the effectiveness of a more fundamentally effective design decision. This archetype is comprised of two balancing loops with an outer reinforcing loop, as in Fig. 3.1a. Another relevant archetype is fixes that fail. In this scenario, a design decision has a more directly adverse impact on its original objective. It is composed of an inner balancing loop and an outer reinforcing loop, as in Fig. 3.1b. These archetypes, though, are highly general and not descriptive enough to inform design changes.

Of the more granular archetypes, there are some that describe unintended consequences. Specifically, a number of archetypes of organizational safety proposed



(a) Shifting the Burden.



(b) Fixes that Fail.

Figure 3.1: Existing system archetypes describing unintended consequences.

by Marais et al. [68] describe unintended consequences. This includes stagnant safety practices in the face of technological advances, which is effectively a more specialized form of shifting the burden in Fig. 3.1a. In this example, the reduction in safety as a result of the lack of understanding of the new technology is the unintended consequence. Other archetypes from Marais et al. that relate to unintended consequences are unintended side effects of safety fixes and fixing symptoms rather than root causes [68]. This research aims to identify archetypes of similar level of granularity to these archetypes, but relating more to the design of complex engineered systems.

3.3 A Theoretical Framework for Unintended Consequences in Engineering Design

In this section, a theoretical framework for unintended consequences as they relate to engineering design is proposed. This framework enables the development of a theory-driven methodology to extract archetypes of unintended consequences in the next section. The theoretical framework is driven primarily by the appli-

cation of systems theoretic concepts, organized as two distinct systems "lenses" through which unintended consequences are viewed. From each lens, a principle is extracted.

3.3.1 First Lens: Control of Complex Systems

The model of socio-technical control from Leveson and Rasmussen can be used to conceptualize the system development process as a hierarchy of control actions [74, 86]. Within this model, there are two pillars – one for system development (design) and one for system operation. Each pillar has a number of levels of hierarchy, meeting at the lowest level of hierarchy: the physical process itself (e.g. flight dynamics). Each level of hierarchy exerts control actions, which act upon the hierarchical level(s) below each, and also receives information via feedback from the lower hierarchical level(s). The behavior at each level is controlled by the actions at each of the higher levels. A particular overarching design policy can govern large sets of design actions and thereby change what objectives and requirements are salient, eventually affecting the kinds of constraints that become active. Within this perspective, a number of different actions taken within system development and operation can be conceptualised as control actions. Generally, in the context of engineering design, designers take control actions such as design decisions to influence the engineered system.

Unexpected behaviors, however, can occur due to interfaces between the designed system and other systems with which it interacts. Control of complex systems is challenging because, by the definition of complexity, it is challenging to accurately model all these interfaces and predict the related consequences of the control action on other systems. Ashby's Law of Requisite Variety has implications on the nature of such actions [87]. Regulation acts to constrain the values of a certain variable, x; that is, to lessen the variety of potential outcomes [87]. In order to successfully regulate a system, the Law of Requisite Variety states that the regulator must have at least as many states as the number of states in the system that is being controlled [87]. Thus, when regulating a complex system, the na-

ture of the regulation itself is non-trivial, and regulations that have an insufficient number of states will fail to regulate the system effectively.

In short, the challenges of control of complex systems may lead to unexpected behaviors. Because of complexity, implementing constraints, as is the nature of the system development process according to the model of socio-technical control, is likely to lead to side effects. This mental model can be used as a lens through which to view unintended consequences, and is embodied in the first principle:

Principle 1 *Control of complex systems indicates that attempts to control the behavior of complex systems may lead to unintended consequences.*

3.3.2 Second Lens: Boundary Critique

Boundary critique utilizes, at its core, the concept of boundary judgments. The idea of a boundary judgment is that analysts constantly make judgments as to which pieces of information are relevant during system analysis. Information within a boundary is considered relevant; information outside a boundary is not. Boundaries are often defined using the area over which an emergent property, which is a system property not shared by the individual parts [88], is identifiable [89]. For example, an atom has properties that are not shared by its constituent protons, neutrons, and electrons. However, Churchman argues that boundaries are "social and personal constructs" [90], which implies that boundaries are, at least to some degree, arbitrary. The placement of these boundaries can have significant implications on the results of any analysis performed of the system [91]. That is to say, if a system analyst measures the performance of a system, that measurement may differ depending on where system boundaries are drawn. As a result, conflict occurs when two analysts draw different boundaries, thereby making different judgments regarding information that is considered relevant. The area included by one analyst but not the other is marginalized [91, 92]. Boundary critique is useful in a variety of fields. For instance, Midgley et al. applied boundary critique in developing housing services for older individuals [90].

Boundary critique becomes particularly important when considering which aspects of external systems should be considered as part of the systems engineering process. The relationship between an engineered system and other systems is aptly described by a system hierarchy: for example, an automobile is a part of a broader transportation system. The automobile can be conceptualized as a subsystem of the country's transportation system. The boundaries between hierarchical levels have traditionally been regarded as rigid; however, in actuality, the structure of the hierarchy is more flexible [93]. It can be difficult, then, to define so-called hierarchical boundaries, which are boundaries that occur between the hierarchical levels of a system [94], leading to the marginalization of external variables. For instance, designers of a particular subsystem of the automobile may marginalize important higher-level properties of the full automobile.

This phenomenon is partially unavoidable; more than one person is generally involved in the design of an automobile and many other systems. However, dividing work into rigid silos, when not carefully integrated, can lead to problems in heavily interconnected systems. There are many possible explanations for the marginalization of important external system elements. First, analysts could be ignorant of the marginalized elements [95]. Experts for one system may not be experts for another system, which could hinder their ability to identify key variables that are external to the engineered product. Second, studies have shown that interdisciplinary collaborations can be challenging and often devolve into individual disciplinary collaborations [96]. Thus, it may be challenging to involve experts in other domains, who may better be able to identify critical external elements. Third, intense emphasis on one objective can lead to the the neglect of others [97, 95]. This case is aptly described by an organization that is focused on short-term profit rather than on sustainability.

Regardless of the cause, marginalization of important elements during the systems engineering process can lead to unexpected behaviors. As such, it is important to critically consider the implications of these boundary judgments in the course of system design. Clearly, it is impossible to model all possible systems with which a

system may interact within the context of a systems engineering process. However, when important variables are marginalized, it is possible for an engineered system to have unintended consequences. If variables are marginalized, unintended consequences may be undetected [70] because of interactions with unforeseen or ignored variables.

In sum, considering system design through the lens of boundary critique emphasizes the importance of critical consideration of system boundaries, particularly those involved in a hierarchical structure. Consideration of these boundary judgments will reduce the marginalization of important system elements, thereby reducing the occurrence of unintended consequences of engineered systems. This lens enables an improved mental model for understanding unintended consequences and suggests possible methods for predicting and mitigating unintended consequences. This conceptual advance is embodied in the second principle:

Principle 2 *Boundary critique reveals potential adverse effects when system elements are excluded from analysis of effects.*

3.3.3 Propositions

We next present propositions based on the proposed theoretical principles.

3.3.3.1 Proposition 1: The Hierarchy of Control Actions

Within the model of socio-technical control [74], control actions occur within a hierarchy. High level control actions include design policies, whereas lower level control actions include design decisions. Control actions effectively influence all of the levels below the point of their implementation. Therefore, control actions at the lower levels of the model of socio-technical control influence only a portion of the system influenced by higher level control actions. Further, the systems controlled by higher level control actions are larger in scale and are often multi-disciplinary and/or heterogeneous. As a result, it is expected that it will be more difficult to trace the consequences of control actions at higher levels of the hierarchy.

Proposition 1 *Higher level control actions have more unintended consequences than lower level control actions.*

3.3.3.2 Proposition 2: Design Novelty

There is inherently less relevant historical knowledge available when designing novel systems. Therefore, it is more likely that there will be marginalized variables. As such, control actions will tend to produce unintended consequences.

Proposition 2 *More novel designs have more unintended consequences than less novel designs.*

3.3.3.3 Proposition 3: Design Methods

Causation-like processes choose the means in order to produce a certain effect [98]. In contrast, effectuation-like processes choose the effect based on the means available [98]. Engineering design processes generally are more causation-like than effectuation-like. Effects of engineering design processes, i.e. functions, are selected in the form of requirements and design objectives. After these effects are chosen, the design process selects the means needed to produce those effects. However, this causation-like process inherently involves prediction. That is, when selecting means in order to produce an effect, there is inevitably prediction involved in whether those means will produce the effect. There will inevitably be marginalized variables that prevents human designers' perfect prediction capability. Thus, a design process that minimises the amount of prediction required will be more effective in producing unintended consequences as compared to a causation-like process.

Proposition 3 *Design processes that aim to control possible effects decrease unintended consequences more effectively than processes that simply try to prevent effects.*

3.3.4 Framework Evaluation: Urban Air Mobility

In this section, a real-world example is used to evaluate the theoretical framework's explanatory power. Specifically, an urban air mobility (UAM) model is evaluated in order to ascertain whether theorized elements of unintended consequences are present in the model and whether they interact in a way that is consistent with the proposed theory. Urban air mobility refers to the idea of "air taxis", i.e. a number of unpiloted aircraft vehicles that transport people within an urban environment. The model is shown in Figure 3.2. All variables, including types, are provided in Table 3.1. Notes regarding assumptions made and initial values for stocks are also given in Table 3.2. The evaluation seeks to address the question: does the proposed theoretical framework help explain the unintended consequences of urban air mobility?

Table 3.1: Variables in system dynamics model.

Symbol	Variable Name	Type	Units
a_{UAM}	Airspace Allowed to UAM	Stock	m^2
e^T_{auto}	Automobile Emissions	Auxiliary	$tCO_2e/year$
s_{auto}	Automobile Seat Allowance	Parameter	$passengers/automobile$
v_{avg}	Average Airspeed	Parameter	m/s
t_{flight}	Average Flight Time	Auxiliary	s
t_{hover}	Average Hover Time	Parameter	s
r	Average Range	Parameter	m
d_{batt}	Battery Density	Parameter	Ws/kg
E_{batt}	Battery Energy	Auxiliary	J
m_{batt}	Battery Mass	Auxiliary	kg
D_c	Clean Energy Demand	Auxiliary	$J/year$
I_c	Clean Energy Initiative	Parameter	J/tCO_2e
e_E	Emissions from Energy Production	Auxiliary	$tCO_2e/year$

Continued on next page

Table 3.1 – *Continued from previous page*

Symbol	Variable Name	Type	Units
e_{auto}	Emissions per Automobile	Parameter	$tCO_2e/year/automobile$
D_E	Energy Demand	Auxiliary	$J/year$
D_f	Fossil Fuel Demand	Auxiliary	$J/year$
F_f	Fossil Fuel Emissions Factor	Parameter	tCO_2e/J
g	Gravitational Constant	Parameter	m/s^2
p	Incidents of Privacy Violation	Flow	$incidents$
N	Noise	Flow	dB
F_N	Noise Influence Factor	Parameter	$\%/year/dB$
N_{UAV}	Noise per UAV	Parameter	dB/UAV
n_{auto}	Number of Automobiles	Auxiliary	$automobiles$
n_{UAV}	Number of eVTOL UAVs for UAM	Stock	$UAVs$
m_{pass}	Passenger Mass	Parameter	$kg/passenger$
PWR	Power to Weight Ratio	Parameter	W/N
P	Public Acceptance	Stock	%
F_P	Privacy Influence Factor	Parameter	$\%/year/incident$
F_p	Privacy Violation Factor	Parameter	$incidents/m^2$
e^T	Total Emissions	Auxiliary	$tCO_2e/year$
E^T_{UAV}	Total UAV Energy Consumption	Auxiliary	$J/year$
T	Transportation Needs	Parameter	$automobiles$
C	UAM Deployment	Flow	$UAVs/year$
C_P	UAM Initiative	Parameter	$UAVs/tCO_2e/\%$
m_{UAV}	UAV Airframe Mass	Parameter	kg/UAV
F_a	UAV Airspace Factor	Parameter	m^2/UAV
m_{empty}	UAV Empty Mass	Auxiliary	kg/UAV
E_{UAV}	UAV Energy Consumption	Auxiliary	J/UAV
$m_{payload}$	UAV Payload Mass	Auxiliary	kg/UAV
s_{UAV}	UAV Seat Allowance	Parameter	$passengers/UAV$

Continued on next page

Table 3.1 – *Continued from previous page*

Symbol	Variable Name	Type	Units
m_{UAV}	UAV Mass	Auxiliary	kg/UAV
w_{UAV}	UAV Weight	Auxiliary	N/UAV
v	Visual Pollution	Flow	$UAVs/m^2$
F_v	Visual Pollution Influence Factor	Parameter	$\%/year/(UAV/m^2)$
R_{UAV}	Yearly Trips per UAV	Parameter	$1/year$

System dynamics modeling reveals four unintended consequences in the model. First, UAM interacts with existing energy systems such that a finite ability to produce clean energy to power the UAVs increases fossil fuel demand. This unintended consequence is modeled as a reinforcing loop between UAM Deployment and Fossil Fuel Demand, with the intended consequence being the balancing loop between UAM Deployment and Total Emissions. Second, UAM creates noise pollution. Third, UAM creates visual pollution. Fourth, UAM raises privacy concerns. The latter three unintended consequences have the effect of reducing Public Acceptance, which in turn decreases the availability of UAM Deployment as a control action.

First, Principle 1 is evaluated in terms of its ability to explain the unintended consequences of urban air mobility. This principle states that attempts to control the behavior of complex systems may lead to unintended consequences. Thus, in evaluating this principle's explanatory power, it is necessary to determine that there is an attempt to control the behavior of a system. Using the model of socio-technical control, an attempt to control the behavior of a system can be described as a control action. In the urban air mobility model, the effects that are modeled are primarily related to the introduction of urban air mobility into society. In other words, the control action of interest is UAM Deployment. From the control of complex systems lens, then, all unintended consequences in the model should be traceable to the control action, UAM Deployment. Second, Principle 2 is evaluated. This principle uses boundary critique as a means of understanding how

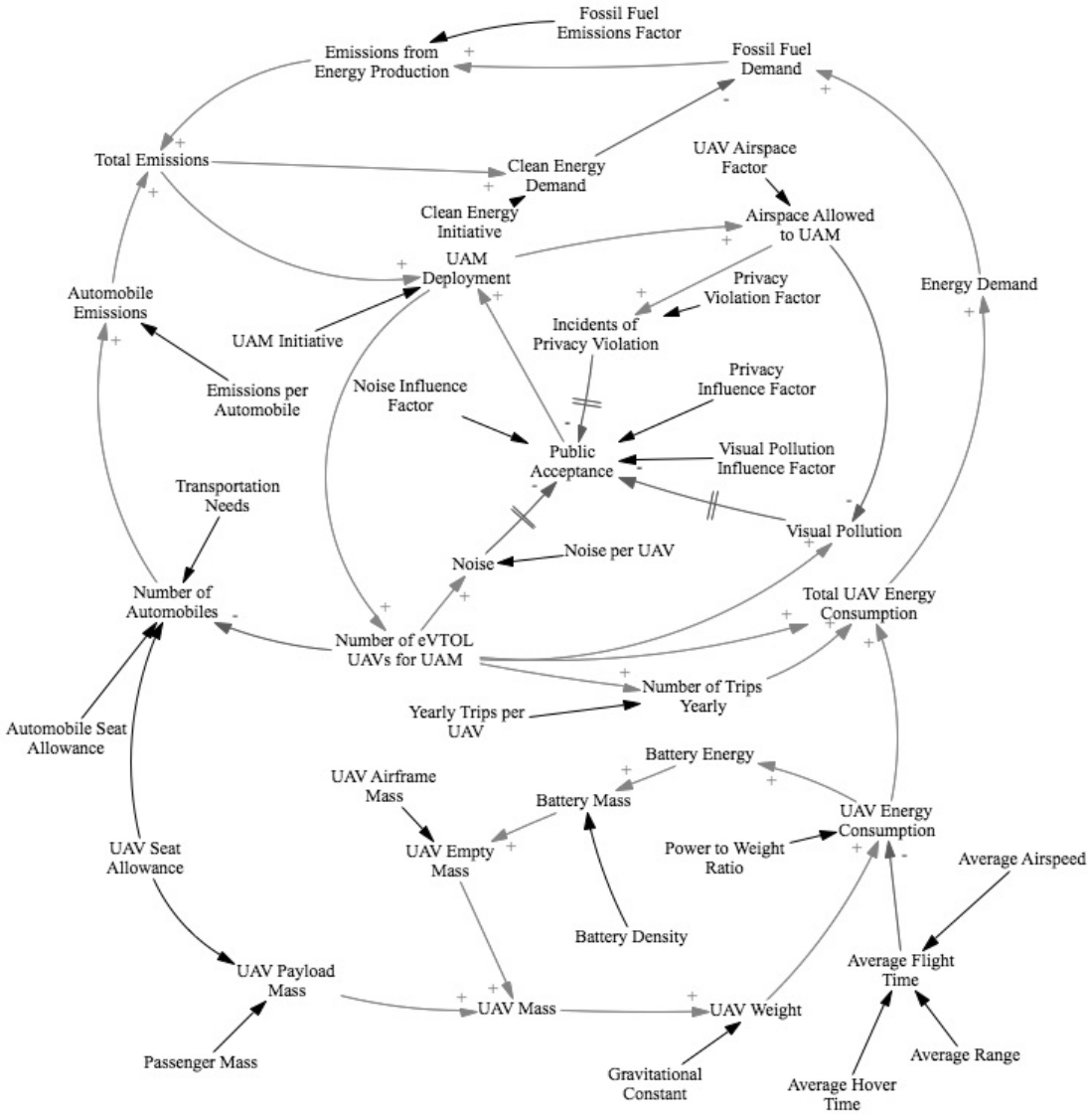


Figure 3.2: System dynamics model of eVTOL UAVs for UAM application, made in Vensim.

Table 3.2: Assumptions in system dynamics model.

Variable	Notes & assumptions
Airspace Allowed to UAM	Assume no vertical expansion; initial value 100
Automobile Emissions	Neglect growth of electric car industry
Automobile Seat Allowance	Average
Average Airspeed	Neglect stop-and-go emissions
Clean Energy Demand	Assume zero emissions for clean energy
Emissions from Energy Production	Includes only emissions from fossil fuels
Emissions per Automobile	Average
Energy Demand	Neglect other energy consumers
Fossil Fuel Emissions Factor	Estimate
Gravitational Constant	Assume operation close to Earth's surface
Incidents of Privacy Violation	Assume no change in privacy laws
Noise	Only for UAVs for UAM
Noise per UAV	Average
Number of Automobiles	Neglect growth of electric car industry
Number of eVTOL UAVs for UAM	Assume no unorganized use; initial value 10
Passenger Mass	Average; including luggage
Public Acceptance	Initial value 37% (approval)
Privacy Violation Factor	Average per unit airspace
Total Emissions	Yearly, only from automobiles and UAM
Total UAV Energy Consumption	Assumes no unorganized use of UAVs
Transportation Needs	Assume no change from $0 < t < T$
UAV Airframe Mass	Empty mass without batteries
UAV Energy Consumption	Assume no change from $0 < t < T$
UAV Payload Mass	Average
UAV Seat Allowance	Assume full autonomy is possible
UAV Mass	Average
UAV Weight	Assume constant altitude of operation
Visual Pollution	Average

the marginalization of key system elements may cause unintended consequences. In the UAM model, failure to consider (i.e. marginalization of) the effects of UAM energy demands on fossil fuel demand could lead to an increase in emissions, as opposed to a reduction in emissions. Marginalization of privacy, visual pollution, and noise concerns could also reduce the use of UAM Deployment if unaddressed. Overall, the proposed theoretical framework provides a useful mental model for considering the causes and considerations involved in unintended consequences.

3.4 Detecting and Characterizing Archetypes of Unintended Consequences

The previous section provides a theoretical foundation for the study of unintended consequences. In this section, archetypes of unintended consequences are identified in relation to a number of leading design indicators, which in this research are called risk factors. The goal of this section is to (1) identify and descriptively model a number of archetypes of unintended consequences that occur in complex engineered systems and (2) provide a path forward for predicting unintended consequences by understanding how certain risk factors interact to increase the likelihood of occurrence of an unintended consequence.

3.4.1 Methodology

The methodology used in this research for extracting the archetypes of unintended consequences is divided into three parts, each detailed in this section. In the first part, the method used for archetype detection is presented. In the second part, the detected archetypes are characterized using system dynamics modeling. In the third part, a validation method for the archetypes is described in which the identified archetypes are compared to the high level known archetypes. All stages are summarized in Fig. 3.3.

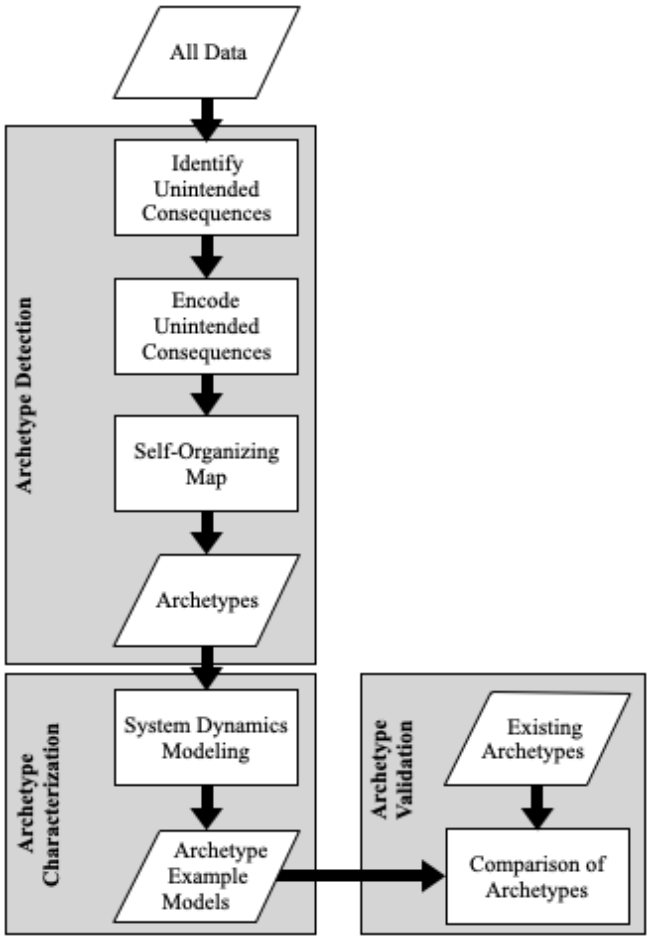


Figure 3.3: Flowchart of the methodology used to identify, characterize, and validate archetypes of unintended consequences.

3.4.1.1 Data Set

This research uses a data base of historical cases of unintended consequences in order to identify unintended consequences. Specifically, the National Aeronautics and Space Administration (NASA) Lessons Learned Information Systems (LLIS) [1] data base is used. This data base contains 2079 lessons learned.

3.4.1.2 Archetype Detection

First, archetypes of unintended consequences are detected using a combination of human encoding and machine learning. In this stage, the theory-driven human coding scheme enables the identification of unintended consequences from historical cases. Human coders also use rules to encode leading design indicators, which are called risk factors, of the cases of unintended consequences. The last step is to encode unintended consequences into a self-organizing map. The self-organizing map then identifies archetypes in the form of clusters of encoded characteristics that frequently occur together. Each of these steps will be detailed in the remainder of this section.

Identifying Lessons with Unintended Consequences Not all lessons within the database contain unintended consequences; therefore, a method is necessary for determining whether an unintended consequence is contained in a given lesson. This set of rules is derived from the definition of unintended consequences and prior literature that differentiates unintended consequences from other concepts such as emergence, as described in Section 2. A simple tool is proposed in Table 3.3 in order to check that the conditions are met for a scenario to contain an unintended consequence. An identifiable control action is required. A control action is an action taken during system development, such as a design decision. The model of socio-technical control from Leveson and Rasmussen can be used to conceptualize the system development process as a hierarchy of control actions [74, 86]. Control actions can be high level, such as policy and regulations, or lower level, such as

Table 3.3: Framework used for identifying unintended consequences from the dataset.

Element	Description
Control Action	Organized action such as design policy
Intended Effect	Behavior desired in taking control action
Side Effect	Behavior resulting from the control action, separate from intended effect

design decisions. Systems engineers utilize control actions in order to influence the completed design. The control action must have both an intended effect, i.e. goal of the control action, and a side effect, which is an unintended consequence. If these three elements can be identified, then the scenario contains an unintended consequence. A human coder uses this tool in order to identify whether each lesson contains an unintended consequence.

As an example, consider the following scenario. Following September 11, 2001, commercial aircraft were retrofitted with a locking door between the cockpit and passenger area. If a pilot were to exit the cockpit, the first officer would be required to unlock the door using a control inside the cockpit to allow the pilot to re-enter. This door lock control was similar in terms of placement and operation to the rudder control in the Boeing 737-700, causing a first officer on one occasion to inadvertently activate the rudder control instead of the door lock control [99]. This error sent the aircraft into a nose dive [99]. Conventionally, this event could be described as a human error. However, in the context of this research, it can also be described as an unintended consequence. Using Table 3.3, the control action is the requirement to add a locking door between the cockpit and passenger area, the intended effect is to prevent illicit takeover of the cockpit area, and the side effect is inadvertent activation. Since all three conditions are met, this scenario qualifies as an unintended consequence.

Encoding Unintended Consequences Using Risk Factors Next, each lesson that contains an unintended consequence is analyzed to determine which risk

factors, given in Table 3.4, contribute to the occurrence of the unintended consequence. Similarly to the use of safety indicators in accident analysis [100, 101], risk factors are contributors to adverse events. Risk factors are identified during the encoding process, and are added to the list of risk factors in Table 3.4 as they are discovered in the data base. The identified risk factors must completely describe, at least based on the information given in the data base, the occurrence of the unintended consequence, and are based on the characteristics of the system *before* the unintended consequence occurs. Encoding the risk factors for each lesson is a four step process:

1. Read the entire lesson. Identify the intended (main) and the unintended (side) effects. There may be multiple unintended consequences in one lesson. In this case, focus on the unintended consequence with the most severe impacts.

2. Define the system boundary. Control actions act on a system boundary and are not themselves contained in the defined system boundary. This step helps with the identification of the control action, next.

3. Identify the control action. If there are multiple, choose the one relating to the most severe unintended consequence.

4. Identify the risk factors. Identify only risk factors that relate to the qualities of the system or of the control action, *not* qualities of the effect or outcome.

Table 3.4: Risk factors with example text from the NASA LLIS database [1].

	Risk Factors	Description	Example
1	Complexity	Inability of a human to assess the entire state of the system	Pulse width modulator circuit card assemblies (PWM CCAs), valued at $\$150,000$ each were inadvertently excessed to the KSC excess property storage area. The PWM CCAs were attached to holding fixtures, and were subsequently purchased as scrap metal by a local salvage company. The sole decision criterion for excessing items was usage rate. The computerized logistics system documented the holding fixtures as not being in use. The individuals involved in the excessing process *believed that they were only excessing holding fixtures.* The holding fixtures were entered as separate items from the PWM CCAs, because *the system lacked the capability to document and track integrated components.*

Continued on next page

Table 3.4 – *Continued from previous page*

	Risk Factors	Description	Example
2	Coupling	Elements within the system are tightly interconnected and interdependent, e.g. presence of a bridging node [102]	TCS blankets by nature have a *close and critical dimensional relationship to the surrounding hardware* they are insulating.
3	Human-machine interaction	There is a human operator interacting with the system or within the system	Clean room certified adhesive tape 5413 was used inadvertently in construction of multi-layer insulation (MLI) blanket for a vacuum chamber (instead of clean room certified adhesive tape 1205). The *two tapes look virtually identical and are packaged similarly.*
4	Integration into an existing system	A new feature or technology is added to an existing system	*Using new equipment* in hazardous environments without full understanding all the capabilities and restrictions can easily cause injury or equipment destruction.

Continued on next page

Table 3.4 – *Continued from previous page*

	Risk Factors	Description	Example
5	Lack of isolation or protection from environment	Interactions between the system and its environment are not appropriately isolated or protected	After the launch of the NOAA-15 spacecraft, an EMI/EMC interference affecting science data was noted in the Advanced Microwave Sounding Unit (AMSU)-B data. This interference was subsequently attributed to the SAR and S band transmitters and was due to inadequate *AMSU-B shielding.*
6	Scale of effect	A large number of individuals or units can be affected by the control action or the system, e.g. high degree nodes	The *combined problems* caused a broadcast storm across the network and affected the VLANs on the trunked interfaces and rendered the network unable to process user traffic and thus causing the user servers and field instrumentation to lose connectivity.

Continued on next page

Table 3.4 – *Continued from previous page*

	Risk Factors	Description	Example
7	Decentralized decision making	An action requires the decision-making of a number of individuals or units	On April 29, 2010, *a subcontractor employee* received second and third degree burns while working at Building 01385, also known as the Mission Control Center (MCC). The Injured Person (IP) and Coworker were tasked with preparing a transformer, associated with an electrical substation, for removal as part of the MCC demolition project. It was their understanding that this electrical substation was de-energized.

Continued on next page

Table 3.4 – *Continued from previous page*

	Risk Factors	Description	Example
8	Centralized decision making	An action requires the decision-making of a central prioritized individual or unit	Space Vision System (SVS) targets have become an undue burden during EVA. The requirement to restrain cables away from SVS targets added 45 minutes to 1 hour EVA time and will change during the mission. The EVA community has been informed that the targets can be easily damaged by EVA loads, yet the targets are in or near translation paths. A better understanding of the criticality of keeping the targets free of obstructions and the sensitivity of the targets to EVA contact is required. The *SVS community must understand the EVA environment* and *not place unrealistic constraints on the conduct* of the EVA.
9	Time dependency	The need to take action is time-sensitive	Add to this dangerous situation the "*hurry up, let's get this job done*" attitude, is only adding fuel to an accident waiting to happen.

Once identified, risk factors are also given weights on a scale of $[0, 1]$ in increments of 0.1 representing their relative importance. A weight of 1 is the most important while a weight of 0 is the least important. The sum of the risk factor weights for a single lesson must sum to 1. To obtain weights for each risk factor, the human coder counts the number of times a risk factor is mentioned (often implicitly) in the lesson. Then, the risk factor count for a single risk factor is divided by the total number of risk factor mentions to obtain the weight for an individual risk factor. If rounding is needed, less important risk factors are rounded down.

The encoding is validated using a second coder for a portion of the data set. The validation measures the consistency between the two coders. The second coder first reviews a training set generated by the first coder, which contains identified risk factors and weights along with the original textual description of the risk factor. Then, the second coder assesses a different portion of the lessons containing unintended consequences, approximately 10% of these lessons. This data is then used to validate the coding scheme using Krippendorff's alpha [103, 104] using a MATLAB implementation [105]. Upon completion of the validation, the human coders discuss their disagreement and come to consensus on the lessons on which they originally disagreed.

Identifying Archetypes with a Self Organizing Map Once the data has been encoded, a self-organizing map is used for clustering and archetype identification. A self-organizing map is an artificial neural network (ANN) based approach to clustering. There are two layers in a self-organizing map. The input layer has dimensions $n \times m$, where n is the number of lessons and m is the number of risk factors. Input layer data should in general be normalized [106]. In this study, the data is normalized in the previous step when weights are assigned. The next layer, called the Kohonen layer, has dimensions $l \times l$ where l^2 is the number of neurons in the map, which is also the maximum number of clusters. The second layer can either have a rectangular or hexagonal structure. In a rectangular structure, each non-edge neuron has four neighbors [106]. In a hexagonal structure, each non-edge

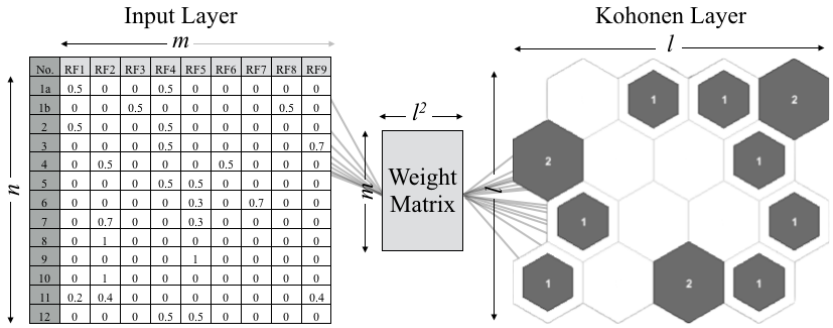


Figure 3.4: Self-organizing map minimal example.

neural has six neighbors [106]. A hexagonal map is used in this research. A $m \times l^2$ weight matrix lies between the two layers. A heuristic is used to estimate the required map size for a given data set size, Eq. 3.1 [107]. The structure of the self-organizing map is visualized in Fig. 3.4.

$$l^2 \approx 5 \times \sqrt{n} \tag{3.1}$$

The map is implemented in the SOM Toolbox in MATLAB [108]. The learning algorithm is as follows:

1. Initialize weights w with small random values.
2. Select input vector x_i.
3. Identify neuron k with the highest similarity to the input.
4. Reward winning weights and neighbor weights w.
5. Set $t = t + 1$. Repeat Steps 2–4 until convergence.

The third step is sometimes called the competitive step since it selects a "winning" neuron based on the similarity of its weights w_j to the input x_i. Similarity is measured using Euclidean distance, given in Eq. 3.2 and sometimes called the discriminant function, where d_k is the Euclidean distance for neuron k and m is

the number of attributes. Eq. 3.3 is used to find c_i, the winning neuron for input x_i [106], once all discriminant functions have been computed. In this equation, t is the iteration step.

$$d_k(t) = \sqrt{\sum_{j=1}^{m}(w_{jk}(t) - x_{ij})^2} \tag{3.2}$$

$$c_i(t) = \arg\min_k\{d_k(t)\} \tag{3.3}$$

The fourth step is sometimes called the combined cooperative and adaptive step. In this step, the weights of the winning neuron and its neighbors are updated. Weights are updated more significantly for the winning neuron, and less significantly for its neighbors. The weight update is governed by Eq. 3.4, which is dependent on $\alpha(t)$, which is the learning rate, and $h_{ck}(t)$, which is the proximity to the winning neuron. $\alpha(t)$ should decrease with respect to t [106] and falls between 0 and 1. $h_{ck}(t)$ is called the neighborhood function and is defined in Eq. 3.5 [109], where r is the location of the neuron and $\sigma(t)$ controls the width of the neighborhood function and decreases with respect to t [109].

$$w_k(t+1) = w_k(t) + \alpha(t)h_{ck}(t)\big[x_i(t) - w_k(t)\big] \tag{3.4}$$

$$h_{ck}(t) = \exp -\frac{||r_c - r_k||^2}{2\sigma^2(t)} \tag{3.5}$$

Convergence of the learning algorithm is determined after a specified number of iterations has been completed, usually 500 times the number of neurons [106]. After learning, the neurons containing at least one lesson are interpreted as a cluster. Clusters contain lessons with similar sets of risk factors. Then, silhouette coefficient is used to validate the cluster quality. For each sample i, $s(i)$ measures how well the sample fits into its cluster C_i. Higher silhouette coefficients indicate a high quality cluster. Low values may indicate that the number of clusters should be reassessed.

3.4.1.3 Archetype Characterization Using System Dynamics

The groups of risk factors are modeled using system dynamics in order to better understand the causal mechanisms at play and to visualize their relationship to control actions and unintended consequences. The previous clustering step produces an empirical result in which certain risk factors are found to occur together. These themselves can be interpreted as archetypes, but for consistency with more common models, it is necessary to model a sample of lessons using system dynamics. Five archetypes are selected for modeling; from each of these archetypes, one lesson is chosen.

Building system dynamics models involves returning to the original textual description of the lesson. Specifically, causal loop diagrams are used. These are mainly used for visualization of system behavior. Loops are important structures in causal loop diagrams. Balancing loops, which contain an odd number of positive edges, show stabilization over time in terms of behavior. Reinforcing loops, which contain an even number of positive edges, show exponentially increasing or decreasing behavior over time. Initially, all relevant variables should be included in the system dynamics model. Once the model is completed, the modeler can simplify in order to compare the lesson model with the general archetype. Generally, the control action, side effect (unintended consequence) and main or intended effect will be represented in the final system dynamics model.

3.4.1.4 Archetype Validation

The causal loop diagrams from each of the five selected archetypes are compared to existing, known archetypes of unintended consequences in order to validate their consistency with existing theory. The simplified causal loop diagrams are easily comparable to more general archetypes by analyzing the fundamental loop structure. If the same loop structure is found, the more granular archetype can then be said to be a more specialized version of the general archetype.

Table 3.5: First ten lessons identified as unintended consequences in a total of 381 lessons. Detailed information about each lesson is available from the NASA LLIS database [1].

No.	Description	Lesson No.
1	Thermal Control System blankets	2716
2	Aligning system development models	24502
3	Liquid hydrogen tank	5004
4	Aircraft tow incident	884
5	Hold-down strap problem	631
6	Fan screen placement problem	628
7	SOFIA tow incident	857
8	KAO bulkhead fit problem	928
9	Test procedure deviation	1601
10	Sheet metal handling	1032

3.4.2 Results and Analysis

Approximately 18% of the lessons in the database, for a total of 381 lessons, are identified as being unintended consequences. The first ten of these lessons are given in Table 3.5. The lessons that are not unintended consequences may describe failures of the main effect. Others describe successes rather than adverse events. Finally, other lessons do not contain sufficient information to determine whether an unintended consequence is present. For an initial analysis of the breadth of topics in the 381 identified lessons of unintended consequences, topic modeling is performed. Specifically, a goodness of fit of latent Dirichlet allocation (LDA) models is used. This analysis finds forty topics in the lessons containing unintended consequences, given in Fig. 3.5 along with the probability of each topic occurring. This analysis confirms that a breadth of topics is studied, making this data set suitable for archetype identification. This is because archetypes describe behavioral patterns occurring in different contexts, so it is desirable for the data set to contain many different situations.

Inter-rater reliability, as measured using Krippendorff's alpha, is found to be 0.7239, where a value of 1 indicates perfect agreement. Krippendorff's alpha in this research measures the agreement between the clusters found from ratings of two different raters and the data is treated as nominal. Krippendorff's alpha is

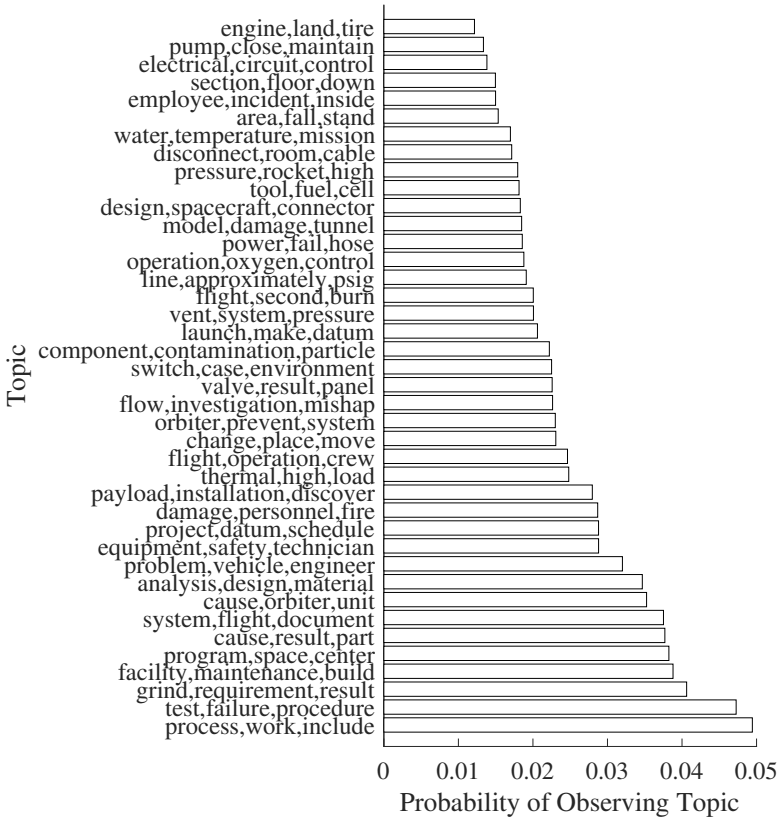


Figure 3.5: Probability of observing each topic in the unintended consequence data set illustrating breadth of topics.

Table 3.6: Krippendorff's alpha for inter-rater reliability of each risk factor.

Risk Factor	**Krippendorff's Alpha**
Complexity	0.92
Coupling	0.93
Human-Machine Interaction	0.80
Integration into an Existing System	0.99
Lack of Isolation or Protection from Environment	0.89
Scale of Effect	0.78
Decentralized Decision Making	0.60
Centralized Decision Making	0.73
Time Dependency	1.00

also computed by risk factor, treating the data as interval data. These results are given in Table 3.6. Decentralized decision making has a relatively low inter-rater reliability, indicating that it may require further definition and clarification for future research. Other risk factors including integration into an existing technology and time dependency, however, were particularly well agreed upon.

A self-organizing map of dimensions 10×10 is used. This results in the identification of sixty-six archetypes, with a mean cluster silhouette coefficient of 0.9699. The occurrences of the archetypes are given in Fig. 3.6. As in Fig. 3.6, many of the archetypes occur infrequently, i.e. their probability of observation is low. At least in this data set, these archetypes are considered rare. This may imply that these archetypes are unlikely to be anticipated due to a lack of relevant prior experience. A larger scale study is necessary to verify this finding.

Additionally, Table 3.7 provides the relative influence of each of the risk factors in the data set. Influence of a risk factor is measured as the sum of all its ratings in the study. The most influential risk factor is coupling. Interestingly, decentralized decision making is the second most influential risk factor, while centralized decision making is the second least influential risk factor. This implies a potential design rule: centralize decision making in order to prevent more unintended consequences. Further research is needed to verify this finding across larger data sets.

The identified archetypes contain between one and four risk factors, roughly normally distributed, as shown in Fig. 3.7a. Therefore, only 11%–44% of the

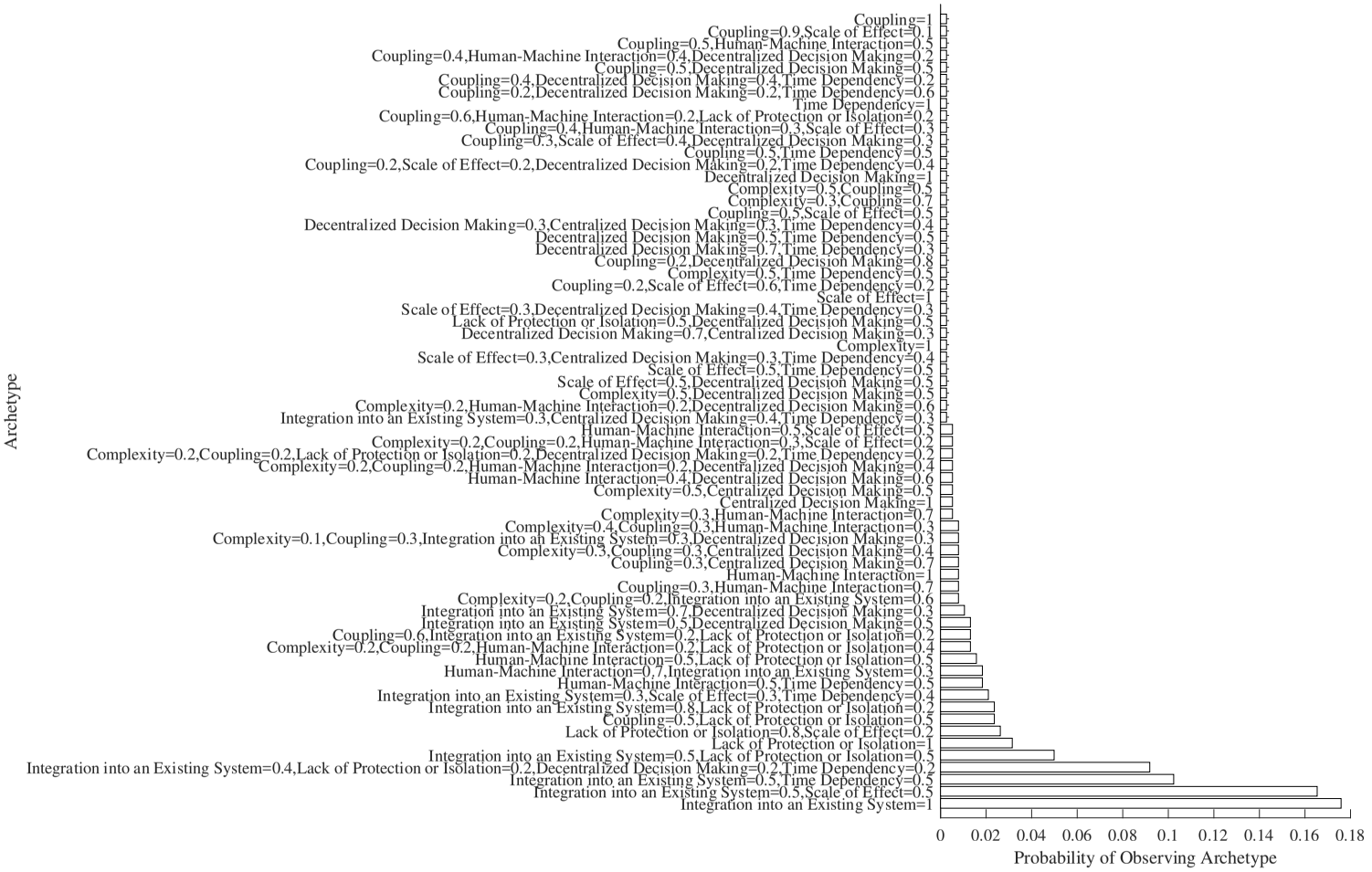
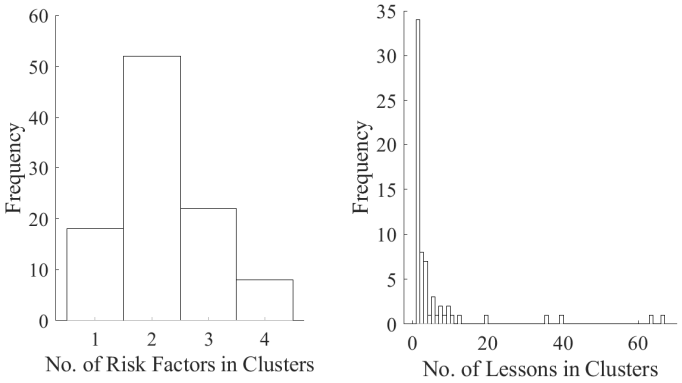


Figure 3.6: Probability of observing each archetype.

Table 3.7: Risk factor influence in the dataset, ranked by sum of ratings across all lessons.

	Risk Factor	**Sum of Ratings**
1	Coupling	94.7
2	Decentralized Decision Making	86.7
3	Lack of Protection or Isolation from Environment	50.3
4	Integration into an Existing System	47.1
5	Human-Machine Interaction	36.5
6	Scale of Effect	21.4
7	Complexity	17.0
8	Centralized Decision Making	14.3
9	Time Dependency	12.9



(a) Histograms of risk factors per cluster.

(b) Histogram of lessons per cluster.

Figure 3.7: Histograms of archetype data.

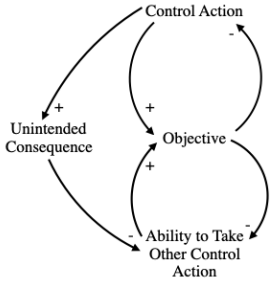
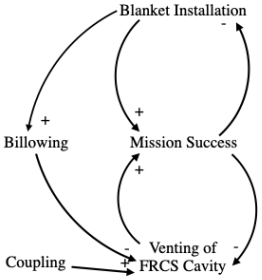
risk factors are required in order to fully describe the formation of an unintended consequence, implying that the archetypes are relatively well differentiated. As shown in Fig. 3.7b, most archetypes describe only a small number of lessons. This, along with Fig. 3.6, implies that the data set contains rare archetypes.

Causal loop diagrams for one lesson from five different archetypes are given in Table 3.8. These models describe how risk factors contribute to the formation of unintended consequences. As an example, Lesson 18 describes how Human-Machine Interaction, Coupling, and Complexity contribute to the unintended consequence, which is that Retainer Usage causes Undetected Cracks. As demonstrated in Table 3.8, each of the lesson CLDs is a derivative of one of the high level system archetypes. This is easily demonstrated by comparing the loops between the general archetypes and lesson CLDs in Table 3.8. This finding verifies the consistency between the causal mechanisms that generate unintended consequences more generally and those at work in the lesson CLDs. The lesson CLDs, however, are more granular and specialized because they include risk factors. In Lesson 1333, for instance, Lack of Protection or Isolation from Environment creates the conditions under which Radiation Vulnerability will increase. In other words, it creates a bifurcation in which COTS Hardware Usage increases Radiation Vulnerability. Without this risk factor, COTS Hardware Usage may not increase Radiation Vulnerability, or at least not to the degree that it would affect Mission Success. This finding implies that the identified archetypes of unintended consequences are specialized versions of the general archetypes.

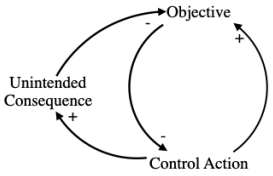
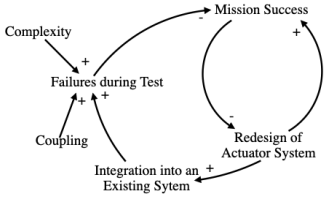
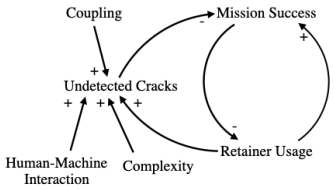
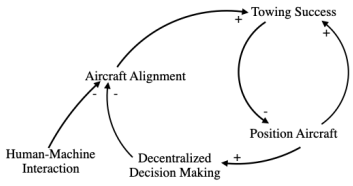
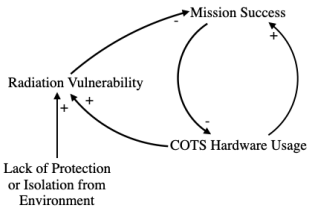
This research has identified sixty-six archetypes of unintended consequences – significantly higher than the existing number of archetypes of unintended consequences. This finding is likely due to the use of machine learning on a large data set. Machine detection enables the processing of large amounts of lessons which would be infeasible for a human to model and process. The identified archetypes are also more specialized than some of the existing archetypes. This means, necessarily, that they are also less generalizable. Each set of archetypes, the more general and the more specific, is useful depending on the application.

Table 3.8: Causal loop diagrams of five lessons, each from a different archetype. One of these archetypes is a specialized case of the more general, well-known shifting the burden archetype. Four are specialized cases of fixes that fail.

Shifting the Burden

General Archetype	***Lesson CLDs***
	 (a) CLD of lesson 2716 with risk factor coupling=1.

Fixes that Fail

General Archetype	***Lesson CLDs***
	 (b) CLD of lesson 1607 with risk factors complexity=0.2, coupling=0.2, integration into an existing system=0.6.
	 (c) CLD of lesson 18 with risk factors complexity=0.2, coupling=0.2, human-machine interaction=0.6.
	 (d) CLD of lesson 884 with risk factors human-machine interaction=0.4, decentralized decision making=0.6.
	 (e) CLD of lesson 1333 with risk factor lack of protection or isolation from environment=1.

An interesting observation from this study is the risk factors that are more or less influential in producing unintended consequences. Specifically, decentralized decision making, much more than centralized decision making, is associated with the formation of unintended consequences. This finding is of particular concern considering the decentralized systems such as the Internet of Things, which are becoming more widespread. Research on risk factors that are likely to produce unintended consequences may be useful for improving design strategies surrounding the prevention or mitigation of unintended consequences. Additionally, some of the risk factors identified from the database correspond to the propositions in Section 3.3. Specifically, integration into an existing system, which is one of the risk factors identified, has a clear correlation to Proposition 2, which is related to the novelty of a design.

3.5 Chapter Summary

In understanding the structural characteristics of robustness, this chapter begins with a high-level view of a system and other systems with which it interacts. At this level, system dynamics models enable an assessment of the structure of the system. Patterns and characteristics of robustness are explored using archetypes of unintended consequences. The approach utilizes a scheme for human raters to encode semantic descriptions of unintended consequences such that clusters are identifiable using machine learning. These clusters are then modeled using system dynamics and interpreted as archetypes of unintended consequences. Sixty-six archetypes are identified, compared to the small existing number of archetypes. The archetypes are verified to be specialized versions of existing, high-level archetypes of unintended consequences. In general, archetypes are useful as tools to test various dynamic theories of adverse events. Given a number of identified risk factors, their associated archetypes can be used as risk assessment tools in order to identify potential hazards that can be incorporated into, for example, an FMEA.

This section primarily investigates side effects of systems on other systems. Some of the unintended consequences studied in this chapter, however, occur be-

tween subsystems of highly complex, large scale systems, rather than within a system of systems context. In other words, some unintended consequences are perturbations of one subsystem on another subsystem – meaning they occur at a system level rather than at a system of systems level. At this level, it is possible to investigate how system architectural decisions affect fragility. The second research question – explored in the next chapter – addresses the impact of system architecture on fragility.

Chapter 4: Identifying Robust Architectures for Complex Engineered Systems

This chapter addresses the second core research question: which system topologies are most vulnerable to failure? A network-based approach is taken in order to analyze vulnerabilities in complex engineered systems. The content of this chapter was published in the Journal of Mechanical Design and in the Proceedings of the 2018 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference and was cowritten by Hannah S. Walsh, Andy Dong, and Irem Y. Tumer [110, 111].

4.1 Motivation

Engineering design utilizes a number of design rules and principles. Examples of the use of design rules include axiomatic design, in which the independence axiom states that functional requirements should be independent and the information axiom states that the design with the lowest information content should be selected [112]. By the principle of simplicity, a design should be explicitly simple, and any necessary complexity should be intrinsic [113]. Using more elements than needed improves reliability, according to the principle of redundancy [114] – although, the system topology also plays a significant role, as is explored in this research.

One design principle of particular interest is modularity. In general, modularity refers to the division of a system into smaller parts, usually with tighter interconnectivity within modules than between modules. Modularity can be defined more specifically according to the context. *Modularity in production* aims to manufacturing components separately in order to streamline production [115]. *Manufacturing modularity* considered the dependencies related to manufacturing [116]. *Modularity in use* enables freedom for the end user to customize their prod-

uct by selecting from a set of modules according to their own needs [115], such as purchasing a mouse, keyboard, and computer from separate companies. In this work, however, the focus is *modularity in design*, also known as modular design. In modular design, systems are divided into subsystems in which the interfaces between subsystems are well defined [115, 117]. System modularization can be accomplished by dividing a system into subsystems during the design process. Additionally, modules can be identified using the system's specified architecture [118]. Ideally, modules should have minimal interactions with other modules [119].

Modularity is widely utilized in system design, especially for management of complex products and systems. However, by modularizing a system and studying the behavior and properties of the modules separately, it is often difficult to trace emergent behaviors that arise from the interaction between modules. While modularity-based approaches are convenient for dividing work on a large-scale project, issues can occur when the system is integrated later in the design process, at which point it is more difficult and more expensive to make design changes. Often, emergent behaviors are not identified until the integrated system is tested. This can lead to unexpected failure modes and underperformance. In this chapter, the impact of modular architecture on system robustness is considered. That is, are modular systems more or less robust than less modular systems?

4.2 Background

In this section, the literature on two topics will be reviewed. First, modularity and its seemingly paradoxical relationship with robustness will be reviewed. Second, network theory and its use in modeling and analysis in complex engineered systems will be reviewed.

4.2.1 The Paradox of Modularity

High degree of modularity offers a number of benefits in system design [120]. For example, modularity enables the parallel completion of design and produc-

tion [117], reuse of components in other products [120], and change of components over the product life cycle [117]. Additionally, modularity enables failed components to be swapped out for new components, rather than replacing the entire systems. The same is true if there is a design flaw in one module. Finally, modularity enables the management of complexity by dividing a large-scale complex task into smaller, more manageable tasks [117]. These benefits can ultimately save cost [121]. However, there are some concerns regarding the universal applicability of modularity as a design principle. Systems with significant intermodule interactions may require significant design work, negating the benefits of modularity [122]. Further, systems with significant technical constraints have been found to have more integral architectures [121]. This means that it may become necessary to violate modularity when performance is critical [121].

One concern about modularity that is of particular interest in this work is the relationship between modularity and robustness. In robust systems, performance is minimally affected by variation in environmental conditions, faults, or input parameters [123] – all of which are sources of uncertainty that affect a system throughout its life cycle. [124]. Robustness is particularly important in safety critical systems in which significant variations in system behavior can put lives and property at risk; unfortunately, many of these same systems, such as many aerospace systems, are the same systems that benefit from the principle of modularity. Findings on the relationship between modularity and robustness have so far been inconclusive. Some authors argue that modularity increases robustness because faults will be relatively well contained within modules [125, 126], as in the case of cascading failures [127]. Other evidence suggests that modularity decreases robustness [127, 128]. Consider a conceptual example. A targeted attack on a bridging node (connection between modules) is likely to cut off the flow of information entirely between two modules, as in the left hand model in Fig. 4.1. In the right hand model in Fig. 4.1, however, the higher coupling between modules means more connections to absorb a fault [129]. Thus, this conceptual exercise leads to the hypothesis that robustness and modularity are negatively correlated.

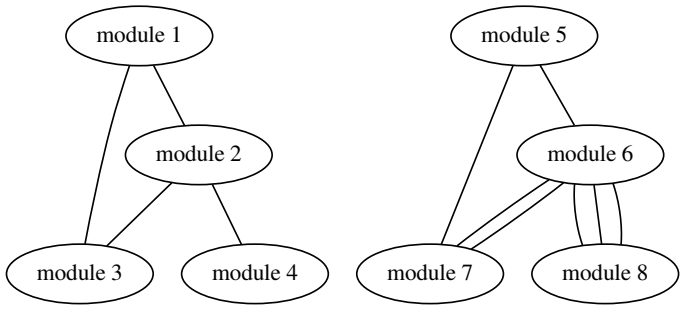


Figure 4.1: Left: conceptual model of a highly modular system. Right: conceptual model of a less modular system. If the flow between module 2 and module 4 were removed, those modules would no longer be connected. However, if the flow between module 6 and module 8 were removed, the modules would still be connected.

Beer's string of pearls analogy can also be used to describe this concept [130]. Pearls strung together in a necklace are vulnerable to failure since breakage of a single link causes the entire necklace to unravel, whereas pearls are arranged in a more integrated structure in which every pearls is connected to every other pearl are more robust since the breakage of a single link would leave the pearls intact [130].

However, existing findings are inconclusive, in part due to their spanning across different disciplines [129, 125] and, perhaps more importantly, due to varying types of perturbations used in the studies [129, 127]. Further, in engineering, failures can be viewed from component, functional, and/or behavioral perspectives. Many studies on modularity in engineering focus on architectural or functional perspectives, whereas many robustness approaches rely on behavioral simulation. Therefore, it can be challenging using conventional approaches to test such a relationship. Thus, it is unclear whether the relationship between modularity and robustness varies based on this modeling perspective.

In addressing this question of modularity and robustenss, this work focuses on a system's response to random injected faults using a network-based approach based

on the developments of the previous chapter. The system's robustness and modularity are measured using various network-based representations of the system. Analysis of robustness and modularity using the same network models enables a consistent and less biased comparison of robustness and modularity as compared to, for example, comparing a system's architectural modularity from a design structure matrix (DSM) to its robustness as computed through a Monte Carlo simulation, which rely on separate underlying models. Three different systems are represented using four different network representations, each representing a different aspect of the system's architecture and behavioral morphology.

4.2.2 Network Theory

This chapter utilizes network theoretic modeling and analysis in order to identify the topological role of vulnerable system elements. Network theory provides a number of powerful computational tools for identifying key features and interactions within such systems. These tools enable efficient analysis of large-scale complex systems. In this section, network theory and the use of networks in modeling and analyzing engineered systems will be reviewed.

The origin of the study of networks is often traced to Euler's famous 1736 solution to the Königsberg bridge problem. As described by Barabási [131], in the city of Königsberg, Prussia (modern day Kaliningrad, Russia), there were seven bridges connecting four land masses separated by a river. At the time, people in the city undertook the challenge of attempting to cross all seven bridges in one walk of the city without crossing any bridge more than once; however, no-one could find such a path across the bridges [131]. Euler, however, realized that the only relevant feature of the problem was the sequence of bridges crossed, rather than the path through the land area, and that the problem could be reduced to what we now recognize as a network, or graph, in which nodes represent land area and edges represent bridges [131]. In the modern language of network thinking, Euler realized that it is the connectivity of the network that is important rather than the properties of the nodes. Based on his network representation Euler realized

that the degree, or number of connections, of the nodes in the network determined whether or not such a walk was possible, and he showed that such a path is only possible if there are exactly zero or two nodes with odd degree [131]. A path through all bridges in the city without repeating a single bridge was impossible unless another bridge were added [131]. Solving this problem in such a manner is often cited as the origin of network thinking.

Subsequent significant developments to network theory included the work of Erdős and Rényi, who proposed the idea of a random graph [132]. In a random graph, all nodes have equal probability of connections. This means that its degree distribution, which is a histogram showing frequency of degrees, or numbers of connections, of all the nodes in the network, has a bell-curve shape. However, very few, if any, truly random graphs are found in nature. Most graphs representing phenomena found in the real world display some degree of clustering. For this reason, other models such as scale-free networks and small world networks are often used. Scale-free networks have a degree distribution following a power law [133]. In small world networks, the distance between nodes is related to the logarithm of the number of nodes in the network [134].

Complex networks are networks with large numbers of nodes arranged into irregular structures, often evolving over time [135]. The structure of complex networks can be characterized by certain network metrics such as node degree, which is the number of connections of a node, and network diameter, which is the longest path between nodes in a network [135]. Complex networks are incredibly useful for modeling, visualizing, and analyzing complex systems due to their representational and computational power, particularly at scale. Subsequently, modeling and analyzing natural phenomena using complex networks has led to insights in various domains including social networks [136], the world wide web [137], and biology [138, 139].

In engineering design, network theory has been used to analyze system modularity [140], predict customer responses to technological changes [141], analyze the effects of design changes [142], and analyze the network structure of product

development processes [143]. Applying network theory to the analysis of complex engineered systems is based on the assumption that complex engineered systems can be modeled as complex networks. The most common justification for this assumption is that there are meaningful similarities between complex networks and complex systems [144]. In particular, there is a relationship between structure and function in complex networks, much as there is in complex systems.

Much as connectivity affects how information can spread through social networks [145], the topological features of engineered systems affect their robustness, modularity, performance, maintainability, and other aspects. This work is particularly focused on the relationship between an engineered system's topological structure and its robustness. Specifically, this work exploits the similarity between the attack tolerance of complex networks and the robustness of engineered systems. This is possible because the structure of networks is relatable to their response to faults and failures, much as some engineered systems are more robust than others. For example, the spread of a contagion through a biological network is much like failure propagation in an engineered system [146]. A common way of representing failure in complex networks is by attacking or removing nodes, which results in the fragmentation of the network into smaller connected components (parts). These attacks can be either random or targeted, where targeted attacks generally focus on high degree nodes. Networks with different structures tend to resist certain types of attack differently [147, 148]. In network modeling of engineered systems, this node removal or attack mirrors component degradation or failure, and the subsequent fragmentation of the network represents loss of functionality.

Previous work has already developed the fundamental concepts and mathematics for representing engineered systems as networks [146, 149, 150]. However, many of the approaches presented thus far have focused primarily on architectural representations of the system, abstracting or neglecting important functional and behavioral relationships. Component architectural models often make varying assumptions as to what designates a connection between components. Typically physical connections count, but architectural models make varying assumptions

about other types of connections, such as heat transfer. In this work, the robustness of network models will be considered from a behavioral perspective as well as an architectural perspective. This will enable the identification of vulnerabilities – the ultimate goal of this robustness research, and the one that directly addresses the research question – from multiple perspectives.

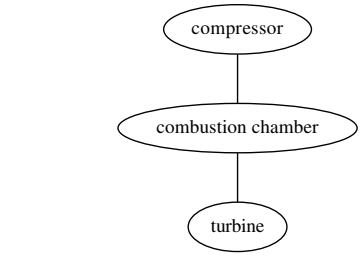
4.3 Methodology

In this work, network-based methods are used to model the systems, analyze their modularity, and analyze their robustness. Similar modeling techniques to those used in Chapter 5 are used in this work. The difference is that in this work, both architectural and behavioral characteristics are modeled individually as well as within the same network. Two metrics – a graph distance based principle and a linear degradation principle – are used in this chapter to analyze the robustness of each network representation.

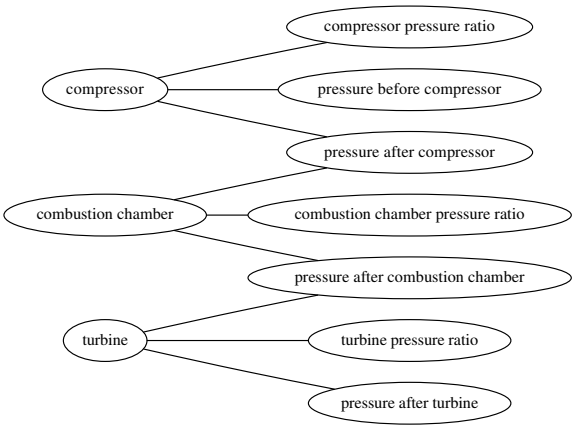
4.3.1 Modeling Complex Engineered Systems as Networks

The systems used in the study are modeled using four different network representations: one purely architectural, one purely behavioral, and two that include aspects of both. The architectural (component) network models the system's components and their connectivity, similarly to in Section 5.3, except with undirected edges. In this network, edges represent physical interfaces between components. The behavioral network models functions and parameters, as in Section 5. The other two types of networks have not yet been introduced in this work. One includes components and parameters. The other includes components and functions. The component network is unipartite (only one type of node), while the other three networks are bipartite (two types of nodes). The bipartite networks are transformed to unipartite-like networks for analysis [18].

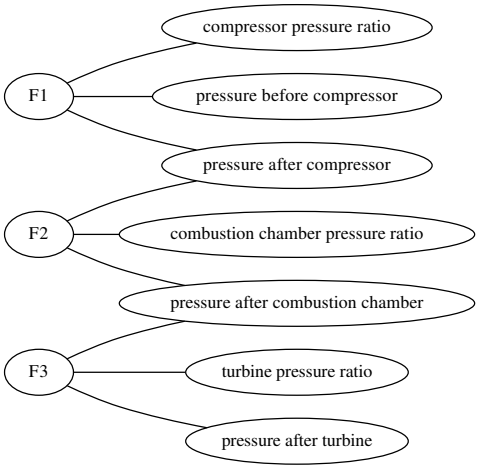
The method of network construction is illustrated using a simplified jet engine model. Three of the main components of the jet engine are the compressor,



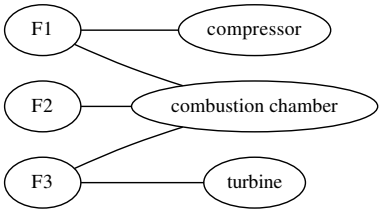
(a) Component network



(b) Component-parameter network



(c) Function-parameter network



(d) Component-function network

Figure 4.2: Network models for simple jet engine example at high degree of abstraction.

	compressor	*combust. ch.*	*turbine*
compressor	0	1	0
combustion chamber	1	0	1
turbine	0	1	0

Figure 4.3: Adjacency matrix for component network. This matrix is square and symmetric since the network is uni-partite.

combustion chamber, and turbine. At a high level of abstraction, the jet engine can be modeled as a network of these three components, as in Fig. 4.2a. In Fig. 4.2a, each component is a node. These nodes share an edge if there is a physical interface between their corresponding components, such as the connection between the compressor and combustion chamber. In the bipartite component-parameter network, nodes can represent either components or parameters. For instance, the compressor pressure ratio is a property of the compressor, so the compressor node and compressor pressure ratio node are connected. As in all bipartite networks, nodes of one type may only connect to nodes of the other type. A simple example of the component-parameter network is given in Fig. 4.2b. In the function-parameter network, Fig. 4.2c, functions and parameters are modeled as nodes. Parameter nodes are connected to function nodes if they are involved in the completion of that function. In the given example, the functions are changing the pressure across each component. In each case, domain knowledge is used to build the first three networks.

The final network, the component-function network given in Fig. 4.2d, can be obtained by multiplying the adjacency matrix of the component-parameter network and the inverse of the adjacency matrix of the function-parameter network. Adjacency matrices for networks show the node connections in matrix form. Adjacency matrices for unipartite networks are square and symmetric, as in Fig. 4.3. These matrices have dimensions $n \times n$ where n is the number of nodes in the network. Bipartite networks, in contrast, have rectangular adjacency matrices of size

$$
\begin{array}{c}
 \\
\textit{compressor pressure ratio} \\
\textit{pressure before compressor} \\
\textit{pressure after compressor} \\
\textit{combustion chamber pressure ratio} \\
\textit{pressure after combustion chamber} \\
\textit{turbine pressure ratio} \\
\textit{pressure after turbine}
\end{array}
\begin{array}{c}
\begin{array}{ccc} F1 & F2 & F3 \end{array} \\
\begin{pmatrix}
1 & 0 & 0 \\
1 & 0 & 0 \\
1 & 1 & 0 \\
0 & 1 & 0 \\
0 & 1 & 1 \\
0 & 0 & 1 \\
0 & 0 & 1
\end{pmatrix}
\end{array}
$$

Figure 4.4: Adjacency matrix for function-parameter network. This matrix is rectangular since the network is bi-partite.

$n \times m$, where n is the number of nodes of one type and m is the number of nodes of the other type, as in Fig. 4.4. To generate the component-function network, the component-parameter adjacency matrix (dimensions $C \times P$, where C is the number of components and P is the number of parameters) is multiplied by the inverse of the function-parameter adjacency matrix (dimensions $P \times F$, where F is the number of functions) in order to obtain an adjacency matrix with dimensions $C \times F$, the component-function matrix. In the case that some of the resulting matrix elements are greater than one, which sometimes occurs when, for instance, a component is associated with two parameters, these values are reduced to 1. Using this method, the generation of the fourth network can be automated using knowledge of two other networks.

4.3.2 Q-Modularity

Modularity metrics are typically applied to component-based representations of system architecture. Examples include Whitney Index (WI), Change Cost (CC), and Singular Value Modularity Index (SMI) [151]. WI considers the number of interactions among elements of the system [151], whereas CC uses a visibility or reachability matrix to measure the percentage of modules affected by a change in

one modules [151]. SMI is computed from the design structure matrix (DSM) and considers singular value decay structure [151, 152]. More recently developed measures include metrics that consider the distribution of complexity in a system [153]. In complex networks, Q-modularity is typically used. Q-modularity considers the strength of intra-module connections and of inter-module connections in determining modularity, rather than the number of modules in the network. In networks, modules are sometimes called communities. Modules represent groupings of nodes that generally have emergent properties. In the network-based models of engineered systems used in this research, a module can be interpreted as a component or functional sub-unit, depending on the specific network representation.

The process for computing Q-modularity begins with the identification of modules in the network, which is typically done using a community detection algorithm. Modules are identified using the approach outlined in Section 5.2.2. Modularity can then be computed using Eq. 5.16. Q-modularity is a particularly useful measure of modularity for this research because it can be used on various different network representations of systems (architectural, functional, behavioral, etc.).

4.3.3 Network-Based Measures of Robustness

Typical approaches to robustness analysis include measuring the likelihood of fulfilling intended functionalities even in the presence of external perturbations [154]. These approaches tend to involve the representation of variation in input parameters and quantifying the effect of this variation on system outputs [154]. One example of such an approach is a Monte Carlo simulation in which standard deviation of the output distribution is used to measure robustness [155]. Other approaches include measuring the probability of fulfilling intended product functions in the presence of input variation [154]. Such approaches indicate the probabilities of fulfilling certain functions [154, 156].

Recent advances in network-based representations of complex engineered systems have measured robustness in using a conceptually different approach. Robustness is generally defined as a property of systems with little variation in perfor-

mance even in the presence of faults or other variation in input parameters [123]. That is, performance variation that results from faults is relatable to robustness. Robust systems will continue to perform, at least to some degree, even if, for instance, a fault reduces the coefficient of friction of a car tire. In network science, degradation of performance of a part is understood as node attack or removal. In systems, when the performance of a part is degraded, the performance of the system is degraded as well. For example, when a car tire is worn, the handling capabilities of the car may worsen. Similarly, in networks, when a node is attacked, it affects the topological structure of the network. Conceptually, the network's resistance to attack is analogous to the automobile's ability to maintain its performance, to some degree, even in the presence of faults. This loss of performance even after a fault, which is related to the system robustness, is captured using network topological concepts. On the other hand, other types of failures are better captures using cascading failures, such as a worn tire causing alignment problems, which causes problems with the steering mechanism.

4.3.3.1 Average Shortest Path Length

Certain network metrics can be used to measure network robustness. The first, average shortest path length (ASPL), is an *a priori* measure of network robustness. That is, it can be used to assess a network's resistance to attack without directly simulating network attack. ASPL is the mean of the shortest distances between all pairs of nodes in the network. Networks with low ASPL tend to be more robust [157] because nodes are still reachable within a relatively short distance even if some paths are attacked. In the string of pearls example, the robust string with every pearl connected to every other pearl has an ASPL equal to one, whereas the standard string has an ASPL greater than one (assuming the string has greater than two pearls). ASPL is defined in Eq. 4.1 [158], where N is the number of nodes and d is the shortest distance between nodes i and j as identified using Dijkstra's algorithm. This method of measuring robustness is efficient and can be used in any of the networks presented. However, note that the bipartite networks

are treated as unipartite-like networks in computing this metric [18].

$$ASPL = \frac{1}{N^2} \sum_{ij} d_{ij} \tag{4.1}$$

4.3.3.2 Robustness Coefficient

The second network-based measure of robustness involves the simulation of node attack. Specifically, robustness coefficient (RC) measures the changing size of the largest connected component in the network as nodes are systematically attacked [125]. In robust networks, the largest connected component decreases in size more slowly than in less robust networks. Robustness coefficient is computed according to Eq. 4.2, where S is the size of the largest connected component after k nodes have been attacked. RC falls between 0 and 100.

$$RC = \frac{200 \sum_{k=0}^{N} S_k - 100\, S_0}{N^2} \tag{4.2}$$

4.3.4 Systems Studied

Three systems are used in this study: a bicycle drivetrain, an automobile drivetrain, and an aircraft. These particular systems are selected in order to include various system sizes (see Table 4.1), degrees of compelxity, and architectures. The four different network models are generated for each system. Characteristics of these networks are summarized in table 4.2. Network diameter is the longest of the shortest paths between all nodes in the network. To study whether each network contains a sufficient degree of granularity, a sensitivity analysis is performed in order to test the sensitivity of the network's Q-modularity to random node deletion. These results are summarized in Table 4.3. The small modularity changes, especially for the larger models, indicate that an acceptable degree of granularity is used in the modeling process. The bicycle model is more sensitive to node deletion, which is expected since it is a smaller network overall. The degree distributions

Table 4.1: Sizes of systems used in case study. Number of nodes in each network can be derived from this information. For example, the aircraft function-parameter network has 34 function nodes plus 81 parameter nodes for a total of 115 nodes.

	Components	Functions	Parameters
Bicycle Drivetrain	10	3	8
Automobile Drivetrain	19	8	23
Aircraft	375	34	81

Table 4.2: Network characteristics.

	Component	Component -parameter	Function -parameter	Component -function
Minimum degree				
Bicycle	1	1	1	1
Automobile	1	1	1	1
Aircraft	1	3	1	17
Maximum degree				
Bicycle	3	4	4	8
Automobile	6	7	5	10
Aircraft	58	375	13	375
Network diameter				
Bicycle	7	11	6	4
Automobile	8	7	10	5
Aircraft	18	4	10	4

of the networks (histograms of node degrees) are also analyzed in Fig. 4.5. The variety of characteristics in the dataset reduce the probability of bias in the results. All networks are visualized in Fig. 4.6.

The first system considered, and the simplest one, is a bicycle drivetain. This model is the smallest at 10 components, 8 parameters, and 3 functions, as given in Table 4.4. The second system is larger at 19 components, 23 parameters, and 8 functions. This is an automobile drivetrain, presented in work by Haley et al. [18]. The third model is the largest – a high-level model of an aircraft in steady, level flight. The models for this system are the most detailed at 375 components,

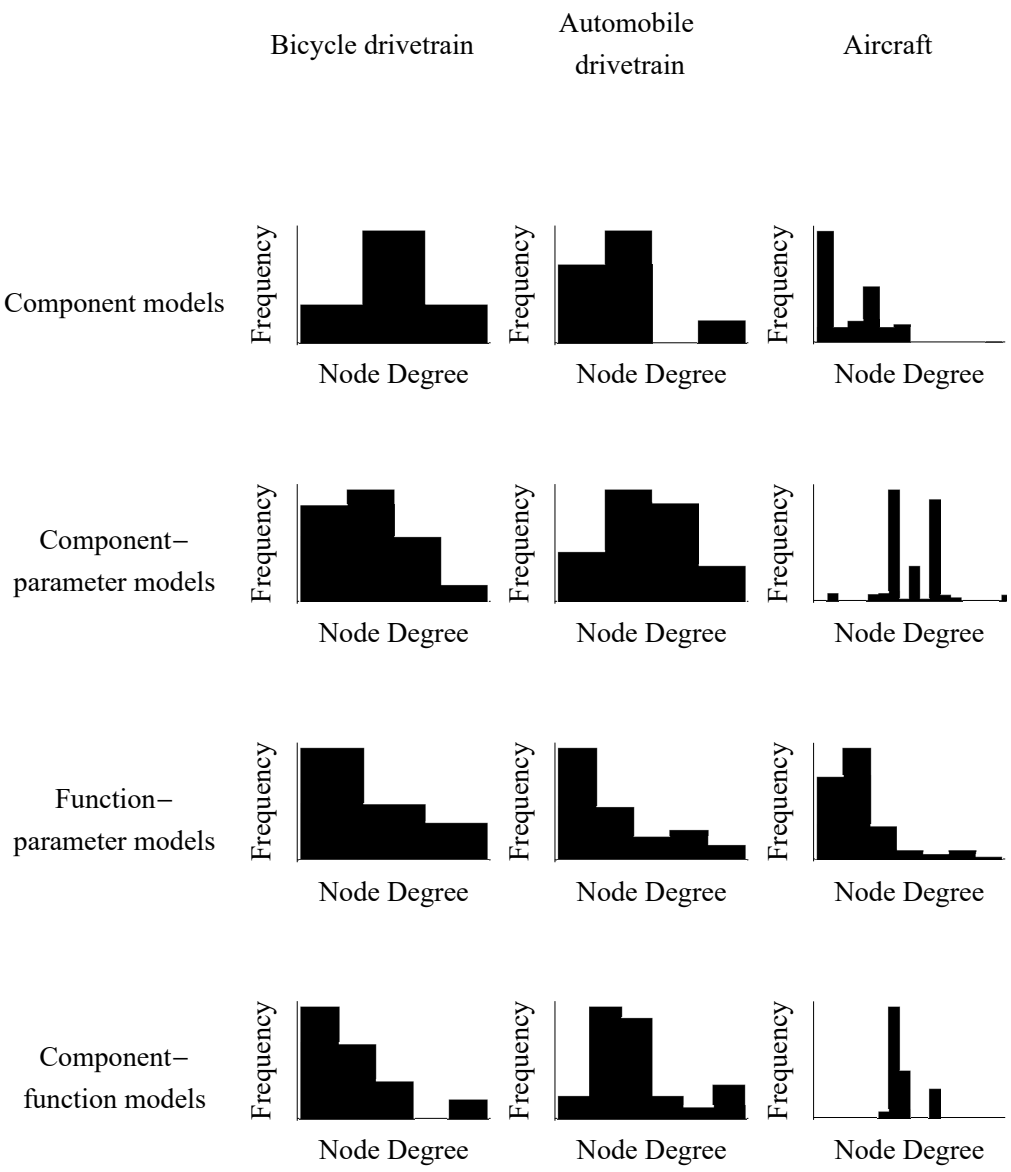


Figure 4.5: Degree distributions for each network model used in the case study. Each column shows different network representations of the same system. Each row shows different systems represented using the same combination of component, parameter, and function information.

Table 4.3: Percent change in Q-modularity due to node removal, averaged over independent removal of each node in the network. Low percentages show that the Q-modularity estimates are not significantly impacted by choice of model granularity.

	Component	Component -parameter	Function -parameter	Component -function
Bicycle	17%	3%	11%	15%
Automobile	3%	1%	2%	3%
Aircraft	$< 1\%$	$< 1\%$	2%	$< 1\%$

Table 4.4: Components, functions, and parameters for the bicycle model.

Components	Functions	Parameters
Pedal	F1: $F_{crank} = \frac{F_{applied} \times R_{crankarm}}{R_{chainring}}$	F_{crank}: force from crank arm
Crank arm	F2: $F_{cogset} = F_{crankarm}$	$F_{applied}$: force applied to pedal
Chain	F3: $F_{propel} = \frac{F_{cogset} \times R_{cogset}}{R_{rearwheel}}$	$R_{crankarm}$: length of crank arm
Chain rings		$R_{chainring}$: radius of chain ring
Cogset		F_{cogset}: force from cogset
Rear derailleur		R_{cogset}: radius of cogset
Rear spokes		$R_{rearwheel}$: radius of rear wheel
Rear hub		F_{propel}: force propelling bicycle
Rear rim		
Rear tire		

81 parameters, and 34 functions. Rib and spar connectivity is determined using work by Perry as reference [159]. The structural arrangement of the fuselage is determined using Roskam [160] and Gudmundsson [161]. In the behavioral model, thrust is equal to drag and lift is equal to weight for steady, level flight. The equations for lift, weight, thrust, and drag are as described by Roskam [162].

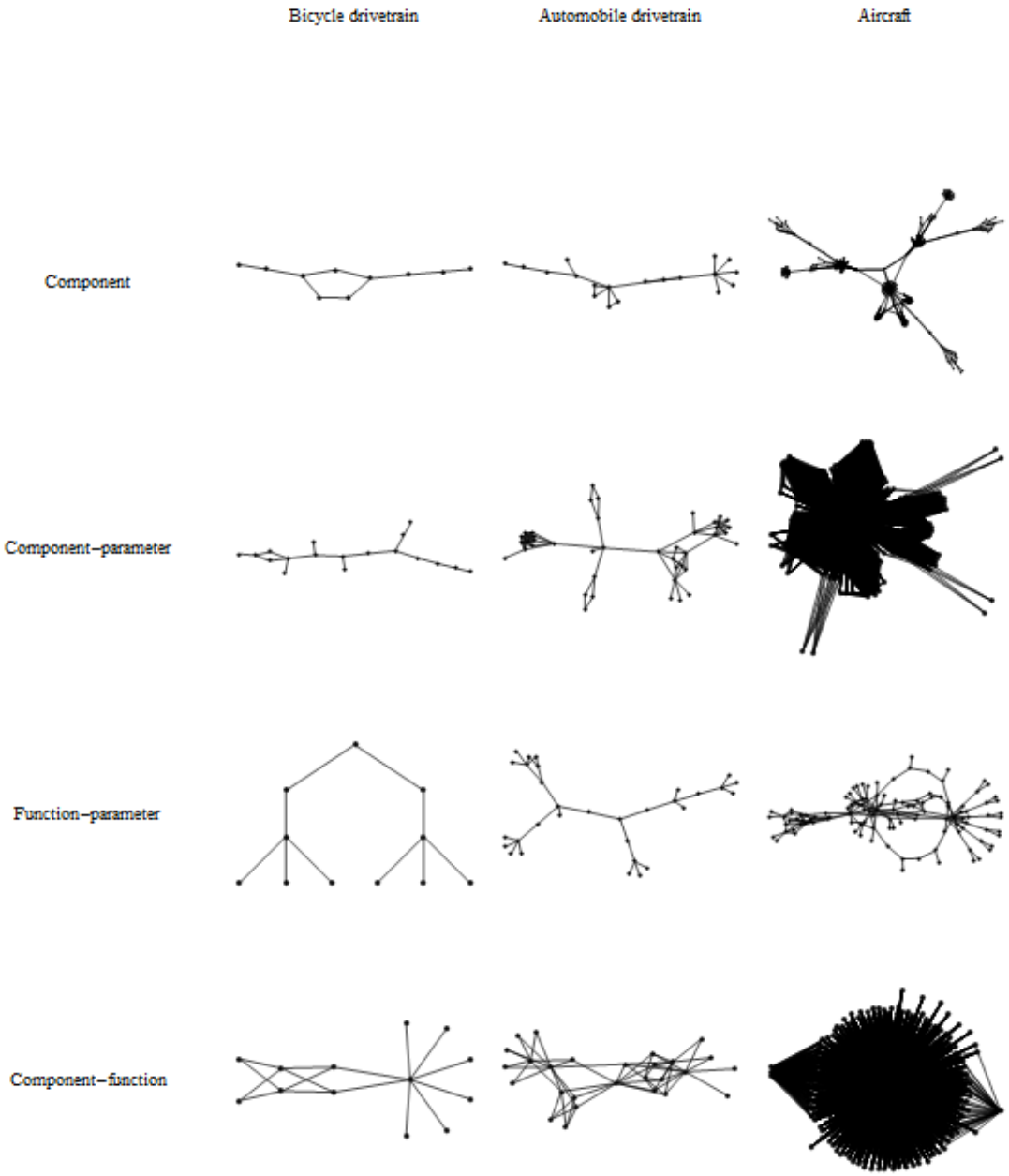


Figure 4.6: All twelve models used in the study. Each column shows different network representations of the same system. Each row shows different systems represented using the same kind of network.

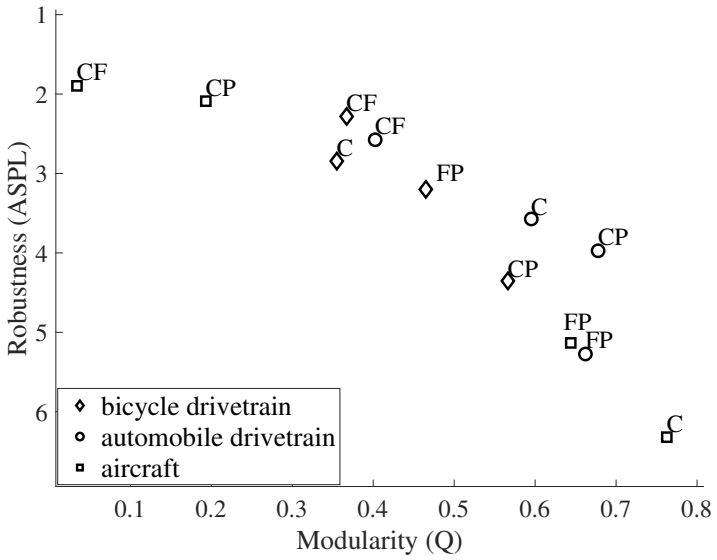


Figure 4.7: Plot of ASPL versus modularity showing a negative correlation. Higher modularity is associated with lower robustness (higher ASPL). Points are labeled by network kind: C (component), CF (component-function), CP (component-parameter), and FP (function-parameter).

4.4 Results and Analysis

The Q-modularity and robustness of the sampled systems are plotted in Fig. 4.7, indicating a positive correlation between modularity and ASPL with a linear correlation coefficient $R = 0.8869$ ($p < 0.001$). A quadratic model also fits the data with $R^2 = 0.8910$. Since high ASPL means lower robustness, this result is interpreted as a negative correlation between modularity and robustness. Since this result is found using four different network models and three systems, it is unlikely to be exclusive to any of the types of networks used in this study. There is also a negative correlation between modularity and robustness when RC is used instead of ASPL ($R = -0.7733$, $p = 0.003$), as in Fig. 4.8.

Interestingly, Table 4.5 indicates that modularity estimates are significantly different for the different network models of the same system. This finding sug-

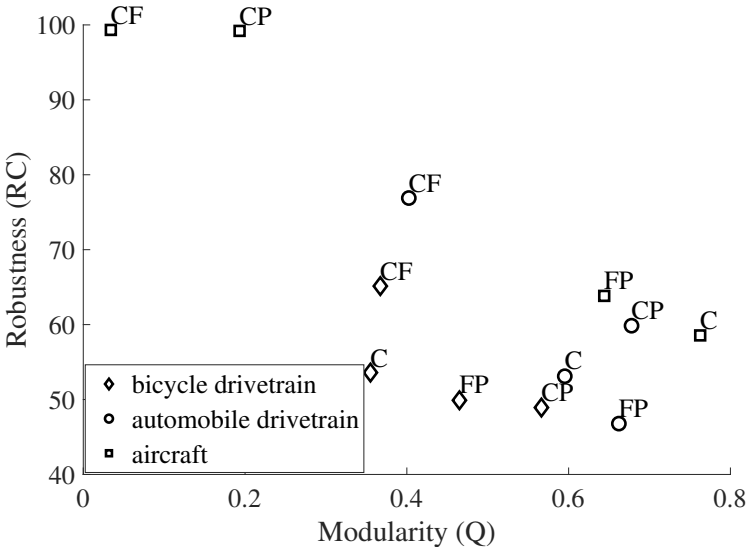


Figure 4.8: Plot of RC versus modularity showing a negative correlation. Points are labeled by network kind: C (component), CF (component-function), CP (component-parameter), and FP (function-parameter).

Table 4.5: Q-modularity in each model. Larger difference in Q-modularity signals that the error in modularity estimation is higher.

	Component	Component -parameter	Function -parameter	Component -function	Max difference
Bicycle drivetrain	0.3550	0.5663	0.4650	0.3671	0.2113
Automobile drivetrain	0.5952	0.6777	0.6621	0.4024	0.2753
Aircraft	0.7626	0.1932	0.6439	0.0340	0.7286

gests that considering only a single perspective (architectural, functional, etc.) in modularizing a system may lead to under- or overestimates of system modularity. For instance, the component-parameter network is more modular than the component network in both the bicycle and automobile. However, in the aircraft, the component network is more modular than the component-parameter network. This is likely due at least in part to a small number of very high degree nodes in the aircraft component-parameter network, namely the load factor (the lift to weight ratio). Because every component experiences stress due to load factor, this node has a degree of 375, significantly reducing the modularity of the network. Systems with large differences in modularity between models may present problems in modularizing the system. For instance, if the behavioral model is much more integrated than the component model, behavioral interactions may complicate interfaces between modules, reducing the benefits of modularity.

The finding that modularity is negatively correlated with robustness indicates a design trade-off between the benefits of modularity and robustness. Further, the findings on the differences in modularity between network models may indicate that a system is "not as modular as they think" due to behavioral or functional interfaces neglected by component-based assessments of modularity. This is a particularly interesting considering that it is often desirable for there to be a one-to-one mapping between structure and function [163]. The findings in this study indicate that this one-to-one mapping may be difficult to attain, particularly for systems with higher complexity. Importantly, these empirical results relate only to the robustness of systems to random faults. They do not apply to systems prone to cascading failures. Previous studies have indicated a possible connection between cascading failures and modularity [126, 146] in that modular systems are more resistant to cascading failures. This is likely because in highly modular systems, faults are contained in modules. Further work is required to verify this relationship between system modularity and other types of failure such as cascading failure. Further, it should be noted that even in highly modular systems, it is still possible to attain highly reliable systems using rigorous maintenance and highly

proceduralized operation, as well as redundancy. These factors are not considered in the robustness measures in this research. This means that even highly modular systems can still be very safe.

4.5 Chapter Summary

This chapter explores the use of network metrics to examine the benefits and drawbacks of modularity as a design principle, specifically as it relates to system robustness. The new findings indicate a trade-off between modularity and robustness in engineered systems. Further, the findings indicate discrepancies between behavioral and architectural modularity, potentially complicating processes for modularizing systems. Thus, the benefits of increasing a system's modularity should be carefully weighed against desired robustness and required health management systems, and modularization should not be considered a universal design principle. Given the usefulness and ubiquity of certain design principles, it is essential to examine their rationality. Understanding the use case of design principles can help designers create safer and more innovative systems in the future.

The new finding that modularity influences robustness implies that the interfaces between modules are especially important to robustness analysis. When there are more variables that interface between modules, there are more pathways for information to travel between modules from a network perspective. These variables serve as essential pathways for information spread – and for maintaining performance even when there are faults present. This leads to the hypothesis that certain system elements, or nodes from a network perspective, are more significant contributors to fragility than others, based on their structural role in the system. This is investigated in the following chapter.

Chapter 5: Identifying Vulnerable Elements in Complex Engineered Systems

This chapter addresses the first research question: what is the topological role of the variables that are key to controlling a system's robustness to failure? To this end, the behavioral aspects of the system – i.e. its governing equations – are modeled and analyzed as an undirected network in order to identify critical variables. This research was published in Design Science and in the Proceedings of the 2017 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference and was cowritten by Hannah S. Walsh, Andy Dong, and Irem Y. Tumer [164, 165]. As a further elaboration of this research question, a methodology is also proposed for identifying structurally important nodes in a particular system architecture, Structural Consequence Analysis (SCA). This proposed method is effectively a more accessible version of a fault tree that also takes into consideration the cost of component failures and can be implemented earlier in the design process. This research was published in the Proceedings of the 2019 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference and was cowritten by Hannah S. Walsh, Mohammad Hejase, Daniel Hulse, Guillaume Brat, and Irem Y. Tumer [166].

5.1 Motivation

System wide vulnerabilities can be difficult to detect in large-scale, complex engineered systems. Obtaining accurate behavioral models for the purposes of performing a sensitivity analysis is difficult in complex systems due to under-recognized vulnerabilities. Specifically, these approaches tend to struggle to capture emergent behavior that occurs due to the integration of different modules that are often designed and modeled independently. Instead, this research utilizes network

modeling to identify vulnerabilities, effectively using abstraction to study vulnerabilities in systems. These vulnerabilities can be confirmed by observing patterns in large quantities of network-based models of systems.

In this chapter, a behavioral network approach is used to identify vulnerabilities in engineered systems. Specifically, general topological vulnerability patterns are identified in behavioral network models of complex engineered systems. Behavioral networks represent the governing equations of a system using a network. This representation, proposed in prior work [167, 18], is used to study robustness by relating the local failure of nodes to the network's structural degradation. Robustness in this work is conceptualized as the system's resistance to high-level behavioral degradation, measured by the topological fragmentation of the behavioral network, caused by faults injected in parameter nodes. This work utilizes the modeling and robustness analysis approaches of this prior work in order to identify bridging nodes as a generalized pattern of vulnerability in behavioral network models of engineered systems. In other words, the hypothesis of this work is whether the behavioral degradation of complex engineered systems is correlated with faults in bridging nodes.

To test this hypothesis, an experimental study is constructed in which forty engineered systems are modeled as behavioral networks. Behavioral networks are selected in this study because, as shown in the previous chapter, more conventional architectural networks do not always fully capture a system's morphology. A network robustness metric, average shortest path length (ASPL), is measured when faults are injected into bridging and non-bridging nodes in each of these systems. Fault injection is performed by adjusting the weights of the edges associated with the node into which a fault is being injected, a form of node attack comparable to the degradation of performance of the relevant parameter. In order to perform this experiment, existing modeling approaches are also extended in order to describe embedded behaviors and logical behavior.

5.2 The Role of Bridging Nodes in Behavioral Network Models of Complex Engineered Systems

Determining system sensitivity to variation in component performance in early design aids in planning for mitigation strategies such as redundancy or health management systems before major design decisions are made, ultimately saving cost and time. This assessment can be challenging, however, especially in complex systems. This section utilizes network based modeling and analysis of complex engineered systems to enable computationally efficient analysis in early design. While network models are ultimately relatively abstract models of the system, they enable the analysis of the topological structure and behavioral morphology of the system without the need for computationally expensive sensitivity analyses. Additionally, their abstraction enables the analysis of large scale models – that is, the entire system can be analyzed in one model, rather than relying on separate subsystem-level models. This enables the assessment of emergent behavior that may arise due to interaction between subsystems. This chapter results in the identification of bridging nodes as vulnerable system elements. Knowledge of bridging nodes can be used as an *a priori* assessment of vulnerability during system design.

5.2.1 Conceptual Proof

In the previous chapter, it was shown that modularity is negatively correlated with robustness. This finding implies that the interfaces between modules may be particularly influential in controlling a network's robustness. Thus, in this chapter, the role of bridging nodes in controlling a system's robustness is investigated. Before presenting the experiment, a conceptual proof will be used to show how network attacks (changes in edge weights associated with a node, i.e. fault injection) on bridging nodes tend to induce larger changes in ASPL than in non-bridging nodes. The definition of ASPL is the average of all of the distances (shortest paths) between each pair of nodes in a network. ASPL is computed according to Eq. 5.1 [158], in which n is the number of nodes in the network, d is the shortest

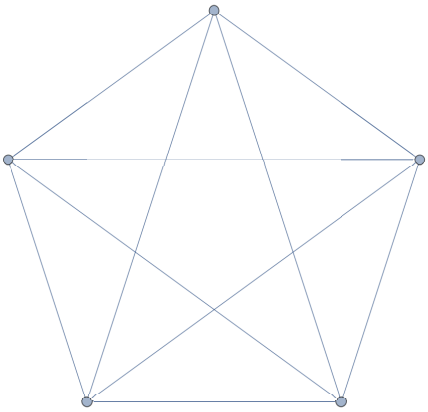


Figure 5.1: Small example network with ASPL = 1.

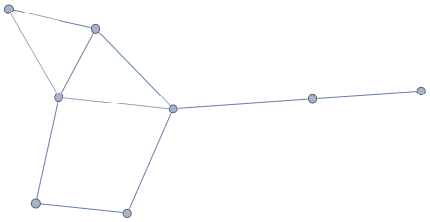


Figure 5.2: Small example network with ASPL = 2.

distance between two nodes, and the indices i and j refer to individual nodes. Distance between a pair of nodes is computed by adding the edge weights of all edges along the shortest path between the nodes. This concept is visualized in Fig. 5.1 and Fig. 5.2. In Fig. 5.1, the distance between each pair of nodes in the network is one, i.e. all nodes are neighbors of each other. In contrast, in Fig. 5.2, some pairs of nodes have longer distances between them, resulting in a higher ASPL.

$$ASPL = \frac{1}{n^2} \sum_{ij} d_{ij} \tag{5.1}$$

Given a pair of nodes i and j in a network, if the distance between node i and node j decreases, the ASPL will decrease as well. In other words, local changes to edge weights are detectable with ASPL. Specifically, given an attack that decreases the edge weights associated with a node, the ASPL for a network under attack is lower than the nominal network. Consider the changes in ASPL for attacks on bridging and non-bridging nodes. Bridging nodes are nodes that connect communities, i.e. have an edge between communities. Reducing the weight of an edge between communities will shorten the paths between all nodes in these communities, reducing the ASPL significantly. In comparison, a reduction in edge weight between two nodes within a community is unlikely to have the same effect.

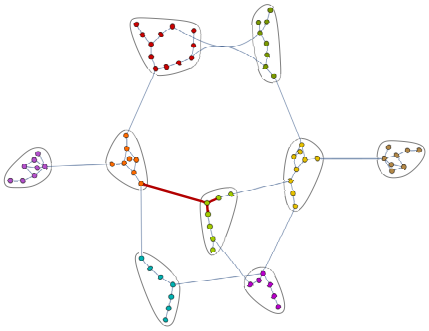


Figure 5.3: Behavioral network for voltage divider circuit with bridging node highlighted.

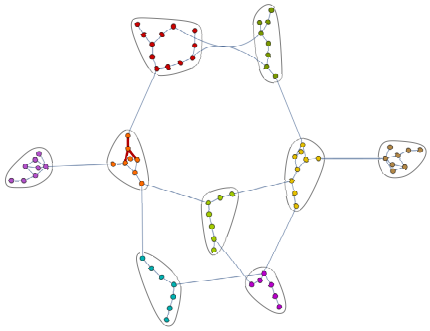


Figure 5.4: Behavioral network for voltage divider circuit with non-bridging node highlighted.

This comparison is visualized in Fig. 5.3–5.4. The nominal ASPL of the network (all edge weights equal to one) is equal to 8.059. Injecting a fault into a bridging node (reducing the weights of its edges to 0.5) reduces the ASPL to 7.796. In contrast, when a fault is injected into a non-bridging node, the ASPL is reduced to 7.819. Therefore, a fault injected into a bridging node has a larger effect on the degradation of network structure (and by extension the system behavior, because this is a behavioral network) than a fault injected into a non-bridging node. In the next section, a full experimental study will test this relationship with more nodes and in larger networks.

5.2.2 Methodology

The methodology is summarized as follows. First, a set of forty engineered systems are identified and modeled as behavioral networks. Second, the engineered systems are modeled as behavioral networks. Third, for each network, each node is injected with a fault and the resulting change in ASPL is measured. Fourth, in each network, bridging nodes are identified. Finally, an independent samples t-test with unequal variance is performed in order to test the hypothesis that bridging nodes result in a higher change in ASPL when in a fault state than non-bridging nodes. Each step will be detailed in the remainder of this section.

5.2.2.1 Description of Systems Used

The systems are originally modeled in OpenModelica [168] in order to represent their behavior. The forty systems used ranged in size from 33 to 1732 edges in their behavioral network representation, as shown in Fig. 5.5. The systems represent a range of different disciplines, including electrical, mechanical, fluid/heat transfer, and magnetic, as in Table 5.2. Of forty total systems, thirty-eight are example models from OpenModelica [168], one is a simple voltage divider circuit, and one is synthetically generated. Table 5.1 provides high-level descriptions of each of the forty systems. Additionally, degree distribution plots for four of the behavioral networks, one from each disciplinary category, are given in Fig. 5.6a–5.7b. Degree distribution plots are histograms of the degree of nodes in the network and reveal topological characteristics of networks and can be used to identify whether a network is scale-free, random, or regular. All four networks roughly follow the distribution of a homogeneous network, indicating that the networks are not particularly vulnerable to targeted attacks (attacks on vulnerable nodes).

Table 5.1: Descriptions of systems used in Section 5.

	System Name	Number of Edges	System Description
1	Electrical analog rectifier	335	B6 diode bridge, three-phase sinusoid voltage, and DC current load
2	Simple triac circuit	263	Simple triac used in alternating current circuit
3	AIMC DOL	1007	Asynchronous induction machine, squirrel cage, and direct on line starting
4	AIMC inverter	878	Asynchronous induction machine, squirrel cage, and ideal inverter
5	AIMS start	1617	Asynchronous induction machine, slipring rotor, and resistance starting

Continued on next page

Table 5.1 – *Continued from previous page*

	System Name	Number of Edges	System Description
6	AIMC Steinmetz	943	Asynchronous induction machine, squirrel cage, and Steinmetz connection
7	AIMC transformer	1732	Asynchronous induction machine with squirrel cage, transformer supplies three-phase voltage
8	AIMC YD	1189	Asynchronous induction machine and squirrel cage, Y-D starting
9	SMEE generator	1246	Excited synchronous induction machine used as a generator
10	SMEE load dump	1459	Excited synchronous generator, loaded with generator
11	SMEE rectifier	1203	Excited synchronous generator, loaded with rectifier
12	SMPM current source	1280	Synchronous induction machine with permanent magnets fed by current source
13	SMPM inverter	1092	Permanent magnet synchronous induction machine and ideal inverter
14	SMPM voltage source	1435	Synchronous induction machine with permanent magnets fed by voltage source
15	SMR inverter	1064	Synchronous induction machine, reluctance rotor, and ideal inverter
16	Multiphase rectifier	528	Diode bridge rectifier with star-connected voltage source, line reactor, and DC burden
17	Multiphase test sensors	796	Sinusoid source loaded with resistor and inductor

Continued on next page

Table 5.1 – *Continued from previous page*

	System Name	Number of Edges	System Description
18	Transformer YD	675	Y-D transformer with star-connected voltage source and load resistor
19	Transformer YY	665	Y-Y transformer with star-connected voltage source and load resistor
20	Cascode circuit	188	JFET cascode circuit
21	Electrical oscillator	341	Oscillator circuit with BJT transistors
22	Synthetic system	173	Test system with multiple mechanical blocks
23	Heat flow, one mass	187	One hot mass cooling
24	Indirect cooling	435	Heat source dissipates heat with thermal conductor and inner coolant cycle
25	Saturated inductor	185	Inductor with a saturated ferromagnetic core
26	Accelerate	33	Demo moving a mass with predefined acceleration
27	Grounded drive train	117	Drive train with motor inertia, motor torque, and grounded elements
28	Preload	318	Preload spool for hydraulic valve
29	Rolling wheel	84	Rolling wheel demonstrating coupling between rotational and translational components
30	Sensors	84	Demo of sensors used for translational systems
31	Simple drive train	122	Drive train with motor inertia and motor torque
32	Elasto gap	191	Demo model with elasto gap, springs, and dampers

Continued on next page

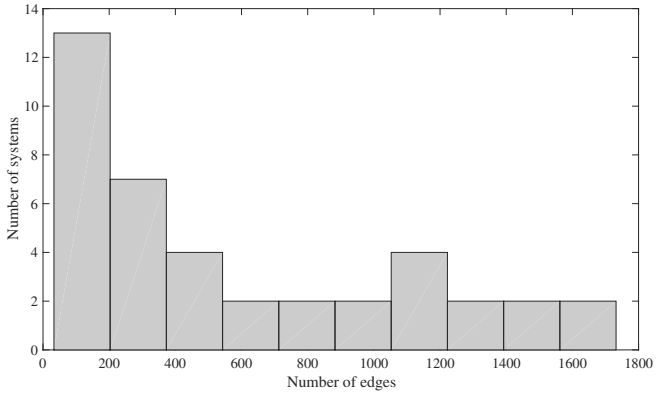


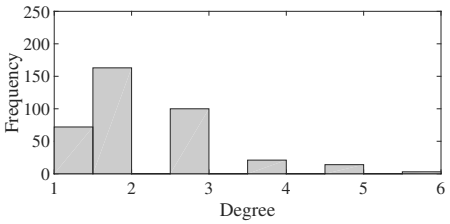
Figure 5.5: Sizes of systems studied.

Table 5.1 – *Continued from previous page*

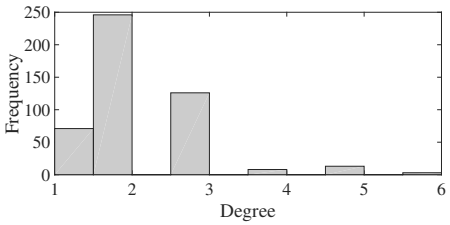
	System Name	Number of Edges	System Description
33	Parallel cooling	383	Cooling circuit, parallel branches
34	Pump and valve	280	Pump and valve cooling circuit
35	Pump drop out	203	Drop out of pump cooling circuit
36	Parallel pump drop out	383	Drop out of pump cooling circuit with parallel branches
37	Simple cooling	203	Heat source dissipates heat with a thermal conductor, coolant flow, and pump
38	Controlled temperature	121	Demo of controlling temperature of resistor
39	Heat transfer, two masses	48	Conduction between two mass elements
40	Voltage divider	82	Voltage divider circuit

Table 5.2: Characteristics of systems

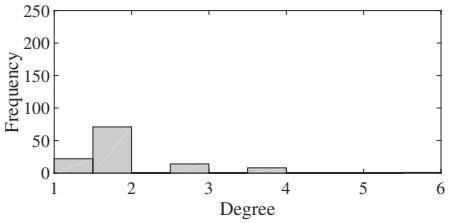
System Category	Number of Systems
Mechanical	8
Fluid/Heat Transfer	9
Magnetic	1
Electrical	22



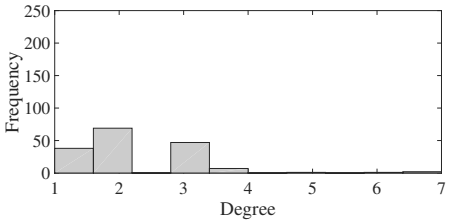
(a) Degree distribution plot for behavioral network of indirect cooling system.



(b) Degree distribution plot for behavioral network of electrical multiphase rectifier system.



(a) Degree distribution plot for behavioral network of simple drivetrain system.



(b) Degree distribution plot for behavioral network of magnetic saturated inductor system.

5.2.2.2 Modeling Techniques

Behavioral information from OpenModelica is used to build behavioral networks of each system. In general, it is also possible to use behavioral models from SysML and Simulink rather than OpenModelica. When using OpenModelica, behavioral information is obtained from the model instantiation information, which contains the governing equations of the system. This information is exported to a text file and processed with a text processing script in MATLAB in order to extract key information about the functions and parameters of the system, which are used to build the behavioral network. The nodes in a network behavioral represent functions and parameters. The network is bipartite, such that functions and parameters are represented as different node types. In such a bipartite network, functions may only connect to parameters and parameters may only connect to functions via their edges. This modeling approach, detailed next, is based on the approach proposed by Haley et al. [167, 18] and is extended in this work to include a broader range of behaviors.

Standard Modeling Procedure Each equation in the set of governing equations obtained from the OpenModelica instantiation information is first assigned a numbered function, e.g. $F1$–$F10$ for a set of ten functions. Next, each parameter in a given equation is assigned as a parameter node, which shares an edge with its corresponding function node. For instance, consider the example of the first two functions from one of the systems used in the study, a rolling wheel. The first two functions are $F1$ and $F2$, as provided in Eq. 5.2–5.3. By identifying the parameters in Eq. 5.2–5.3, it is determined that there are four parameter nodes connected to $F1$ and two parameter nodes connected to $F2$, as shown in Fig. 5.8.

$$F1 : inertia.J \times inertia.a = inertia.flange\_a.tau + inertia.flange\_b.tau; \quad (5.2)$$

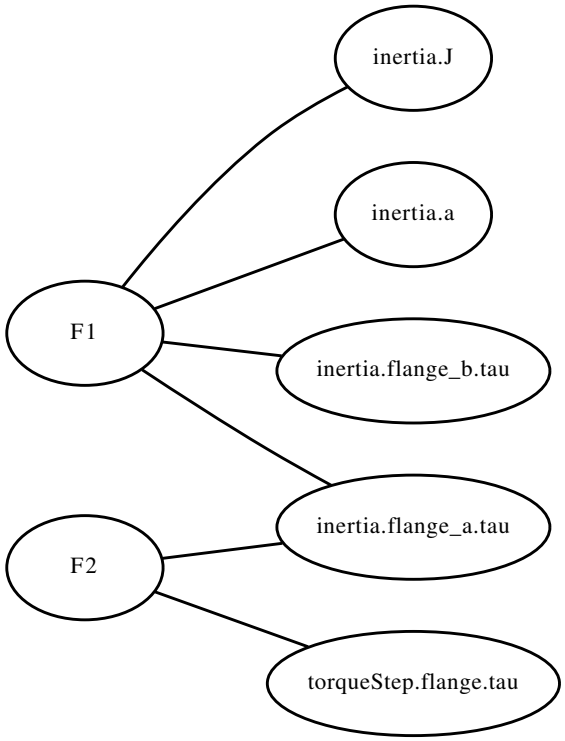


Figure 5.8: Network segment for basic BNA technique example from rolling wheel system for Eq. 5.2–5.3.

$$F2 \quad : \quad inertia.flange\_a.tau \; + \; torqueStep.flange.tau \quad = \quad 0.0; \quad (5.3)$$

During the process of generating behavioral networks of the forty systems, it is discovered that there are some behaviors that are not well-defined by existing modeling techniques. Specifically, embedded behaviors and logical behavior, which have specific mathematical manifestations in the OpenModelica instantiation information, are not well-defined by this process. Modeling them correctly is critical to understanding the real behavior of these systems.

Modeling Embedded Behavior Embedded behaviors are represented mathematically with a function call. In the systems studied, the function calls are sometimes used for a computation that is performed several times, or for a sub-

system. Since embedded functions can sometimes be called several times, each set of function outputs is unique, and should therefore be given a unique identifier. For example, within a system's set of governing equations, Eq. 5.4–5.8 define a function and that function is called in Eq. 5.9. The two outputs are y and $y0$, but these outputs are dependent on the values of inputs used, and it is possible that those inputs are different each time the function is called. So, y and $y0$ must be given unique identifiers each time the function is called. Each instance of the function call is also given a new function node, for example $F2$ in Fig. 5.9.

$$function\ ToSpacePhasor \tag{5.4}$$
$$input\ Real[3]\ x; \tag{5.5}$$
$$output\ Real[2]\ y; \tag{5.6}$$
$$output\ Real\ y0; \tag{5.7}$$
$$end\ ToSpacePhasor; \tag{5.8}$$

$$F1 : (electricalPowerSensor.i\_, \_) = ToSpacePhasor(\{electricalPowerSensor.plug\_p.pin[1].i, electricalPowerSensor.plug\_p.pin[2].i, electricalPowerSensor.plug\_p.pin[3].i\}); \tag{5.9}$$

$$y(\{electricalPowerSensor.plug\_p.pin[1].i, electricalPowerSensor.plug\_p.pin[2].i, electricalPowerSensor.plug\_p.pin[3].i\}) \tag{5.10}$$

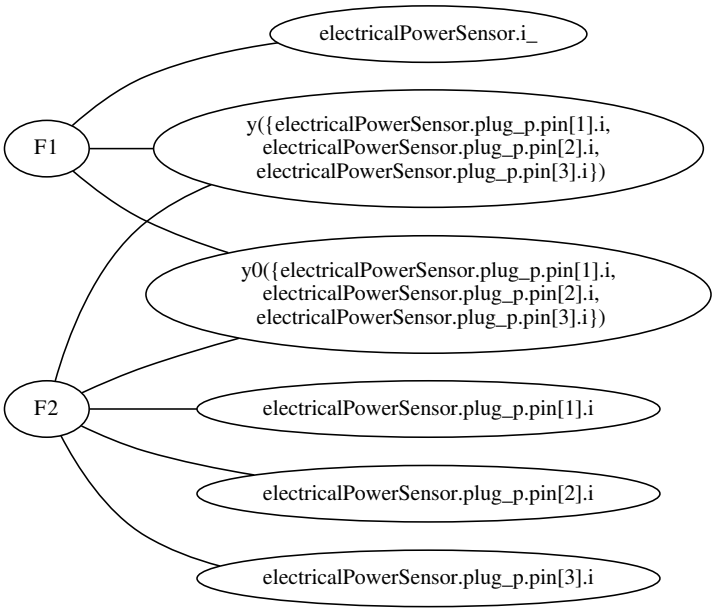


Figure 5.9: Resulting network segment for function call example from SMEE generator system.

$$y0(\{electricalPowerSensor.plug\_p.pin[1].i,\ electricalPowerSensor.plug\_p.pin[2].i,\ electricalPowerSensor.plug\_p.pin[3].i\}) \quad (5.11)$$

Modeling Logical Behavior Another new modeling technique is introduced for logical behavior, which is mathematically represented as discrete equations such as if-clauses, when-clauses, and assert statements. In such a scenario, the parameters in the condition of the if-clause are treated in the same manner as the parameters in the body of the if-clause. That is, parameters in both the condition and the body of the if-clause are modeled as parameter nodes and are connected to the same function node. For instance, in Eq. 5.12–5.14, all parameters are connected to the same function node as in Fig. 5.10.

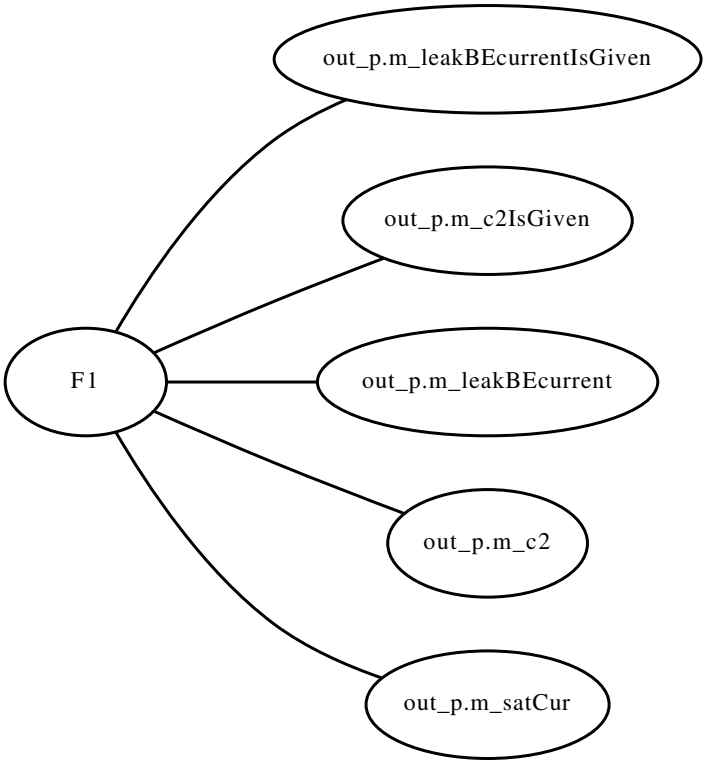


Figure 5.10: Resulting network segment for if-clause example from electrical oscillator system

$$if\ not\ out\_p.m\_leakBEcurrentIsGiven > 0.5\ then \tag{5.12}$$

$$out\_p.m\_leakBEcurrent := \tag{5.13}$$
$$out\_p.m\_c2 \times out\_p.m\_satCur;$$

$$end\ if; \tag{5.14}$$

5.2.2.3 Fault Injection

After the function and parameter nodes are identified for all forty systems, this information is exported to Mathematica for the next portion of the methodology. In particular, for each network, faults are injected into each node and the resulting change in ASPL (compared to the nominal i.e. non-faulted state of the network) is measured. This value of interest is the ΔASPL, computed as in Eq. 5.15, where ASPL is defined according to Eq. 5.1. ΔASPL is a relative measure of vulnerability for parameters in a behavioral network in which higher values indicate higher vulnerability.

$$\Delta ASPL = ASPL_{nominal} - ASPL_{fault} \tag{5.15}$$

Fault injection is performed in network models using the concept of network attack. Failure in a network is simulated generally by either removing a node or by reducing the edge weights associated with that node in order to signify the change in the ability of information to travel through the node. Consider node n in a fault state. To represent the presence of a fault in node n, all the weights of all edges associated with node n are reduced. This reduction in edge weight represents a degradation in performance in that node. Edge weights in the nominal case are equal to one. In the fault case, the edge weights are equal to the value of the fault variable chosen for the fault injection procedure. In this study, a fault variable of 0.5 is chosen. Note that faults are injected only into parameter nodes, not function nodes, due to the ambiguous physical meaning of injecting faults into functions rather than parameters. The physical meaning of attacking a parameter node is an off-nominal value due to operating conditions, manufacturing tolerances, or component failure.

5.2.2.4 Identification of Bridging Nodes

Bridging nodes are identified by using a community finding algorithm and the definition of bridging nodes [169, 170]. Any node with an edge connected it to a

community different than the one in which it is a member is considered a bridging node. The specific community finding algorithm used in this study is called modularity maximization, which identifies communities (modules) by iteratively testing various possible community structures and selects the one that maximizes the network's modularity, measured as Q-modularity in Eq. 5.16. In Eq. 5.16, where m is the number of edges, A_{ij} is an element of the adjacency matrix, k_i is the degree of vertex i, δ is the Kronecker delta, and c_i is the community in which node i belongs [158]. Positive Q-modularity indicates a higher degree of interconnectedness within the communities in the network than is likely to occur by chance. Negative Q-modularity indicates the opposite.

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j) \tag{5.16}$$

5.2.3 Results and Analysis

The average ΔASPL values for bridging and non-bridging nodes in a sample of the systems studied are provided in Table 5.3. For comparison and validation of the results, a second community finding algorithm, a spectral method, was also used to identify bridging nodes. This method partitions the graph using eigenvectors and eigenvalues of the adjacency matrix representation of the network. The average ΔASPL is compared between bridging and non-bridging nodes for the hypothesis test, which is an independent samples t-test with unequal variance. The results of the t-test are given in Table 5.4. The results indicate that bridging nodes have a significantly higher ΔASPL than non-bridging nodes for both community finding methods. Additionally, the effect sizes are strong and moderate for each respective community finding method. The standard deviation measures are provided in Table 5.5 for reference.

Additionally, the behavioral networks generated are visualized in Mathematica. A few examples are given in Fig. 5.11–5.14 in which the communities (determined by modularity maximization) are circled and the edges associated with the most

Table 5.3: ΔASPL and bridging nodes: average ΔASPL of bridging parameter nodes and non-bridging parameter nodes in a representative selection of systems.

System	Category	Non-Bridging, Modularity Maximization	Bridging, Modularity Maximization	Non-Bridging, Spectral	Bridging, Spectral
Multiphase Rectifier	Electrical	0.0519	0.0894	0.0508	0.0774
Saturated Inductor	Magnetic	0.0998	0.1664	0.0984	0.1399
Pump and Valve	Thermal /Fluid	0.0751	0.1587	0.0718	0.1290
Rolling Wheel	Mechanics	0.1309	0.1935	0.1338	0.1638

Table 5.4: ΔASPL and bridging nodes: t-test results.

Method	P-Value (one-tail)	Effect size	Non-bridging node mean ΔASPL (all 40 systems)	Bridging node mean ΔASPL (all 40 systems)
Modularity Maximization	< 0.001	1.2961	0.0828	0.1409
Spectral	0.005	0.7920	0.0818	0.1160

Table 5.5: ΔASPL and bridging nodes: standard deviation.

Method	Non-bridging node standard deviation of ΔASPL (all 40 systems)	Bridging node standard deviation of ΔASPL (all 40 systems)
Modularity Maximization	0.0448	0.0712
Spectral	0.0431	0.0712

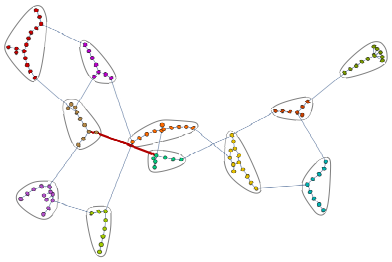


Figure 5.11: Behavioral network for simple drive train with grounded elements with edges associated with most vulnerable parameter node darkened and communities circled.

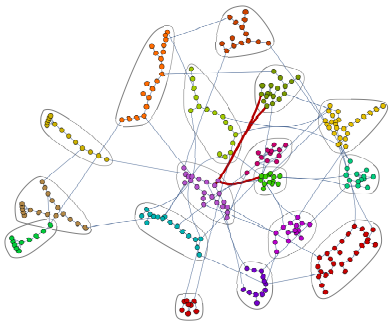


Figure 5.12: Behavioral network for electrical rectifier circuit with edges associated with most vulnerable parameter node darkened and communities circled.

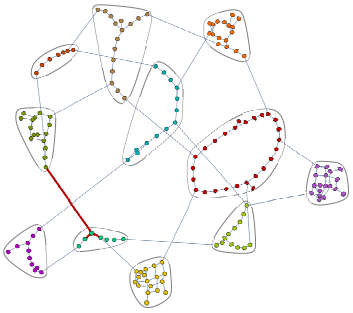


Figure 5.13: Elasto gap behavioral network with edges associated with most vulnerable parameter node darkened and communities circled.

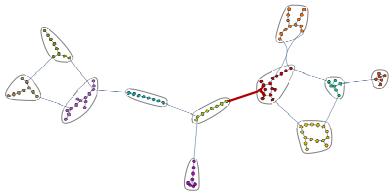


Figure 5.14: Control temperature of a resistor behavioral network with edges associated with most vulnerable parameter node darkened and communities circled.

vulnerable node in the network are highlighted. In each of these four cases, the highlighted node is a bridging node.

To further validate the results, the value of the fault variable is varied in order to test its effect on the results. To reiterate, a value of 0.5 is used for all nodes in all networks in the main portion of the study. In this test, the study is repeated with four different fault variable values: 0.2, 0.5, 0.7, and 0.9. The results are presented

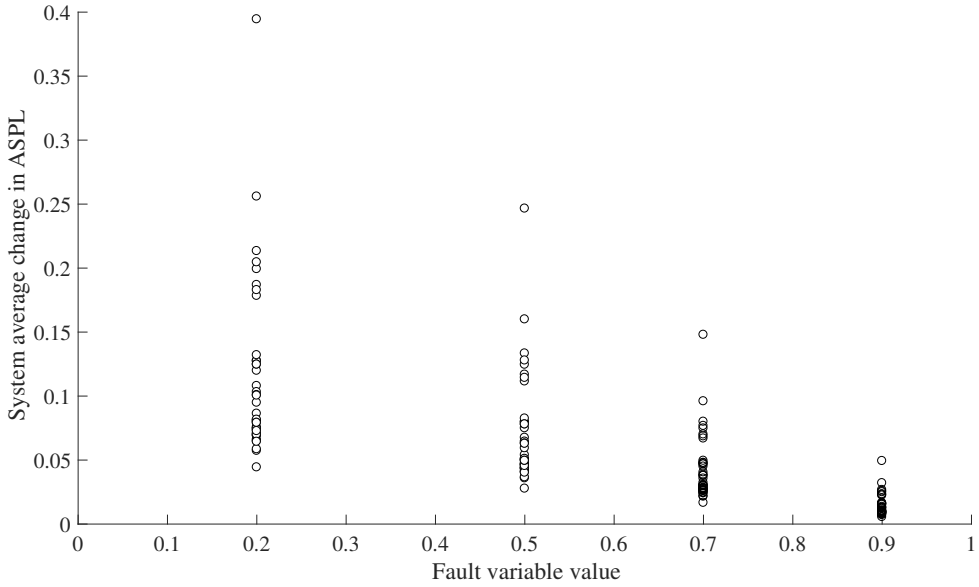


Figure 5.15: Effect of fault variable value on average ΔASPL in each system.

in Fig. 5.15, which indicates that the average ΔASPL decreases approximately linearly with an increasing fault variable. The ranking of systems from smallest to largest average ΔASPL, however, does not change with varying values of the fault variable. That is, while the fault variable does change the value of ΔASPL, it does not change the relative vulerability of the parameter nodes.

5.3 Measuring Component Importance with Structural Consequence

In the previous section, bridging nodes are identified as key topological features for controlling the robustness of engineered systems in general. In this section, a methodology for identifying vulnerable components in specific systems will be proposed. In contrast to the previous section, an architectural network is used, where functionality is assumed via directed network edges. The primary motivation for the proposed methodology is to identify components that are likely to require risk mitigation strategies early in the design process such that these mitigation strate-

gies can be accounted for early. While risk prevention is essential to increasing the safety of engineered systems, it often increases weight, cost, complexity, and performance. One example is the space shuttle, which had redundant computers with a voting system for on-board decision-making [171]. A second example is cold standby redundancies in which a system for failure detection and mitigation is required [172]. In general, the complexity added from such mitigation strategies can increase development cost and time [173]. As such, there is a trade-off between the necessary problem of failure prevention and mitigation and the negative effects of redundancy and other mitigation strategies, motivating the need for an early design method for including risk into the design with as little cost as possible.

It is useful to consider a component's individual contribution to overall system risk in order to identify components most in need of mitigation strategies. Two classical approaches are Statistical Importance and Structural Importance. These metrics are based on a fault tree, however, which do not fully represent consequences of failure in addition to causes of failure, which is necessary for a complete representation of safety architecture [174]. In this section, an alternative measure of component importance, structural consequence, is proposed. Rather than relying on a fault tree, which is often unavailable early in the design process, Structural Consequence Analysis (SCA) relies on a causal directed network quantifying the consequence of failure. This approach has the additional benefit of a more complete understanding of failure, with its emphasis on consequence of failure rather than probability of failure alone. This approach enables designers to consider risk and design mitigation strategies into a system earlier.

5.3.1 Directed Network Modeling

SCA involves the representation of system architecture using a directed acyclic network which enables an explicit representation of redundancy as well as architectural and causal information. Network representation of system architecture is a natural extension of the commonly used design structure matrix (DSM), which represents connections between system elements with a matrix. A network is sim-

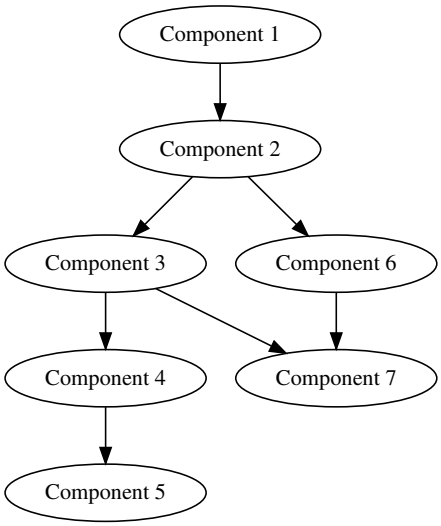


Figure 5.16: Simple example of an architectural representation of system architecture.

ply a graphical representation of a DSM. Networks by definition are simply a collection of nodes and edges. In directed networks, the edges also have directionality. In SCA, edges represent a physical connection between components and/or the transmission of information or energy between components. A simple example is presented in Fig. 5.16. There are three types of nodes in SCA: source nodes, middle nodes, and sink nodes. Source nodes, such as Component 1 in Fig. 5.16, have outgoing edges (edges directed away from the node) but no incoming edges (edges directed toward the node). Middle nodes, such as Component 2, have both incoming and outgoing edges. Sink nodes, such as Component 5, have incoming edges only. All nodes in the network are assumed to be functionally necessary for the operation of at least one of the sink nodes.

5.3.1.1 Representing Redundancy

Redundant components are represented as separate nodes, named in a specific manner, and generally share the same incoming and outgoing edges as the compo-

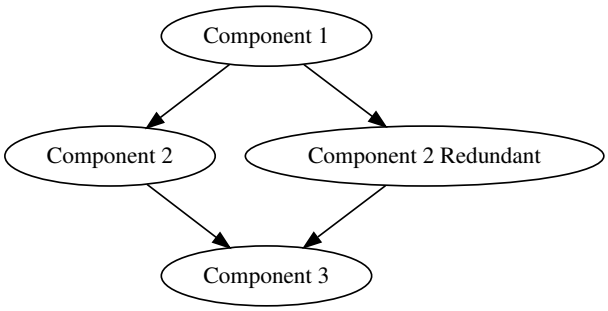


Figure 5.17: Representation of redundancy, where the redundant component is called Component 2 Redundant.

nents to which they are redundant. An example is given in Fig. 5.17. As per the naming convention, Component 2 Redundant is redundant to Component 2. In general, the base component has no specialized name, but the components redundant to the base component have an additional signified: the word "Redundant" followed by a unique number.

5.3.1.2 Eliminating Algebraic Loops

To ensure the graph is acyclic, a procedure is developed for eliminating algebraic loops. Eliminating algebraic loops ensures that the consequence propagation algorithm is applied deterministically. Loops may form due to system feedback or in certain shared redundancy cases. Feedback loops are eliminated by removing the connection between the component providing information and the component receiving information. For example, for a motor voltage sensor, the connection between the sensor and motor is removed. While this feedback behavior is not modeled in the network, this simplification is justified by reasoning that there is no interest in propagating a motor malfunction to the sensor because a sensor failure leads to a motor malfunction (an event that has already occurred). Shared redundancies may also create loops, as in Fig. 5.18. BuR is redundant to Bu1 only. However, since Bu1 and Bu2 are the same type of component, it is possible to use BuR as a redundancy for both, as in Fig. 5.19. This representation, however,

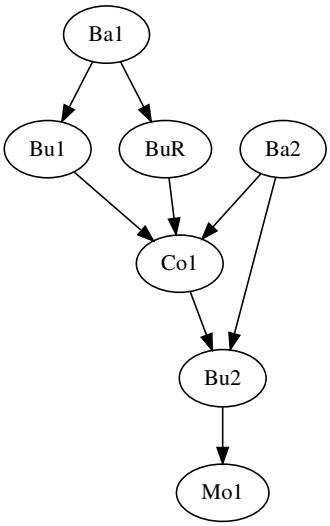


Figure 5.18: Network with a redundant component, BuR, which is redundant to Bu1.

creates a loop that does not physically exist in the system, since BuR will never be redundant to Bu1 and Bu2 simultaneously. To break this loop, BuR is separated into two nodes: BuRA and BuRB, as in Fig. 5.20. This solution correctly implies that BuR can only be redundant to one component at a time, although it can be redundant to either.

5.3.2 Structural Consequence Calculation

The structural consequence metric quantifies the consequence of component failures in the directed acyclic network model of system architecture. The metric is normalized on a scale of $[0, 1]$ such that 0 corresponds to the absence of consequence of failure and 1 corresponds to the most severe consequence possible. To compute the metric, sink nodes are initialized with user-defined consequences, and these consequences are propagated through the directed network to compute consequences for each node. Consequences are computed according to set operations within the consequences space.

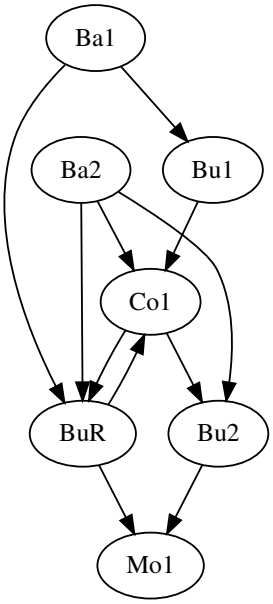


Figure 5.19: BuR is redundant to Bu1 and Bu2. There is a loop between Co1 and BuR.

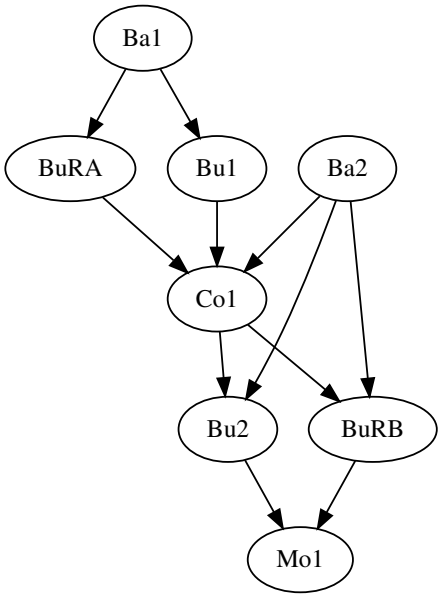


Figure 5.20: Algebraic loop eliminated by splitting BuR intro BuRA and BuRB.

5.3.2.1 Construction of the Consequence Space

The consequences space, Ω, is constructed using the aggregation of all sources of cost. The cost of the entire consequence space is normalized to 1. Failure scenarios representing different proportions of the consequence space are represented within the consequence space as portions of the total set. Some failure scenarios may share consequences, which is represented as an intersection. Set operations are used to calculate consequences of various failure scenarios. In the following subsections, C_{k_1} represents the consequence of node k_1, $E_{k_1:k_2}$ represents the consequence of the edge connecting C_{k_1} to C_{k_2}, and $S(C_{k_1})$, $S(E_{k_1:k_2})$ represents the consequent of component k_1 and edge $E_{k_1:k_2}$.

5.3.2.2 Scoring the Sink Nodes of the Network

The first step in the consequence calculation is to initialize the propagation algorithm by estimating the cost of sink nodes. Estimations may be based on available data such as expert opinion or field data. For instance, if a sink node is an aileron, a cost estimate is made based on previous instances of aileron failures. Fig. 5.21 provides an example of the assignment of consequences in Ω for sink nodes.

5.3.2.3 Scoring the Edges of the Network

Once the sink nodes have been initialized, their consequences are backpropagated through the directed network by computing the consequences of failure of both edges and nodes. Edge consequences represent consequences of the severing of connection or flow of information between two nodes.

Scoring Edges with No Redundancies Assuming no redundancy, failure of an edge leads to a failure of the component to which it supplies information or is connected. Failure of an edge for which no redundancies or protective factors exist leads to the potential failure of the component it feeds (at the head of the edge).

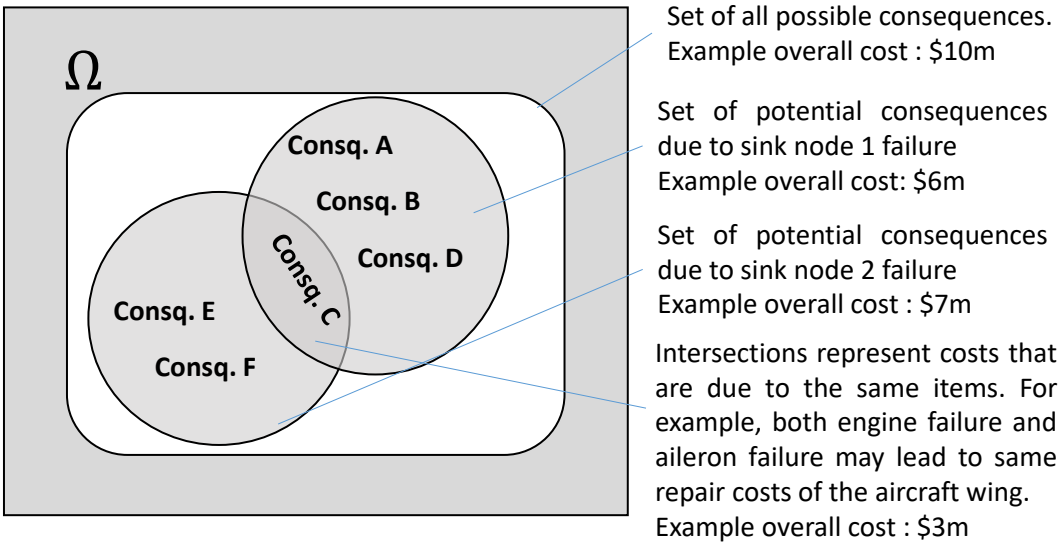


Figure 5.21: Consequence space visualization.

This means that the failure or loss of the edge bears the same consequence as the loss of the component it feeds. Therefore, for no redundancy, Eq. 5.17 holds. This concept is visualized in Fig. 5.22, in which the loss of either of the edges leads to the failure of node A, as well as Property 1.

$$S(E_{i:j}) = S(C_j) \tag{5.17}$$

Property 1: An edge for which no redundancies or protective features exist bears the same consequence as the component it feeds.

Scoring Edges with Redundancies Calculation of edge consequence differs when there are redundancies. Redundant edges share the same function and their leading edges are connected to the same component. If all redundant edges and the original edge fail, the component they supply also fails. Mathematically, the union of n redundant edges, $E_{i_1:j} \cdots E_{i_n:j}$, is equally consequential to the component

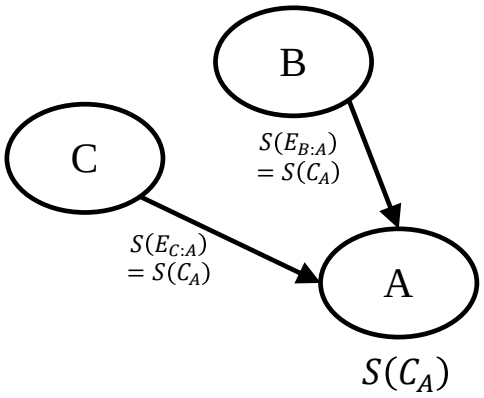


Figure 5.22: Example of scoring edges with no redundancies.

supplied, as in Eq. 5.18, also expressed in Property 2. Further, all redundant edges are assumed to be equally consequential, as expressed in Assumption 1 and Eq. 5.19. Eq. 5.20 can be obtained using Assumption 1 and Property 1.

$$S(E_{i_1:j} \cup E_{i_2:j} \cdots E_{i_n:j}) = S(C_j) \tag{5.18}$$

Property 2: The union of a collection of redundant edges bears the same consequence as the component they feed.

Assumption 1: All redundant edges are assumed to have equivalent consequences, where edge consequences represent mutually exclusive subsets of the component consequence with equivalent consequences.

$$S(E_{i_1:j}) = S(E_{i_2:j}) = \cdots = S(E_{i_n:j}) \tag{5.19a}$$

$$E_{i_k:j} \cap E_{i_m:j} = \emptyset \quad k, m \in \{1, \ldots, n\} | k \neq m \tag{5.19b}$$

$$S(E_{i_1:j}) = S(E_{i_2:j}) = \cdots = S(E_{i_n:j}) = \frac{S(C_j)}{n} \tag{5.20}$$

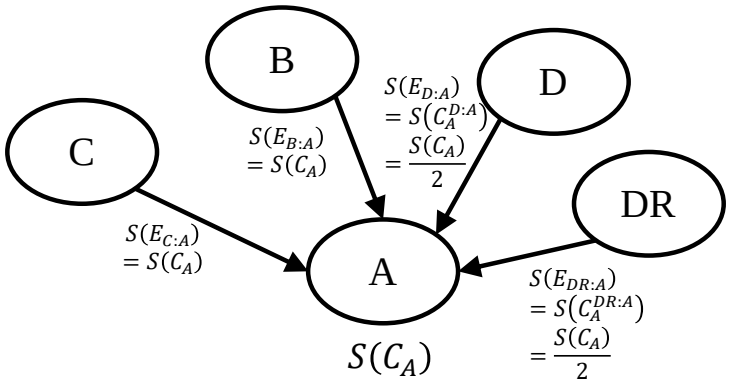


Figure 5.23: Example of scoring edges with redundancies.

By Assumption 1, each redundancy equally shares the total consequences. If the overall consequence C_j is subdivided into a set of shared and mutually exclusive consequences $C_j^{i_1:j}, C_j^{i_2:j}, \cdots C_j^{i_n:j}$ shared by redundant edges $E_{i_1:j} \cdots E_{i_n:j}$ (such that $C_j^{i_k:j} = E_{i_k:j}$), Eq. 5.21 holds. For example, in Fig. 5.23, D and DR are redundant, and the consequence of A is shared equally between the two edges.

$$C_j^{i_l:j} \cap C_j^{i_m:j} = \emptyset \quad l, m \in \{1, \cdots, n\}\ l \neq m \tag{5.21a}$$
$$C_j^{i_1:j} \cup C_j^{i_2:j} \cdots \cup C_j^{i_n:j} = C_j \tag{5.21b}$$
$$C_j^{i_l:j} \cup C_j = C_j \tag{5.21c}$$
$$C_j^{i_l:j} \cap C_j = C_j^{i_l:j} \tag{5.21d}$$

5.3.2.4 Scoring the Nodes of a Network

Node consequences are computed based on the consequences of their outgoing edges. This is because the failure of a component leads to the failure of all information flows originating from that component (Property 3), as in Eq. 5.22,

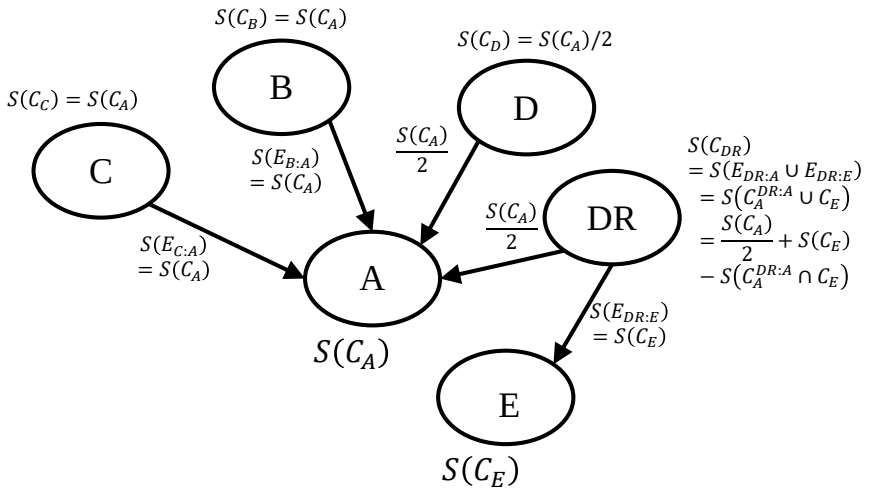


Figure 5.24: Example of scoring nodes.

illustrated in Fig. 5.24. Consequences remain in set theoretic terms until all nodes and edges are evaluated, at which point the set can be quantified as structural consequence.

Property 3: A component bears the same consequence as all of the outgoing edges from that component.

$$S(C_i) = S(E_{i:j_1} \cup E_{i:j_2} \cdots \cup E_{i:j_n}) \tag{5.22}$$

5.3.2.5 Propagation Algorithm

Node and edge consequences are computed in a given order by propagating sink node consequences backward through the network. The propagation algorithm assumes the network has no undirected edges and the network is acyclic (no loops). Propagation begins with the initialized sink nodes. Edges associated with sink nodes are first evaluated. The algorithm then proceeds randomly with a random node with all its outgoing edges are evaluated. This random selection continues until all node consequences are computed. This algorithm is summarized in Fig.

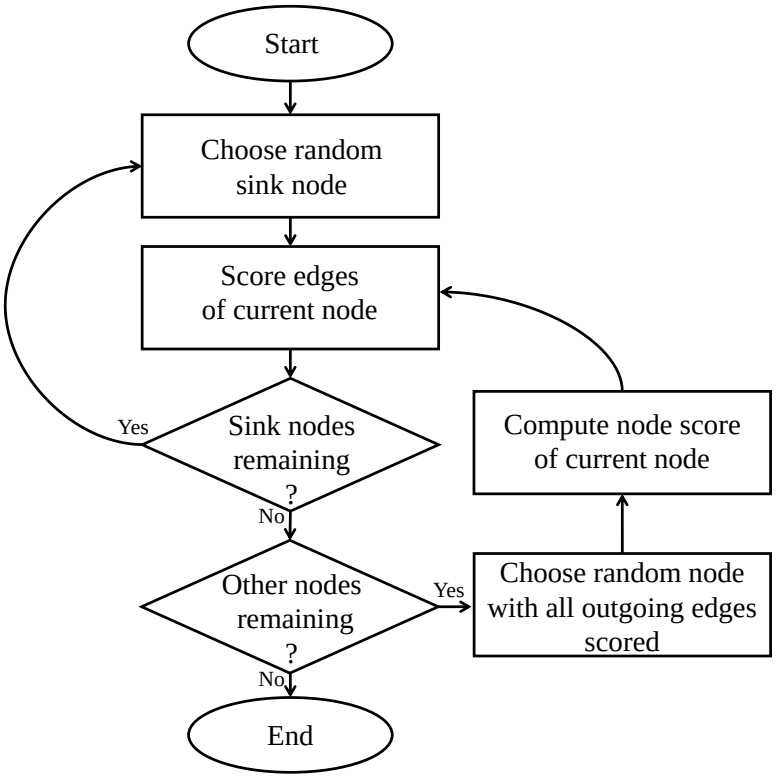


Figure 5.25: Flowchart of propagation algorithm.

5.25. Once propagation is complete, structural consequence of each node and edge can be quantified based on the consequence space Ω.

5.3.3 Case Study: Electric Line for an Unpiloted Aircraft System (UAS) for Urban Air Mobility (UAM)

The proposed approach is demonstrated using a case study of a simplified electric subsystem of an all-electric fixed-wing unpiloted aircraft system (UAS). This system powers electro-mechanical actuators, which manupulate an aileron and flap of the UAS. The system without redundancy is represented as a block diagram in Fig. 5.26. Two different redundancy schemes are evaluated. The first, given in Fig. 5.27, adds a single redundancy for each component. The second, Fig. 5.28, uti-

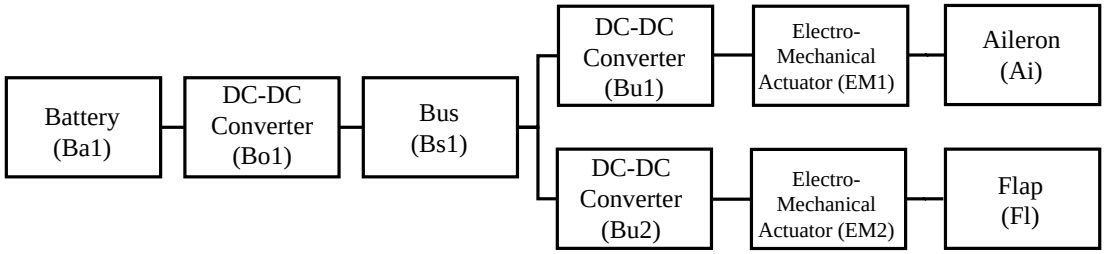


Figure 5.26: Electrical subsystem block diagram for base architecture with no redundancies added.

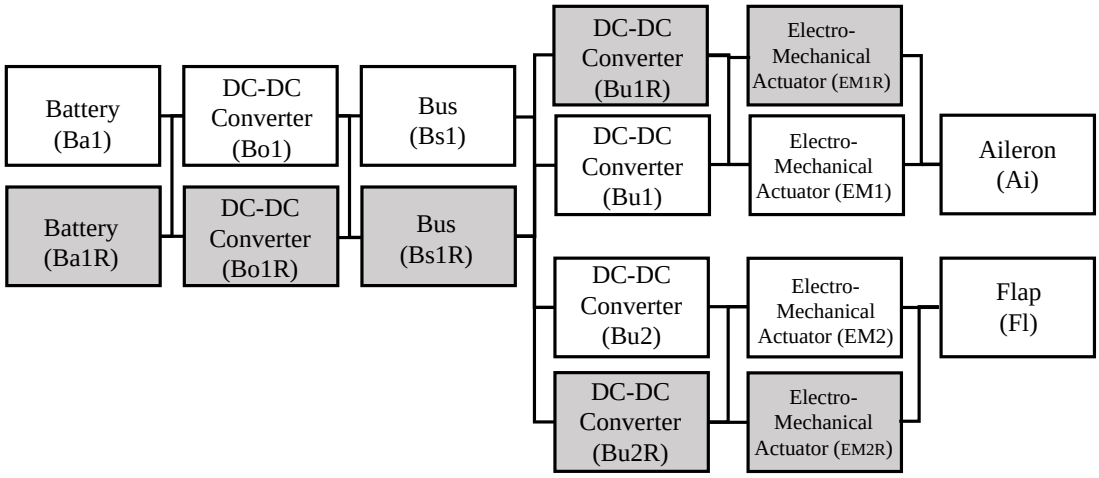


Figure 5.27: Electrical subsystem block diagram for architecture with parallel redundancies.

lizes a re-configurable redundancies, which can be shared by multiple components. Structural consequence is used to evaluate the importance of each component, thereby enabling the assessment of the redundancy configurations.

The consequence space consists of a total cost of $\$800,000$ assigned to aileron failure, $\$300,000$ assigned to flap failure, and $\$100,000$ shared between the two components for $\$1,000,000$ in total cost. All costs are normalized on a $[0, 1]$ scale, meaning $\$1,000,000$ in cost corresponds to a structural consequence $S(\Omega) = 1$, and the sink node consequences are given in Eq. 5.23. A directed network representa-

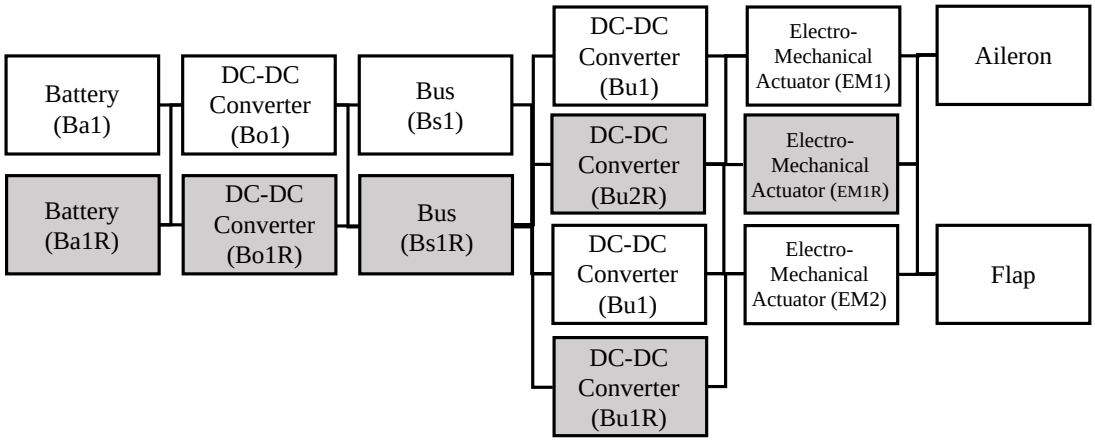


Figure 5.28: Electrical subsystem block diagram for architecture with reconfigurable redundancies.

tion is constructed for each architecture (base, parallel, and reconfigurable), given in Fig. 5.29–5.31. The white and gray portions of BuR1 and EM1R are reconfigurable components with the same connections (essentially, two separate nodes represented in a compact form). Using the approach presented, edge and node consequences for all architectures are presented in Fig. 5.29–5.31 and tabulated in Table 5.6.

$$S(C_{Ai}) = 0.8$$
$$S(C_{Fl}) = 0.3$$
$$S(C_{Ai} \cap C_{Fl}) = 0.1$$

SCA reveals nodes with high consequence of failure based on system architecture. This information enables the early identification of system vulnerabilities. Additionally, the proposed modeling formalism enables the construction of causal chains leading to multiple failures with potentially different consequences, not only a single top-event as in a fault tree. Furthermore, structural consequence allows

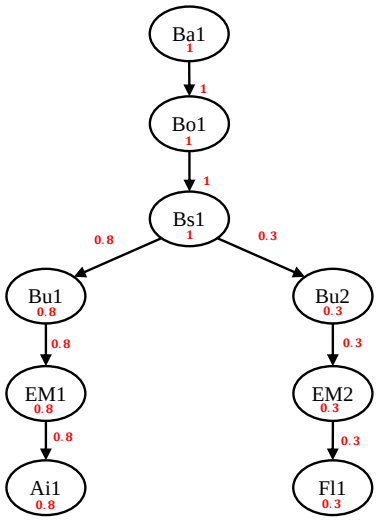


Figure 5.29: Network representation of the base architecture.

Table 5.6: Component consequences for base, parallel, and re-configurable architectures.

Component	Base	Parallel	Re-config.
Ba1	1	0.5	0.5
Ba1R1	N/A	0.5	0.5
Bo1	1	0.5	0.5
Bo1R1	N/A	0.5	0.5
Bs1	1	0.5	0.479
Bs1R1	N/A	0.5	0.521
Bu1	0.8	0.4	0.33
Bu1R1	N/A	0.4	0.4
Bu1R2	N/A	N/A	0.13
Bu2	0.3	0.15	0.15
Bu2R1	N/A	0.15	N/A
EM1	0.8	0.4	0.4
EM1R1	N/A	0.4	0.4
EM2	0.3	0.15	0.15
EM2R1	N/A	0.15	N/A
Ai1	0.8	0.8	0.8
Fl1	0.3	0.3	0.3

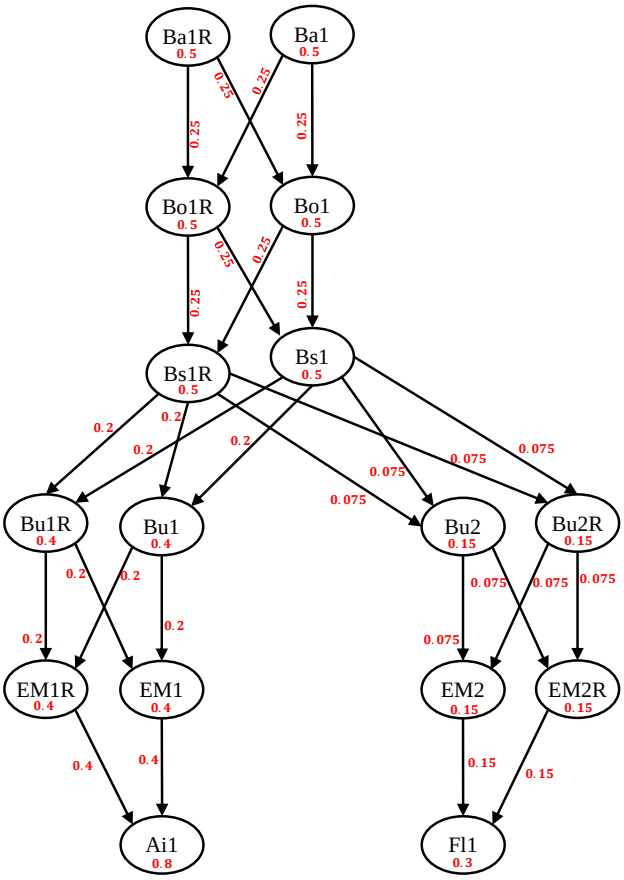


Figure 5.30: Network representation of the architecture with parallel redundancies.

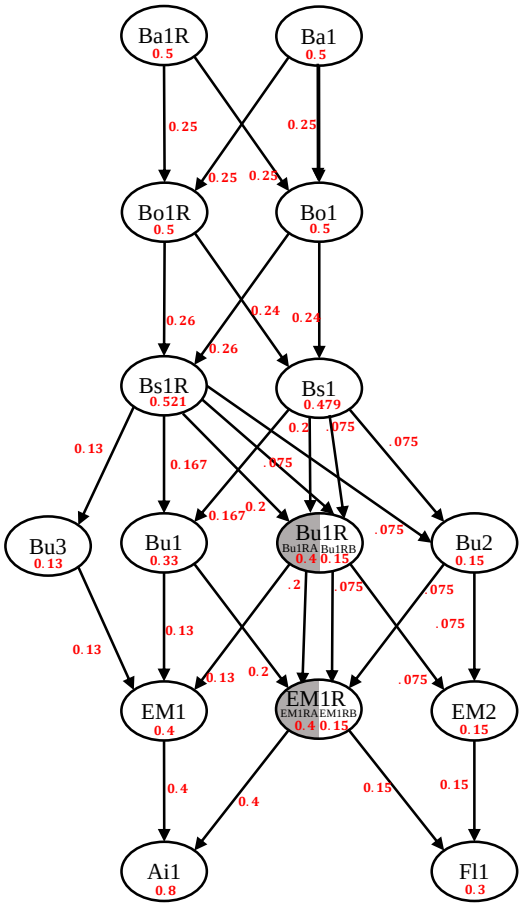


Figure 5.31: Network representation of the architecture with parallel and reconfigurable redundancies (white & gray).

the evaluation of various redundancy schemes. Nodes with high consequence are good candidates for additional redundancy placement, and redundancy schemes with very high consequences overall are less desirable than those with lower consequences. The addition of redundancy reduces the share of consequence among components.

Conceptually, SCA provides a different perspective on vulnerable nodes compared to the analysis of bridging nodes. High-consequence nodes are not necessarily critical from a network perspective, but rather from an estimation of costs associated with their failure. The distinction between SCA and identification of bridging nodes demonstrates a key methodological difference between many graph-based approaches and the network theoretic approach spanning much of this research. In SCA, the graph is primarily a modeling and visualization tool and the computation is performed using set theory, whereas in the bridging node approach, vulnerable nodes are identified through an analysis of network topology. In practice, bridging nodes will be more useful in one of two scenarios: either in early design, when detailed simulations or cost analysis are untenable, or in very large-scale, complex systems where they are infeasible and in which emergent behaviors are expected. Bridging nodes capture emergent behavior and require significantly less data and modeling work to obtain, whereas high consequence nodes may be more suitable for studying certain subsystems independently of the integrated system.

5.4 Chapter Summary

In this chapter, conceptual and empirical arguments for the correlation between network metrics and bridging nodes are presented. Given the community structure (modules) of a system, this finding makes possible the *a priori* prediction of vulnerable system variables without the use of expensive full-scale models. In other words, the structure of the equations representing system behavior is sufficient for gaining a preliminary understanding of system robustness without actually requiring the simulation of these equations. Moreover, key parameters contributing to fragility can be identified by locating bridging nodes in the network-based repre-

sentation of these equations. This contribution directly addresses the first research question by identifying bridging nodes as variables that are key to controlling a system's robustness to failure. This finding furthers the objective of the work by identifying a pattern of vulnerability in engineered systems that can be used to predict problems even in early design, when full-scale models are unavailable, and in complex and/or large systems, when even the most sophisticated models may fail to accurately predict behavior due to emergent behavior and unintended consequences. As an alternative perspective on this research question, in the second part of this chapter, a method is presented for identifying important components in a system architecture in terms of their consequence of failure. This method enables the early prediction of components that are likely to require redundancy or other mitigation strategies. Rather than investigating a general pattern of vulnerability, as in the first section of this chapter, the method presented in this section is used to find vulnerabilities in specific cases. This method is applicable early in the design process, before detailed system models are available, enabling such mitigation strategies to be considered earlier in the design process, saving cost and redesign time. Both methods provide preliminary, low cost assessment of robustness. After identifying these vulnerabilities, designers can focus their efforts on more tightly controlling these vulnerabilities.

Chapter 6: Conclusions

This research has addressed the structural characteristics of robustness in complex engineered systems. Robustness in this context is considered the ability of a system to retain acceptable performance even when faced with faults, perturbations, and/or unintended consequences. This work presents empirical findings in addition to novel conceptualizations and approaches for assessing robustness of complex systems. Namely, this research utilizes machine learning and network theory to identify patterns of vulnerability such that *a priori* analysis even of complex systems that are difficult to model without abstraction and modularization is possible. This research considers robustness at three levels of hierarchy.

This research begins by assessing the structural characteristics of robustness at the macro-scale: first, the effect of an action implemented within one system on another system is considered by identifying archetypes of unintended consequences. To achieve this, the concept of an unintended consequence is first clarified by developing a conceptual framework for unintended consequences in engineering design. Once this has been established, a combination of human coding of semantic descriptions of unintended consequences and machine learning is used to discern archetypes of unintended consequences from a data base of real-world adverse events. In total, sixty-six archetypes of unintended consequences are identified. These archetypes are related to a set of weighted risk factors, which are effectively design-stage leading indicators of unintended consequences. Because of this relationship between risk factors and archetypes, it is possible to use these risk factors as an *a priori* assessment of risk during design. Using machine learning to identify leading indicators of adverse events from data circumvents issues of modeling complex systems, namely abstraction and modularity, by instead treating a system as a black box from which correlations between leading indicators and adverse events can be determined.

Second, at the meso-scale, the effect of various system architectures on robustness is considered. Specifically, this research investigates whether modularity helps or hinders topological robustness. A network perspective is taken in understanding robustness. In networks, robustness is measured as the degree to which the structure of a network changes in response to node attack or removal. There are various network theoretic metrics that can be used to measure a network's resistance to attack, specifically ASPL and robustness coefficient. This research measures network topological robustness and modularity, measured as Q-modularity, in various different network models of real engineered systems. Four types of network models are used. Each network model represents various aspects of the system's architectural and behavioral morphology. The results indicate that there is a negative correlation between modularity and robustness, implying that modularity may come at the expense of topological robustness. This finding provides a caveat to the use of modularity as a design rule when designing complex systems.

Third, at the micro-scale, the relationship between a variable's role within a module and its effect on system robustness is considered. Network theory is again used to model and analyze system robustness. This approach aims to identify specific nodes, which in this case represent variables in the behavioral model of the system, which are key to controlling a system's behavior. The behavioral network model is bipartite, containing function and parameter nodes, and is derived from the governing equations of the system. Faults are injected into the network model by attacking nodes. This is done by reducing the edge weights surrounding a node that is in a fault state. Then, the network's change in global performance as a result of the injected fault is measurable through network metrics such as ASPL. Larger changes in such network metrics due to fault injection are associated with more vulnerable nodes. In this way, it is found that bridging nodes tend to be more vulnerable than non-bridging nodes, where bridging nodes are nodes that connect modules, or communities. This implies that the interfaces between modules or subsystems are more important even than highly influential nodes within a module, from a topological standpoint. With much of complex system design being highly

modularized, this finding directs attention to the importance of integration of subsystems and the design of interfaces.

Taken together, the findings provide a multi-faceted approach to understanding the factors that promote or inhibit robustness in engineered systems, validated through empirical findings. These findings can be used as early design stage indicators of potential vulnerabilities that make a complex system more or less fragile to perturbations, thereby informing robustness analysis. Complex systems present challenges in modeling and analysis. All models are, by definition, abstractions to some degree and will therefore neglect some behaviors or characteristics; modeling complex systems, however, is even more fraught. Complex systems, due to their size, tight coupling, and non-linearity, are difficult to understand for an outside observer. Human observers have limits to their time and mental resources for analysis in decision making – in other words, they are bounded rational. Theoretically, then, it is untenable to model all possible interactions and behaviors at a detailed level in complex systems, and as such, unexpected behaviors may be discovered after the system is integrated and tested. These advances in this research will enable robustness analysis of complex engineered systems that captures emergent behavior and unintended consequences – in other words, under-recognized vulnerabilities in complex systems – before integration and testing occur. As such, the methods in this research are most useful in high complexity, early design situations, differentiating this research from existing methods, as shown in Fig. 6.1. They may additionally be moderately useful in low complexity, early design or high complexity, late design situations.

The perspectives on robustness taken in this research extend conventional theories on robustness in that they are more capable of handling complexity. Specifically, machine learning is able to identify leading indicators that a complex system will respond poorly or, in other words, is fragile to unexpected perturbations. Network theory, on the other hand, uses abstraction to avoid modularization common in more detailed models, enabling the analysis of the interfaces between subsystems in early design and detecting emergent properties such as high degree nodes and

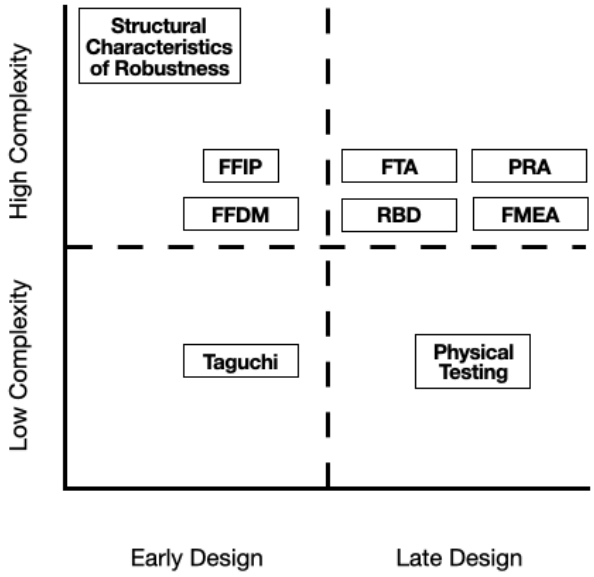


Figure 6.1: The methods discussed in this research are most useful in high complexity, early design situations.

bridging nodes. These advances represent a shift in robustness analysis of complex systems, such that designers can think not of how to improve optimization models or uncertainty representations in robustness analysis, but rather to consider patterns of vulnerabilities in network models and historical data as early indicators of fragility.

Bibliography

[1] NASA. NASA Public Lessons Learned Information System. `https://llis.nasa.gov/`, 2019. Accessed: 2019-11-01.

[2] Genichi Taguchi and Asian Productivity Organization. *Introduction to Quality Engineering: designing quality into products and processes*. The Organization Tokyo, 1986.

[3] Sarah A. Sheard and Ali Mostashari. A complexity typology for systems engineering. *INCOSE International Symposium*, 20(1), 2010.

[4] Mohammad Hassannezhad, Marco Cantamessa, Francesca Montagna, and P. John Clarkson. Managing sociotechnical complexity in engineering design projects. *Journal of Mechanical Design*, 141(8):081101.

[5] Jairo da Costa Junior, Jan Carel Diehl, and Dirk Snelders. A framework for a systems design approach to complex societal problems. *Design Science*, 5(e2), 2019.

[6] S.D. Gribble. Robustness in complex systems. pages 21– 26, 06 2001.

[7] Abhijit Deshmukh and Paul Collopy. Fundamental research into the design of large-scale complex systems. In *13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference, 13 September 2010 – 15 September 2010, Fort Worth, Texas*. 2010.

[8] Government Accountability Office. NASA assessments of major projects. Technical Report GAO-19-262SP, May 2019.

[9] Camila Ludovique Callegari, Alexandre Szklo, and Roberto Schaeffer. Cost overruns and delays in energy megaprojects: How big is big enough? *Energy Policy*, 114:211–220, March 2018.

[10] Yaneer Bar-Yam. When systems engineering fails – towards complex systems engineering. In *SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme - System Security and Assurance*, 2003.

[11] Carliss Baldwin and Kim Clark. *Design Rules: The Power of Modularity*. The MIT Press, 2000.

[12] Robert E. Oberto, Erik L. Nilsen, Ron Cohen, R. Wheeler, P. DeFlono, and Chester Borden. The NASA exploration design team: blueprint for a new design paradigm. In *2005 IEEE Aerospace Conference*, 2005.

[13] Yupeng Li, Zhaotong Wang, Lei Zhang, Xuening Chu, and Deyi Xue. Function module partition for complex products and systems based on weighted and directed complex networks. *Journal of Mechanical Design*, 139(2):021101, 2017.

[14] Sendil Ethiraj and Daniel Levinthal. Modularity and innovation in complex systems. *Management Science*, 50, 01 2004.

[15] Sulaiman F. Alyaqout, Diane L. Peters, Panos Y. Papalambros, and A. Galip Ulsoy. Generalized coupling management in complex engineering systems optimization. *Journal of Mechanical Design*, 133(9):091005, 2011.

[16] Nicolas F. Soria Zurita and Irem Y. Tumer. Towards understanding emergent behavior in complex engineered systems. In *Proceedings of the ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Volume 7: 29th International Conference on Design Theory and Methodology, Cleveland, OH, USA, August 6, 2017–August 9, 2017*, 2017.

[17] Yaneer Bar-Yam. *Dynamics of complex systems*. CRC Press, 2019.

[18] 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.

[19] Tim Kelly, Iain Bate, John McDermid, and Alan Burns. Building a preliminary safety case: An example from aerospace. In *Proceedings of the 1997 Australian Workshop on Industrial Experience with Safety Critical Systems and Software, Sydney, Australia*, pages 1–10, 1997.

[20] Department of Defense. Procedures for performing failure mode, effects, and criticality analysis. Technical Report MIL-STD-1629A, Washington, DC, 1980.

[21] David Jensen. *Enabling safety-informed design decision making through simulation, reasoning and analysis*. PhD thesis, Oregon State University, Corvallis, Oregon, 2012.

[22] W.E. Vesely, F.F. Goldberg, N.H. Roberts, and D.F. Haasl. The fault tree handbook. Technical Report NUREG0492, Washington, DC, 1981.

[23] Richard E. Barlow and Frank Proschan. Importance of system components and fault tree events. *Stochastic Processes and their Applications*, 3(2):153–173, 1975.

[24] Z. Birnbaum. On the importance of different components in a multicomponent system. Technical Report 54, 1968.

[25] Dong Yuhua and Yu Datao. Estimation of failure probability of oil and gas transmission pipelines by fuzzy fault tree analysis. *Journal of Loss Prevention in the Process Industries*, 18(2):83–88, 2005.

[26] R. M. Sinnamon and J.D. Andrews. Improved accuracy in quantitative fault tree analysis. *Quality and Reliability Engineering International*, 13(5):285–292, 1998.

[27] E. Ruijters and M. Stoelinga. Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools. *Computer science review*, 15–16:29–62, 2015.

[28] Hichem Boudali, Pepijn Crouzen, and Marille Stoelinga. A compositional semantics for dynamic fault trees in terms of interactive markov chains. *International Symposium on Automated Technology for Verification and Analysis ATVA 2007: Automated Technology for Verification and Analysis*, pages 441–456.

[29] J.B. Dugan, S.J. Bavuso, and M.A. Boyd. Fault trees and sequence dependencies. *Annual Proceedings on Reliability and Maintainability Symposium*, 1990.

[30] X. Liang, H. Yi, Y. Zhang, and D. Li. A numerical simulation approach for reliability analysis of fault-tolerant repairable system. *Proceedings of the 8th International Conference Reliability, Maintainability and Safety*, pages 191–196, 2009.

[31] X. Zhang, Q. Miao, X. Fan, and D. Wang. Dynamic fault tree analysis based on petri nets. *Proceedings of the 8th International Conference Reliability, Maintainability and Safety*, pages 138–142, 2009.

[32] Yves Dutuit and Antoine Rauzy. Efficient algorithms to assess component and gate importance in fault tree analysis. *Reliability Engineering System Safety*, 72(2):213–222, 2001.

[33] K. Buchacker. Modeling with extended fault trees. *Proceedings of the 5th International Symposium on High Assurance Systems Engineering*, pages 238–246, 2000.

[34] A. Bobbio and D. Codetta-Raiteri. Parametric fault trees with dynamic gates and repair boxes. *Proceedings of Reliability and Maintainability Symposium*, pages 459–465, 2004.

[35] Elmer Eugene Lewis. *Introduction to reliability engineering*, volume 2. Wiley New York et al., 1987.

[36] Reece Clothier, Ewen Denney, and Ganesh J Pai. Making a risk informed safety case for small unmanned aircraft system operations. In *17th AIAA Aviation Technology, Integration, and Operations Conference*, page 3275, 2017.

[37] Stanley Kaplan and B. John Garrick. On the quantitative definition of risk. *Risk Analysis*, 1(1), 1981.

[38] Nathan O. Siu and Dana L. Kelly. Bayesian parameter estimation in probabilistic risk assessment1the views and conclusions in this paper are those of the authors and should not be interpreted as necessarily representing the views or official policies, either expressed or implied, of the us nuclear regulatory commission or the us department of energy.1. *Reliability Engineering System Safety*, 62(1):89 – 116, 1998.

[39] Zahra Mohaghegh, Reza Kazemi, and Ali Mosleh. Incorporating organizational factors into probabilistic risk assessment (PRA) of complex socio-technical systems: A hybrid technique formalization. *Reliability Engineering System Safety*, 94(5):1000 – 1018, 2009.

[40] K. Vierow, K. Hogan, K. Metzroth, and T. Aldemir. Application of dynamic probabilistic risk assessment techniques for uncertainty quantification in generation iv reactors. *Progress in Nuclear Energy*, 77:320 – 328, 2014.

[41] Brandon Haley. Evaluating complex engineered systems using complex network representations. Master's thesis, Oregon State University, Corvallis, Oregon, 2014.

[42] Dimitri Kececioglu. Reliability engineering handbook. Technical report, 2002.

[43] S. Distefano and L. Xing. A new approach to modeling the system reliability: dynamic reliability block diagrams. In *IEEE Reliability and Maintainability Symposium (RAMS06)*, pages 189–195, 2006.

[44] Eric Maass and Patricia D. McNair. *Applying Design for Six Sigma to Software and Hardware Systems*. Prentice Hall, 2009.

[45] Kai-Lu Wang and Yan Jin. An analytical approach to function design. In *Proceedings from the 14th International Conference on Design Theory and Methodology*, 2002.

[46] Jonathan Smith and P. John Clarkson. Design concept modelling to improve reliability. *Journal of Engineering Design*, 16(5):473–492, 2005.

[47] Tolga Kurtoglu and Irem Y. Tumer. A graph-based fault identication and propagation framework for functional design of complex systems. *Journal of Mechanical Design*, 130(5):51401, 2008.

[48] 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*, 29(1), 2018.

[49] Robert B. Stone, Irem Y. Tumer, and Michael Van Wie. The function-failure design method. *Journal of Mechanical Design*, 3(127):397–407, 2004.

[50] K. Grantham Lough, Robert B. Stone, and Irem Y. Tumer. Failure prevention in design through effective catalogue utilization of historical failure events. *Journal of Failure Analysis and Prevention*, 8(5):469–481, 2008.

[51] Nita Yodo and Pingfeng Wang. Resilience allocation for early stage design of complex engineered systems. *Journal of Mechanical Design*, 138(9), 07 2016. 091402.

[52] Mohit Goswami and M. K. Tiwari. A predictive risk evaluation framework for modular product concept selection in new product design environment. *Journal of Engineering Design*, 25(1-3):150–171, 2014.

[53] Yao Cheng and Xiaoping Du. System Reliability Analysis With Dependent Component Failures During Early Design StageA Feasibility Study. *Journal of Mechanical Design*, 138(5), 04 2016. 051405.

[54] Yuming Qiu, Ping Ge, and Solomon C. Yim. Risk-based resource allocation for collaborative system design in a distributed environment. *Journal of Mechanical Design*, 130(6), 04 2008.

[55] Herbert A. Simon. The architecture of complexity. *Proceedings of the American Philosophical Society*, 106(6):467–482, 1962.

[56] Sarah A. Sheard and Ali Mostashari. Principles of complex systems for systems engineering. *Systems Engineering*, 12(4), 2009.

[57] Kemper Lewis. Making sense of elegant complexity in design. *Journal of Mechanical Design*, 134(12), 11 2012. 120801.

[58] Jami J. Shah and George Runger. What is in a name? on the misuse of information theoretic dispersion measures as design complexity metrics. *Journal of Engineering Design*, 24(9):662–680, 2013.

[59] GwangKi Min, Eun Suk Suh, and Katja Holtta-Otto. System architecture, level of decomposition, and structural complexity: Analysis and observations. *Journal of Mechanical Design*, 138(2), 2016.

[60] Kaushik Sinha and Olivier De Weck. Structural complexity quantification for engineered complex systems and implications on system architecture and design. 2013.

[61] Ashwin Gurnani and Kemper Lewis. Collaborative, decentralized engineering design at the edge of rationality. *Journal of Mechanical Design*, 130(12), 2008.

[62] Bumsoo Lee, Kenton B. Fillingim, William R. Binder, Katherine Fu, and Christiaan J.J. Paredis. Design heuristics: A conceptual framework and preliminary method for extraction. In *Proceedings of the ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Cleveland, OH, August 6, 2017–August 9, 2017*, 2017.

[63] Elham Keshavarzi, Matthew McIntire, Kai Goebel, Irem Tumer, and Christopher Hoyle. Resilient system design using cost-risk analysis with functional models. In *Proceedings of the ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Cleveland, OH, USA, August 6, 2017–August 9, 2017*, 2017.

[64] Daniel Kahneman. *Thinking, fast and slow*. Farrar, Straus and Giroux, New York, NY, USA, 2011.

[65] Slavisa Tasic. The illusion of regulatory competence. *Critical Review*, 21(4):423–436, 2009.

[66] Hannah S. Walsh, Andy Dong, and Irem Y. Tumer. Towards a theory for unintended consequences in engineering design. In *Proceedings of the Design Society: International Conference on Engineering Design*, volume 1, pages 3411–3420, 2019.

[67] Hannah S. Walsh, Andy Dong, Irem Y. Tumer, and Guillaume P. Brat. Detecting and characterizing archetypes of unintended consequences in engineering design. In *Proceedings of the ASME 2020 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, 32nd International Conference on Design Theory and Methodology*, 2020.

[68] K Marais, J Saleh, and N Leveson. Archetypes for organizational safety. *Safety Science*, 44(7):565–582, 2006.

[69] Douglas L. Van Bossuyt and Ryan M. Arlitt. Toward a functional failure analysis method of identifying and mitigating spurious system emissions in a system of systems. In *ASME 2019 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Volume 1: 39th Computers and Information in Engineering Conference, August 1821, 2019, Anaheim, California, USA*, 2019.

[70] E.F. Wolstenholme. Towards the definition and use of a core set of archetypal structures in system dynamics. *System Dynamics Review*, 9(1):7–26, 2003.

[71] Elham Keshavarzi, Matthew McEntire, and Christopher Hoyle. Dynamic design using the kalman filter for flexible systems with epistemic uncertainty. In *Proceedings of the ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Volume 2B: 41st Design Automation Conference, Boston, MA, USA, August 2, 2016–August 5, 2016*, 2016.

[72] Rose Crossland, Jon H. Sims Williams, and Chris McMahon. An object-oriented modeling framework for representing uncertainty in early variant design. *Research in Engineering Design*, 14(3):173–183, 2003.

[73] Hussein A. Abbass. *Computational Red Teaming*. Springer International Publishing, 2015.

[74] Nancy Leveson. A new accident model for engineering safer systems. *Safety Science*, 42(2):237–270, 2004.

[75] David Rousseau, Julie Billingham, and Javier Calvo-Amodio. Systemic semantics: a systems approach to building ontologies and concept maps. *Systems*, 6(32), 2018.

[76] Mario Bunge. *Emergence and Convergence: Qualitative Novelty and the Unity of Knowledge*. University of Toronto Press, Toronto, Ontario, Canada, 2003.

[77] Mohammad Moshirpour, Nariman Mani, Armin Eberlein, and Behrouz H. Far. Model based approach to detect emergent behavior in multi-agent systems. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013), Ito, Jonker, Gini, and Shehory (eds.), Saint Paul, MN, USA, May 6, 2013–May 10, 2013*, 2013.

[78] W. Ross Ashby. *An Introduction to Cybernetics*. University Paperbacks, Methuen, London, 1964.

[79] S Mittal and L Rainey. Harnessing emergence: the control and design of emergent behavior in system of systems engineering. In *Proceedings of the Conference on Summer Computer Simulation, Chicago, IL, USA, July 26, 2015–July 29, 2015*, 2015.

[80] Terry Bahill. Diogenes, a process for identifying unintended consequences. *Systems Engineering*, 15(3):287–306, 2012.

[81] M Watz and S Hallstedt. Addressing sustainability in product requirements – a systems perspective. In Philip Ekströmer, Simon Schütte, and Johan Ölvander, editors, *DS91: Proceedings of NordDesign: Design in the Era of Digitalization*. The Design Society, 2018.

[82] Douglas L. Van Bossuyt, Bryan M. O'Halloran, and Ryan M. Arlitt. A method of identifying and analyzing irrational system behavior in a system of systems. *Systems Engineering*, 22(6):519–537, 2019.

[83] Sue Yi, Kevin Lumbard, Nicole B. Damen, Matt Germonprez, and Christine A. Toh. Towards an information archetypes framework: Exploring the types of information used in open source design engagements. In *ASME 2019 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Volume 7: 31st International Conference on Design Theory and Methodology, August 1821, 2019, Anaheim, California, USA*, 2019.

[84] Radim Špicar. System dynamics archetypes in capacity planning. *Procedia Engineering*, 69:1350–1355, 2014.

[85] Jay Forrester. System dynamics, systems thinking, and soft OR. *System Dynamics Review*, 10(2–3):245–256, 1994.

[86] Jens Rasmussen. Risk management in a dynamic society: a modelling problem. *Safety Science*, 27(2/3):183–213, 1997.

[87] W. Ross Ashby. Requisite variety and its implications for the control of complex systems. In *Facets of System Science, International Federation for Systems Research International Series on Systems Science and Engineering, vol 7. Springer, Boston, MA*, 1991.

[88] David Rousseau. Three general systems principles and their derivation: insights from the philosophy of science applied to systems concepts. In *15th Annual Conference on Systems Engineering Research, Redondo Beach, CA, Mar 23–25*. 2017.

[89] David Rousseau, Jule Billingham, and Javier Calvo-Amodio. System semantics: a systems approach to building ontologies and concept maps. *Systems*, 6(3):32, 2018.

[90] G. Midgley, I. Munlo, and M. Brown. The theory and practice of boundary critique: Developing housing services for older people. *The Journal of the Operational Research Society*, 49(5):467–478, 1998.

[91] Gerald Midgley and Luis A. Pinzon. Boundary critique and its implications for conflict prevention. *Journal of the Operational Research Society*, 62(8):1543–1554, 2011.

[92] Maurice Yolles. Viable boundary critique. *Journal of the Operational Research Society*, 52(1):35–47, 2001.

[93] Paul Cilliers. Boundaries, hierarchies and networks in complex systems. *International Journal of Innovation Management*, 5(2):135–147, 2001.

[94] Ben-Tzion Karsh and Samuel J. Alper. Work system analysis: The key to understanding health care systems. In Kerm Henriksen, James B Battles, Eric S Marks, and David I Lewin, editors, *From Research to Implementation (Volume 2: Concepts and Methodology)*. Agency for Healthcare Research and Quality, 2005.

[95] Robert Merton. The unanticipated consequences of purposive social action. *American Sociological Review*, 1(6):894–904, 1936.

[96] D.A. Bella and K.J. Williamson. Conflicts in interdisciplinary research. *Journal of Environmental Systems*, 6(2):105–124, 1976.

[97] David Bella. Emergence and evil. *Emergence: Complexity and Organization*, 8(2), 2006.

[98] Saras D. Sarasvathy. Causation and effectuation: Toward a theoretical shift from economic inevitability to entrepreneurial contingency. *Academy of Management Review*, 26(2):243–263, 2001.

[99] Japan Transport Safety Board. Aircraft serious incident investigation report: Air Nippon Co., LTD. JA16AN. Technical Report AI2014-4, Ministry of Land, Infrastructure, Transport, and Tourism, 2014.

[100] K Øien, I.B. Utne, R.K. Tinmannsvik, and S. Massaiu. Building safety indicators: Part 2 application, practices and results. *Safety Science*, 49(2011):162–171, 2010.

[101] R Flin, K Mearns, P O'Connor, and R Bryden. Measuring safety climate: identifying the common features. *Safety Science*, 34(2000):177–192, 2000.

[102] Hannah S. Walsh, Andy Dong, and Irem Y. Tumer. The structure of vulnerable nodes in behavioral network models of complex engineered systems. In *ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Volume 7: 29th International Conference on Design Theory and Methodology, August 69, 2017, Cleveland, Ohio, USA*, 2017.

[103] A. F. Hayes and Klaus Krippendorff. Answering the call for a standard reliability measure for coding data. *Communication Methods and Measures*, 1(1):77–89, 2007.

[104] K. Krippendorff. Reliability in content analysis. *Human Communication Research*, 30(3):411–433, 2004.

[105] Jana Eggink. Krippendorff's Alpha. `https://www.mathworks.com/matlabcentral/fileexchange/36016-krippendorff-s-alpha`, 2020. Accessed: January 22, 2020.

[106] Umut Asan and Secil Ercan. An introduction to self-organizing maps. In Cengiz Kahraman, editor, *Computational Intelligence Systems in Industrial Engineering*, chapter 14. Atlantis Press, 2012.

[107] Jing Tian, Michael H. Azarian, , and Michael Pecht. Anomaly detection using self-organizing maps-based k-nearest neighbor algorithm. In *European Conference of the Prognostics and Health Management Society 2014*, 2014.

[108] Mathworks. MATLAB Version 2019b, 2019.

[109] Teuvo Kohonen. The self-organizing map. *Neurocomputing*, 21(1):1–6, 1998.

[110] 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.

[111] Hannah S. Walsh, Andy Dong, and Irem Y. Tumer. An analysis of modularity as a design rule using network theory. In *Proceedings of the ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, 30th International Conference on Design Theory and Methodology*, 2018.

[112] Nam Suh. *Axiomatic design: advances and applications*. Oxford University Press, 2001.

[113] James Skakoon. *The Elements of Mechanical Design*. ASME Press, 2008.

[114] James Tomayko. Achieving reliability – the evolution of redundancy in american manned spacecraft computers. *Journal of the British Interplanetary Society*, 38:545–552, Dec. 1985.

[115] Carliss Baldwin and Kim Clark. *Design Rules: The Power of Modularity*, volume 1. The MIT Press, 2000.

[116] John K. Gershenson and G. Jagganath Prasad. Modularity in product design for manufacturability. *International Journal of Agile Manufacturing*, 1(1):99–110, 1997.

[117] Carliss Baldwin and Kim Clark. *Modularity in the Design of Complex Engineering Systems*, chapter 6, pages 175–205. Springer, 2006.

[118] Patrick J. Newcomb, Bert Bras, and David W. Rosen. Implications of modularity on product design for the life cycle. *Journal of Mechanical Design*, 120(3):483–490, 1998.

[119] Somwrita Sarkar, Andy Dong, James Henderson, and P.A. Robinson. Spectral characterization of hierarchical modularity in product architectures. *Journal of Mechanical Design*, 136(1):011006, 2013.

[120] Karl Ulrich. *Fundamentals of Product Modularity*, chapter 12, pages 219–231. Springer, 1994.

[121] Katja Hölttä-Otto and Olivier de Weck. Degree of modularity in engineering systems and products with technical and business constraints. *Concurrent Engineering*, 15(2):113–126, 2007.

[122] Yaneer Bar-Yam. When systems engineering fails-toward complex systems engineering. In *IEEE International Conference on Systems, Man and Cybernetics, 2003*, volume 2, pages 2021–2028. IEEE, Washington, DC, 2003.

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

[124] Elham Keshavarzi, Matthew McIntire, and Christopher Hoyle. A dynamic design approach using the kalman filter for uncertainty management. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 31(2):161–172, 2017.

[125] 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.

[126] Hiroaki Kitano. Biological robustness. *Nature Reviews Genetics*, 5:826–837, 2004.

[127] Eum Suyong, Arakawa Shin'ichi, and Murata Masayuki. Toward bio-inspired network robustness - step 1. modularity. In *2007 2nd Bio-Inspired Models of Network, Information and Computing Systems*, pages 84–87. IEEE, 2007.

[128] Tien-Dzung Tran and Yung-Keun Kwon. The relationship between modularity and robustness in signalling networks. *Journal of the Royal Society Interface*, 10(88):20130771, 2013.

[129] Petter Holme. Metabolic robustness and network modularity: A model study. *PLoS ONE*, 6(2):e16605, 2011.

[130] Stafford Beer. Cybernetics–a systems approach to management. *Personnel Review*, 1(2):28–39, 1972.

[131] Albert-László Barabási. *Linked: The new science of networks*. Plume, 2003.

[132] Paul Erdős and Alfréd Rényi. On random graphs. *Publicationes Mathematicae Debrecen*, 6:290–297, 1959.

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

[134] Duncan J Watts and Steven H Strogatz. Collective dynamics of small-worldnetworks. *Nature*, 393(6684):440, 1998.

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

[136] Phillipa Pattison, Stanley Wasserman, Garry Robins, and Alaina Michaelson Kanfer. Statistical evaluation of algebraic constraints for social networks. *Journal of Mathematical Psychology*, 44(4):536–568, 2000.

[137] Réka Albert, Hawoong Jeong, and Albert-László Barabási.

[138] Ricard V. Sole, Ramon Ferrer-Cancho, Jose M. Montoya, and Sergi Valverde. Selection, tinkering, and emergence in complex networks. *Complexity*, 8(1):20–33, 2003.

[139] Hawoong Jeong, B. Tombor, Réka Albert, Zoltan N. Oltvai, and Albert-László Barabási.

[140] Manuel E. Sosa, Steven D. Eppinger, and Craig M. Rowles. A Network Approach to Define Modularity of Components in Complex Products. *Journal of Mechanical Design*, 129(11):1118–1129, 2007.

[141] Mingxian Wang, Zhenghui Sha, Yun Huang, Noshir Contractor, Yan Fu, and Wei Chen. Forecasting technological impacts on customers' co-consideration behaviors: A data-driven network analysis approach. In *ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume Volume 2A: 42nd Design Automation Conference, page V02AT03A040, Charlotte, North Carolina, USA, 2016. ASME. 10.1115/DETC2016-60015.

[142] Songhua Ma, Zhaoliang Jiang, and Wenping Liu. A design change analysis model as a change impact analysis basis for semantic design change management. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 230:1–14, 2016.

[143] Dan Braha and Yaneer Bar-Yam. Topology of large-scale engineering problem-solving networks. 69(1):016113, 2004.

[144] Melanie Mitchell. Complex systems: Network thinking. *Artificial Intelligence*, 170(18):1194–1212, 2006.

[145] Steven H. Strogatz. Exploring complex networks. *Nature*, 410(6825):268+, 2001.

[146] 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.

[147] Manuel Sosa, Jürgen Mihm, and Tyson R. Browning. Degree Distribution and Quality in Complex Engineered Systems. *Journal of Mechanical Design*, 133(10), 2011.

[148] Réka Albert, Hawoong Jeong, and Albert-László Barabási. Error and Attack Tolerance of Complex Networks. *Nature*, 406(July):378–381, 2000.

[149] Hoda Mehrpouyan, Brandon Haley, Andy Dong, Irem Y. Tumer, and Chris Hoyle. Resilient design of complex engineered systems. In *ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE2013)*, volume 3A: 39th Design Automation Conference, page V03AT03A048, Portland, OR, 2013. ASME.

[150] Dharshana Kasthurirathna, Andy Dong, Mahendrarajah Piraveenan, and Irem Y Tumer. The failure tolerance of mechatronic software systems to random and targeted attacks. In *ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE2013)*, volume 5: 25th, Portland, OR, 2013. ASME.

[151] Qi D. Van Eikema Hommes. Comparison and application of metrics that define the components modularity in complex products. In *ASME 2008 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE 2008)*, volume 4: 20th International Conference on Design Theory and Methodology; Second International Conference on Micro- and Nanosystems, pages 287–296. ASME, New York, 2008.

[152] Katja Hölttä-Otto and Olivier de Weck. Metrics for assessing coupling density and modularity in complex products and systems. In *ASME 2007 International Design Engineering Technical Conferences and Computers and*

Information in Engineering Conference, volume 3: 19th International Conference on Design Theory and Methodology; 1st International Conference on Micro- and Nanosystems; and 9th International Conference on Advanced Vehicle Tire Technologies, Parts A and B, pages 343–352. ASME, 2007.

[153] Kaushik Sinha, Eun Suk Suh, and Olivier de Weck. Integrative complexity: An alternative measure for system modularity. *Journal of Mechanical Design*, 140(5):051101, 2018.

[154] Simon Moritz Göhler, Tobias Eifler, and Thomas J. Howard. Robustness metrics: Consolidating the multiple approaches to quantify robustness. *Journal of Mechanical Design*, 138(11):111407, 2016.

[155] Namwoo Kang, Alparslan Emrah Bayrak, and Panos Y. Papalambros. Robustness and real options for vehicle design and investment decisions under gas price and regulatory uncertainties. *Journal of Mechanical Design*, 140(10):101404, 2018.

[156] Thomas Nirmaier, Jerome Kirscher, Zhanat Maksut, Manuel Harrant, Monica Rafaila, and Georg Pelz. Robustness metrics for automotive power microelectronics. *1st RIIF Workshop at DATE13*, 2013.

[157] Marc Manzano Castro. Metrics to evaluate network robustness in telecommunication networks. Technical report, University of Strathclyde, 2011.

[158] M. E. J. Newman. *Networks*. Oxford University Press, New York, New York, 2010.

[159] David Perry. *Aircraft Structures*. Dover Publications, 2011.

[160] Jan Roskam. *Airplane Design, Part 3*. DARcorporation, 1985.

[161] Snorri Gudmundsson. *General Aviation Aircraft Design: Applied Methods and Procedures*. Butterworth-Heinemann, 2013.

[162] Jan Roskam. *Airplane Design, Part 7*. DARcorporation, 1985.

[163] Robert B. Stone, Kristin L. Wood, and Richard H. Crawford. A heuristic method for identifying modules for product architectures. *Design Studies*, 21(1):5–31, 2000.

[164] 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.

[165] Hannah S. Walsh, Andy Dong, and Irem Y. Tumer. The structure of vulnerable nodes in behavioral network models of complex engineered systems. In *Proceedings of the ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, 29th International Conference on Design Theory and Methodology*, 2017.

[166] Hannah S Walsh, Mohammad Hejase, Daniel Hulse, Guillaume Brat, and Irem Y Tumer. Structural consequence analysis: Towards the quantification of component consequential importance in system architecture design. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, 45th Design Automation Conference*, 2019.

[167] Brandon Haley, Andy Dong, and Irem Y. Tumer. Creating faultable network models of complex engineered systems. In *ASME 2014 International Design and Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE 2014)*, volume 2A: 40th Design Automation Conference, page V02AT03A051, Buffalo, NY, 2014. ASME.

[168] Open Source Modelica Consortium. OpenModelica Version 3.2.2. Retrieved from http://openmodelica.org, 2016.

[169] Jun Liu, Qingyu Xiong, Weiren Shi, Xin Shi, and Kai Wang. Evaluating the importance of nodes in complex networks. *Physica A: Statistical Mechanics and its Applications*, 452:209–219, 2016.

[170] Fenghui Zhu, Wenxu Wang, Zengru Di, and Ying Fan. Identifying and characterizing key nodes among communities based on electrical- circuit networks. *PLoS ONE*, 9(6):1–10, 2014.

[171] J. Sklaroff. Redundancy management technique for space shuttle computers. *IBM Journal of Research and Development*, 20(1):20–28, February 1976.

[172] D. Coit. Cold-standby redundancy optimization for nonrepairable systems. *IIE Transactions*, 33(6):471–478, 2001.

[173] Kaushik Sinha, Eun Suk Suh, and Olivier de Weck. Correlating integrative complexity with system modularity. In *ASME 2017 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE 2017)*, volume 2A: 40th Design Automation Conference, page V02AT03A048. ASME, New York, 2017.

[174] Ewen Denney, Ganesh Pai, and Iain Whiteside. Modeling the safety architecture of uas flight operations. In *International Conference on Computer Safety, Reliability, and Security*, pages 162–178. Springer, 2017.