AN ABSTRACT OF THE THESIS OF

Brandon M. Haley for the degree of Master of Science in Mechanical Engineering presented on June 5, 2014.

Title: Evaluating Complex Engineered Systems using Complex Network Representations

Abstract approved: ____________________________________________________________

Irem Y. Tumer

This thesis is the combination of two research publications working toward a unified strategy in which to represent complex engineered systems as complex networks. Current engineered system modeling techniques segment large complex models into multiple groups to be simulated independently. These methods restrict the evaluations of such complex systems as their failure properties are typically unknown until they are experienced in operation. In an effort to combat the computationally prohibitive simulations required for the analysis of complex engineered systems, complex networks are used to simplify the analysis and provide data during early design when costs for design changes and associated risk are lower. The first publication presents a methodology in which to model complex engineered systems as networks so that nodes are commensurate in ontological category under a common analysis goal. The second publication identifies a model scaling technique in which to evaluate network topology metrics for an evaluation of parameterized failure performance. Each publication utilized a drivetrain model to illustrate and simulate the methods and potential results. It was found that a bipartite behavioral network is capable of consistently identifying system failures within network topology. By analyzing complex engineered systems with complex network techniques, an evaluation of system robustness can be developed in an effort to eliminate variation in system performance.
Evaluating Complex Engineered Systems using Complex Network Representations

by

Brandon M. Haley

A THESIS

submitted to

Oregon State University

in partial fulfillment of the requirements for the degree of

Master of Science

Presented June 5, 2014
Commencement June 2014
Master of Science thesis of Brandon M. Haley presented on June 5, 2014.

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

_________________________________
Brandon M. Haley, Author
This thesis is presented in accordance with the Manuscript Document Format option. Two manuscripts are provided. The first was accepted for publication in the 2014 International Design Engineering Technical Conference and the second is to be submitted for publication to the 2015 International Design Engineering Technical Conference. Journal versions of each publication are in preparation for submission in Summer, 2014.
ACKNOWLEDGEMENTS

I express my gratitude and appreciation to Dr. Irem Tumer. Her continued support and faith in me has made it possible to expand my knowledge in an optimal learning environment. I would also like to thank Dr. Andy Dong at the University of Sydney. Like Dr. Tumer, he has challenged me to think critically and to develop opinions and insights of my own accord and defend them. I would like to thank each of the professors in the Design Engineering Lab, and specifically Dr. Chris Hoyle. His classroom guidance inspired many of the thoughts and ideas that have defined my research. I would also like to thank my peers in the Design Engineering Lab and former Complex Engineered Systems Design Lab for providing a great atmosphere to conduct research while providing constructive criticism and critical insights. Last, I would like to thank my girlfriend, Brittany Tomlin; my parents, Alan and Kathy Haley; my siblings, Jon Bolich and Jennifer Haley; and my close friend, Tyler Lewis. Your support and guidance has been truly immeasurable and I appreciate all that you have done for me.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Definitions</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Contributions</td>
<td>4</td>
</tr>
<tr>
<td>2 Background</td>
<td>6</td>
</tr>
<tr>
<td>2.1 Complex Systems</td>
<td>6</td>
</tr>
<tr>
<td>2.2 Complex Networks</td>
<td>7</td>
</tr>
<tr>
<td>2.3 Failure Modeling Techniques for Complex Engineered Systems</td>
<td>10</td>
</tr>
<tr>
<td>2.3.1 Conceptual Design</td>
<td>11</td>
</tr>
<tr>
<td>2.3.2 Validation Design</td>
<td>12</td>
</tr>
<tr>
<td>3 Related Research</td>
<td>15</td>
</tr>
<tr>
<td>3.1 Complex Engineered System Modeling</td>
<td>16</td>
</tr>
<tr>
<td>3.2 Complex Network Models</td>
<td>18</td>
</tr>
<tr>
<td>3.3 Model Evaluation</td>
<td>19</td>
</tr>
<tr>
<td>3.3.1 CES Model Evaluation</td>
<td>19</td>
</tr>
<tr>
<td>3.3.2 CN Model Evaluation</td>
<td>20</td>
</tr>
<tr>
<td>4 Creating Faultable Network Models of Complex Engineered Systems</td>
<td>24</td>
</tr>
<tr>
<td>4.1 Abstract</td>
<td>25</td>
</tr>
<tr>
<td>4.2 Introduction</td>
<td>26</td>
</tr>
<tr>
<td>4.3 Related Work</td>
<td>27</td>
</tr>
<tr>
<td>4.3.1 Complex Systems</td>
<td>27</td>
</tr>
<tr>
<td>4.3.2 Complex Networks</td>
<td>28</td>
</tr>
<tr>
<td>4.4 Mis-Use of Complex Network Models of Complex Engineered Systems</td>
<td>30</td>
</tr>
<tr>
<td>4.5 Modeling Failures in Complex Engineered Systems with Networks</td>
<td>33</td>
</tr>
<tr>
<td>4.5.1 Uni-partite networks</td>
<td>34</td>
</tr>
<tr>
<td>4.5.2 Bi-partite Networks</td>
<td>36</td>
</tr>
<tr>
<td>4.6 Case Studies</td>
<td>39</td>
</tr>
<tr>
<td>4.6.1 The Drivetrain Model</td>
<td>40</td>
</tr>
<tr>
<td>4.6.2 Modeling the Drivetrain with a Uni-partite Network</td>
<td>41</td>
</tr>
<tr>
<td>4.6.3 Modeling the Drivetrain with a Weighted Bi-partite Network</td>
<td>43</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS (Continued)

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.7 Discussion</td>
<td>48</td>
</tr>
<tr>
<td>4.8 Conclusions and Future Work</td>
<td>50</td>
</tr>
<tr>
<td>5 Evaluating Complex Engineered System Failures with Complex Networks</td>
<td>52</td>
</tr>
<tr>
<td>5.1 Abstract</td>
<td>53</td>
</tr>
<tr>
<td>5.2 Introduction</td>
<td>54</td>
</tr>
<tr>
<td>5.3 Related Work</td>
<td>55</td>
</tr>
<tr>
<td>5.3.1 Complex Systems</td>
<td>56</td>
</tr>
<tr>
<td>5.3.2 Complex Networks</td>
<td>57</td>
</tr>
<tr>
<td>5.3.3 Failure Modeling</td>
<td>58</td>
</tr>
<tr>
<td>5.4 Methodology</td>
<td>60</td>
</tr>
<tr>
<td>5.4.1 Bipartite Network Development</td>
<td>61</td>
</tr>
<tr>
<td>5.4.2 Average Shortest Path Length</td>
<td>63</td>
</tr>
<tr>
<td>5.4.3 Network Diameter</td>
<td>64</td>
</tr>
<tr>
<td>5.4.4 Robustness Coefficient</td>
<td>64</td>
</tr>
<tr>
<td>5.5 Case Study</td>
<td>66</td>
</tr>
<tr>
<td>5.5.1 The Drivetrain Model Simulation</td>
<td>67</td>
</tr>
<tr>
<td>5.5.2 Modeling the Drive-train with a Bipartite Network</td>
<td>70</td>
</tr>
<tr>
<td>5.5.3 Network Evaluation of Failure</td>
<td>72</td>
</tr>
<tr>
<td>5.6 Discussion and Conclusion</td>
<td>74</td>
</tr>
<tr>
<td>5.7 Future Work</td>
<td>76</td>
</tr>
<tr>
<td>6 Conclusion</td>
<td>77</td>
</tr>
<tr>
<td>7 Research Shortcomings and Future Applications</td>
<td>80</td>
</tr>
<tr>
<td>7.1 Usability and Scalability</td>
<td>80</td>
</tr>
<tr>
<td>7.2 Failure Definition</td>
<td>82</td>
</tr>
<tr>
<td>7.3 Additional Modeling</td>
<td>82</td>
</tr>
<tr>
<td>7.4 Implementable Evaluation Framework</td>
<td>83</td>
</tr>
<tr>
<td>Bibliography</td>
<td>85</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>United/Continental Airline Merger Network (Source: United Airlines Presentation) [1]</td>
<td>8</td>
</tr>
<tr>
<td>2.2</td>
<td>Yeast-Protein Interaction Network [10]</td>
<td>9</td>
</tr>
<tr>
<td>3.1</td>
<td>Chart of Related Research Work</td>
<td>16</td>
</tr>
<tr>
<td>4.1</td>
<td>Modelling hierarchy for networks</td>
<td>30</td>
</tr>
<tr>
<td>4.2</td>
<td>Example seven node network graph</td>
<td>34</td>
</tr>
<tr>
<td>4.3</td>
<td>Adjacency matrix (A) for sample 7 node uni-partite network</td>
<td>35</td>
</tr>
<tr>
<td>4.4</td>
<td>Bi-partite network. Two node groups, three sub-nodes within type-1 and four sub-nodes within type-2.</td>
<td>37</td>
</tr>
<tr>
<td>4.5</td>
<td>Bi-partite adjacency matrix (A) for network in Figure 5.1</td>
<td>37</td>
</tr>
<tr>
<td>4.6</td>
<td>Drivetrain model constructed in OpenModelica [90].</td>
<td>40</td>
</tr>
<tr>
<td>4.7</td>
<td>Uni-partite network of drivetrain</td>
<td>43</td>
</tr>
<tr>
<td>4.8</td>
<td>Bipartite network of drivetrain</td>
<td>44</td>
</tr>
<tr>
<td>4.9</td>
<td>Degree Centrality for Drivetrain Variables</td>
<td>46</td>
</tr>
<tr>
<td>4.10</td>
<td>Degree Centrality for Drivetrain Functions</td>
<td>47</td>
</tr>
<tr>
<td>4.11</td>
<td>Eigenvector Centrality for Drivetrain Variables</td>
<td>47</td>
</tr>
<tr>
<td>4.12</td>
<td>Eigenvector Centrality for Drivetrain Functions</td>
<td>48</td>
</tr>
<tr>
<td>5.1</td>
<td>Bipartite network. Two node groups, three sub-nodes within type-1 and four sub-nodes within type-2.</td>
<td>61</td>
</tr>
<tr>
<td>5.2</td>
<td>Bipartite adjacency matrix (A) for network in Figure 5.1</td>
<td>62</td>
</tr>
<tr>
<td>5.3</td>
<td>Drivetrain model constructed in OpenModelica [90].</td>
<td>67</td>
</tr>
<tr>
<td>5.4</td>
<td>Simulation Results of a Drive-train. Nominal (Blue). Slipping Clutch (Red)</td>
<td>69</td>
</tr>
<tr>
<td>5.5</td>
<td>Bipartite network of drivetrain</td>
<td>71</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>5.6</td>
<td>Adjacency Matrix for Nominally Operating Drive-train</td>
<td>72</td>
</tr>
<tr>
<td>5.7</td>
<td>Adjacency Matrix for Failed Drive-train Clutch</td>
<td>72</td>
</tr>
<tr>
<td>5.8</td>
<td>Network Topology Results for Failed $\mu$</td>
<td>73</td>
</tr>
<tr>
<td>5.9</td>
<td>Network Topology Results for the Independent Failure of all Nodes</td>
<td>74</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Terminology Utilized Throughout This Work</td>
<td>3</td>
</tr>
<tr>
<td>3.1</td>
<td>Types of Network Centrality Measures</td>
<td>20</td>
</tr>
<tr>
<td>4.1</td>
<td>Variable Value Definitions [90, 37]</td>
<td>42</td>
</tr>
<tr>
<td>5.1</td>
<td>Variable Value Definitions [90, 37]</td>
<td>69</td>
</tr>
</tbody>
</table>
Chapter 1: Introduction

In 2010, Qantas flight 32, flown with a large-body Airbus A380 aircraft, traveling from Singapore to Sydney, was forced to make an emergency landing at Changi Airport, Singapore [18]. The unscheduled landing was necessary as a result of an explosion of the number two engine turbine disk. Shrapnel from the explosion damaged neighboring components and subsystems which hindered aircraft performance and safety. Following the incident, an investigation by the Australian Transport Safety Bureau (ATSB) resulted in a redesign recommendation to the manufacturer. Rolls Royce was to redesign the engine turbine disk so that the component could withstand the high pressure unbalanced forces associated with turbine overspeed conditions.

The recommendation and subsequent remedial action illustrates a basic problem in the design of complex engineered systems: monitor the behavior of components so that their failures do not produce adverse interactions with other components or sub-systems. It is the type of recommendation that underscores a key finding in complex network research: the vulnerability of a system is at least equally dependent on the topology of the system, taking into account the connectivity of the components, as the components themselves [4]. It is for this reason that design for reliability and evaluations of system performance under failure become so vitally important.

1.1 Motivation

Complex engineered system function, morphology, and failure behavior are highly researched topic areas. Typically, knowledge about one could not be known without an understanding of all the others. It is for this reason that the modeling of failure behavior becomes of such interest for researchers. Failures typically manifest within the performance envelopes of complex engineered systems [72]. When combined with the types of interactions which are exhibited by complex systems, a failure can cause crippling performance
loss in a system. Such interactions might affect system components directly associated to a failed component through a physical connection. Others, like non-conductive heat transfer applications, might affect neighboring components spatially. Regardless, understanding how a failure might affect the performance of the system depends on how a system is designed.

These reasons make complex network modeling approaches attractive. Complex network theory is typically used as a means for evaluating topology. In simplistic terms, identifying how objects within some greater system are connected can provide immediate information regarding the structural stability and resistance to network attack for that system. This kind of thinking has classically been used within transportation systems such as airline, rail, and shipping networks. An evaluation of global and local cluster topology provide the method for this evaluation.

Current research at the intersection of complex engineered system failure analysis and complex network theory suggests that similar properties which define vulnerability in network topology could be used to define vulnerabilities in complex engineered systems from the introduction of failures. When combined with computationally less expensive analysis tools, such as robust topology analysis, evaluations of centrality, algebraic topology tools, and network viral propagation evaluations to be defined in the upcoming sections, the use of complex network models of complex engineered systems is appealing. This becomes vitally important because the models used within model-based design simulations of complex engineered systems are often incomplete, segmented, and computationally prohibitive compared to network evaluations.

1.2 Definitions

Many of the terms utilized within this research have alternate meanings when used in different domains. Therefore, it is necessary to define several terms which are utilized prominently. Table 1.1 provides a list of the definitions as they are used in this work. Each definition is supported within the engineering literature. The list is not meant to provide authoritative restrictions on the use of any one term outside of this research.
<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliability</td>
<td><em>Reliability</em> relates to the operation of a system for a given amount of time.</td>
</tr>
<tr>
<td>Robustness</td>
<td><em>Robustness</em> suggests an invariance in design performance from variations in input parameters caused by uncertain internal and/or external stimuli.</td>
</tr>
<tr>
<td>Resiliency</td>
<td><em>Resiliency</em> is persistence (or recovery) of system functionality despite the existence of a failure.</td>
</tr>
<tr>
<td>Flow</td>
<td>A <em>Flow</em> is an energy, material, or signal acted on by a function.</td>
</tr>
<tr>
<td>Function</td>
<td>A <em>Function</em> is the transformation of a flow.</td>
</tr>
<tr>
<td>Behavior</td>
<td><em>Behavior</em> is how a function is achieved.</td>
</tr>
<tr>
<td>Performance</td>
<td><em>Performance</em> is a specific manifestation of behavior.</td>
</tr>
<tr>
<td>Failure</td>
<td>A <em>Failure</em> is the degradation of performance.</td>
</tr>
<tr>
<td>Fault</td>
<td>A <em>Fault</em> is a component state affecting component behavior which differs from a nominal state.</td>
</tr>
<tr>
<td>Failure Flow</td>
<td>A <em>Failure Flow</em> is a degradation of a flow so that a fault exists.</td>
</tr>
<tr>
<td>Component</td>
<td>A <em>Component</em> is the lowest level of a system with behavior.</td>
</tr>
<tr>
<td>Emergent Interaction</td>
<td>An <em>Emergent Interaction</em> is an interaction between system elements that is not predictable based on component architecture.</td>
</tr>
<tr>
<td>System</td>
<td>A <em>System</em> is a collection of elements, components or otherwise, which combine to perform a function.</td>
</tr>
<tr>
<td>Complex System</td>
<td>A <em>Complex System</em> is a system displaying emergent interactions that are not separable without altering system architecture.</td>
</tr>
<tr>
<td>Complex Engineered System</td>
<td>A <em>Complex Engineered System</em> (CES) is an artificial complex system comprised of interdependent engineered systems performing a function.</td>
</tr>
<tr>
<td>Network</td>
<td>A <em>Network</em> is a collection of interconnected entities.</td>
</tr>
<tr>
<td>Complex Network</td>
<td>A <em>Complex Network</em> (CN) is a network without purely regular or random topological organization.</td>
</tr>
</tbody>
</table>
It is important to note that the research presented here focuses on concepts of robustness. While there are more than likely applications which can be drawn towards broader reliability and resiliency topics, the presented evaluation of complex networks is indicative of the topological robustness within the functional relationships defining system behavior and performance.

1.3 Contributions

This thesis presents two manuscripts which describe a process for converting a model of a complex engineered system into a complex network and the evaluation of that model for failure effects. This was done to achieve a common standard for the conversion of complex engineered system models to complex networks could be developed. The contributions to the field are as follows:

1. *The development of a hierarchy for potential modeling choices according to a modeling goal.*
   A hierarchy of modeling choices was developed that identifies an appropriate level of nodal abstraction for use on network models of complex engineered systems. Different modeling choices represent the various abstractions to be used with different modeling goals, such as topology evaluation, failure evaluations, module evaluation, or interdependent network evaluation.

2. *The development of the behavioral network.*
   In order to evaluate a complex engineered system for potential failure effects, a level of node abstraction was developed such that the information required to represent the failure is captured within the network model, just as it would need to be captured in the system model. This node abstraction resulted in the creation of the bipartite network model of a complex engineered system, or *behavioral* network.

   In order to determine the locations of failures within a complex network model of a complex engineered system, common network centrality tools were extrapolated
and applied to the bipartite network model. This enabled the identification of failure locations (Degree Centrality) within either node group as well as the effect of the failure on connected topology (Eigenvector Centrality).

4. *The development of a process to evaluate the dynamic change of complex engineered system network topology when under the influence of failures.*

In order to compare the output of a complex engineered system model, nominally operating or failed, to the current topology of a complex network, methods for evaluating the topological properties of networks were applied to the bipartite network model. Such methods allow for the dynamic identification of failures within network topology in order to compare the results to model outputs for the complex engineered system.
Chapter 2: Background

Before proceeding, a discussion of whether complex engineered systems satisfy the requirements of complex systems is included. This discussion is then expanded to include the types of behavioral understanding for complex engineered systems that a complex network approach brings.

The following sections present information related to the identification and analysis of complex systems. Following is a discussion on complex network theory, including the applications towards large-scale system infrastructure and uses in identifying failures. Then, a discussion of failure modeling techniques has been included in an effort to identify the current methodologies utilized for the evaluation of failures to complex engineered systems.

2.1 Complex Systems

Complex systems are types of systems with specific sets of defining attributes. The differentiation between a complex [17] and complicated [26] system often leads to confusion about the nature of being ‘complex’. In a complex system, system interactions and relationships are not separable so long as system behavior and function is retained [17]. A complicated system is one that may be large, have many components, or be difficult to build [26].

While it is true that many complex systems are technological in nature, many aspects of biology and biological processes could be considered complex systems. Ecosystem interactions, brain chemistry and structure, and immune responses, to name a few, are complex systems [60].

Many social groups could be considered a complex system as well. While the interactions between people are at times complicated, the existence of a group structure that is dependent on individuals belonging to and interacting with multiple groups may make
these types of interactions complex. While complex systems could be broken down into additional groups, a broader category of complex engineered systems could encompass many, if not all, of the remaining systems.

Complex engineered systems share properties with other complex systems such as emergent behaviors and interactions between components that cannot be fully understood and modeled [81, 59]. However, some authors argue that even very complicated systems such as aircraft carriers are not actually complex [66]. This research assumes that complex engineered systems are indeed complex systems. Although the systems engineering of these systems is thought to be purely rational, properties of self-organization in the design process [66] would indicate that the resulting system design will share an important property with complex networks: the systems will have architectures and behaviors that are neither truly regular nor random, and are dependent upon connection patterns. Additionally, as an engineered system is an organized combination of hardware, software, people, policies, and procedures, the unpredictable interacting nature of all parties creates unforeseen interactions within system architecture, thereby making it complex. The terminology, ‘complex engineered systems’, is used to distinguish them from other complex systems (such as biological ones) to emphasize that they are artificial, intended to perform specific functions, and comprised of interdependent engineered systems. Examples include aircraft, process plants, and satellites [12].

2.2 Complex Networks

Complex networks are networks wherein the topological features of the network are non-trivial. That is, connections between nodes are neither random nor are they regular [70]. There is no predictable pattern governing how a network may be topologically organized.

Networks are represented by a series of nodes with associated edge connections representing an interaction or relationship between nodes, physical or otherwise. Importantly, all nodes are considered to be zero-dimensional objects of similar ontological category. That is, in a network model of airports, both Hartsfield-Jackson Atlanta International Airport (ATL) and Buffalo Niagara International Airport (BUF) would be modeled as a node
even though ATL is clearly much larger in terms of land area than BUF. Figure 2.1 displays an example of an airline network. Figure 2.2 displays a yeast-protein interaction network [10].

The end goal is to show how the topology defined within the network influences the vulnerability of the system to attacks. For example, in Figure 2.1, how many edges must be removed prior to isolation of an airline hub from the rest of the network? Conversely, how would the removal of important nodes, such as San Francisco International Airport (SFO) or Chicago O’Hare International Airport (ORD), impact travel to and from other highly connected airports? Figure 2.2 describes a yeast-protein interaction network wherein the loss of connections means a disassociation between yeast and protein, thereby creating an incomplete biological process.

Figure 2.1: United/Continental Airline Merger Network (Source: United Airlines Presentation) [1]
Figure 2.2: Yeast-Protein Interaction Network [10]

If a network contains nodes of a single type, it is called a uni-partite network; if the nodes are of two types, it is called a bi-partite network. The premise of network theory is that to understand how a system works, such as the air transport system in the US, we must understand the network structure of the system. We transfer this basic premise toward understanding the behavior of complex engineered systems.

Network representations of complex engineered systems are becoming increasingly popular amongst researchers looking for new ways to identify and design against system failures [51, 67]. Complex network theory examines the topology of a network in order to gain structural information. This information is then used to identify weak system topology and to redesign in an effort to increase quality [83]. However, the differentiation between
a complex network and a complex engineered system presents challenges, because a complex engineered system is not a complex network *per se*. Complex engineered systems are not generally constructed as networks because there are many heterogeneous parts (compared to the air transport network wherein all nodes are conceptually the same). In addition, they are not typically represented in the same manner as complex networks. Because of this, some researchers have avoided using complex network models of complex engineered systems [38]. Despite these deficiencies, however, the inherent benefits for emergent interaction detection, analysis of failure effect, and topology analysis make network analysis useful for the reliability analysis of complex engineered systems.

2.3 Failure Modeling Techniques for Complex Engineered Systems

There is a growing desire amongst researchers to advance the identification of failures and evaluations of reliability into earlier stages of the design process [44]. This is so that potential design changes gleaned as a result of performing the analyses are less costly from a risk mitigation standpoint compared to changes acquired from an established design or from physical testing. Complex engineered systems in the early stages of the design process benefit greatly from such an analysis. Because these systems are becoming so complex, identifying the behavioral patterns of these systems which result in potential fault behavior becomes of great importance. This is exactly what complex network theory provides. Network models provide increased awareness to topological faults. Therefore, potential faults caused by emergent behavioral interactions are observable and can be corrected prior to the physical implementation of the system.

The two most highly utilized design stages are conceptual design (early) and validation design (late). The following sub-sections highlight several commonly used methodologies from the engineering literature for evaluating designs in both the early and late design stages.
2.3.1 Conceptual Design

For issues with reliability, common historically utilized methodologies include the Taguchi Method [89, 44, 7] and Design for Six Sigma [61, 44]. Both methods strive to increase system or design reliability by eliminating the effects of variance from the performance of the system. Other methodologies have been produced which handle the variance within component and module level behavior.

The Function Failure Design Method (FFDM) [85, 86, 44] is a method used to connect system faults to a loss of function in the hopes of identifying potential system failures. After developing a functional model of a system, the model is used to generate potential component implementations for functions. This can then be evaluated with historical failure data to make predictions of probable component failure modes. Additionally, another methodology that is used is the Risk in Early Design (RED) [28, 29, 44] method. Here, function-failure likelihood and consequence information is collected for a risk assessment of high and low risk combinations of function and failure. Both FFDM and RED are highly dependent on the availability of historical failure data. Therefore, evaluations of new designs may not be possible as the data is too highly coupled to existing system configurations. Additionally, the methods do not consider the affects of multiple failures, cascading failures, or the impact of system faults caused by component interactions.

The Functional-Failure Identification and Propagation (FFIP) [87, 47, 57, 45, 46, 56, 44] method was produced in order to understand failure behavior based on component, function, and nominal/not-nominal behavior implementations. The goal of the method is to develop an understanding of failure behavior by mapping failure propagation paths with component ‘states’ and function ‘health’. The approach uses a simulation to determine various fault propagation paths. This provides designers with the ability to analyze function and component failures as well as reason about downstream effects of those failures during critical event scenarios which challenge device functionality.

All of these methods are bottom-up approaches which identify the contributions of component states and health to a greater system state. Because of this, each method can help determine likely failure occurrence data associated with evaluations of reliability and potential propagation paths for faults (FFIP). These techniques have proven useful when
information of failure modes is available or predictable. However, they are not readily useful regarding the interaction effects of failures. In addition, their effectiveness is questioned for the identification of the most vulnerable components without significant prior knowledge of the system or specific failures. In any case, significant expertise of engineers and a knowledge-base of current and historical systems and failures is required for any of the methods outlined here.

2.3.2 Validation Design

As mentioned in the previous sub-section, evaluations of a design during the validation stage are generally performed on established design or prototypes. At this stage, evaluations of a design are performed in order to determine if the design meets key reliability, functional, and failure criterion. The two most common methods utilized are a Failure-Modes and Effects Analysis (FMEA) [34, 21, 22, 44] and a Reliability Block Diagram (RBD) [52, 44] analysis. Both methods have many extensions to provide additional reliability, functional, performance, or failure-rate information for a design. The underlying theme behind these methods is a heavy reliance on empirical, historical data, or expert knowledge on a specific component or system.

To predict and avoid failures of important components, setting aside the problem of identifying them, the engineering design literature suggests techniques such as a FMEA [34, 21, 22, 44] (or FFIP should the need arise during conceptual design). In a FMEA, knowledge from engineers and historical data combine to predict the effect on a system given the introduction of a failure. Depending on the type of FMEA, the failure could manifest as a function failure or component failure. Regardless, the process produces a table which identifies a set of faults that are rank-ordered based on a combination of failure severity, likelihood of occurrence, and detectability. Producing an FMEA is a rather tedious process, yet the results typically represent well-thought out evaluations of system behavior given the presence of faults. The process has been widely utilized in many industries, including automobile manufacturing and aerospace. A criticality analysis can be added to an FMEA so that the effects of failures can be quantified for system-level hazards like loss of life or mission failure. In addition to a heavy reliance on expert opinion and detailed system
descriptions, a FMEA is not readily useful for handling multiple and cascading faults. This negates the usefulness of the method during early conceptual design.

A RBD [52, 44] analysis is a graphical methodology meant to represent system architectural topology and component dependencies with failure rate information. A RBD is easily compared to a complex network model of a complex engineered system with component node abstraction. A RBD is a set of blocks connected in series or parallel and are meant to illustrate the physical make-up of a system. Blocks are connected in series when system operation necessitates performing every function. They are connected in parallel if not all functions are necessary for operation (redundant components). Mathematically, a RBD can be broken down into constituent block diagrams connected in series. This way, reliability and potential failure information can be calculated based on the modeling rule used during the diagrams creation.

A fault tree can be created from a RBD. A Fault Tree Analysis (FTA) [68, 44] is a way of representing the paths in a system which lead to undesired system states caused from failures. Using this approach, branches from root cause component failures to system level impacts are discernible. The analysis is performed by utilizing a series of logic gates defining the movement of a fault through a system. The paths incorporate probabilities so that a stochastic evaluation of some high level event can be calculated given some level of fault independence or correlation. Many researchers criticize the use of fault trees with complex systems because of their inherent need to also be large for an accurate representation of the system. As an example, fault trees of large scale complex systems may have hundreds or thousands of different component states and logic gates defining how a fault moves through the system. Additionally, because a fault tree is only valid for the specific case in which the system is being used, several large fault trees may be necessary to accurately represent a complex system and how a failure propagates through that system.

In summary, many of the approaches commonly utilized by engineers to design complex engineered systems are so highly reliant on experts, historical data, or detailed system descriptions that many of them are prohibitive to use on complex engineered systems, especially as they continue to grow in size and complexity. Finding new ways to model these systems and evaluate them for failures without the need for such detailed descriptions
and knowledge will help engineers to evaluate failure risk and design better systems in the future.
Chapter 3: Related Research

The modeling of complex engineered systems and the evaluation of failures are highly researched areas of interest within the engineering literature. This includes everything from performance evaluations to sensitivity analyses with input variables under uncertainty. The main objective associated with this research is to develop a methodology in which to evaluate the reliability of a complex engineered system and the effect of potential failures to system operation.

As a result of completing the two manuscripts included in this document, multiple areas of engineering design have been explored, specifically Complex Engineered System Modeling, Complex Network Development, and Model Evaluation (Figure 3.1). These areas describe the categories under which the design of large-scale complex engineered systems are considered.
Figure 3.1: Chart of Related Research Work

Under each category are sub-topics that have been explored through this research. The following sections describe each category and sub-topic with respect to the research framework presented here.

3.1 Complex Engineered System Modeling

Prior to creating network models and conducting performance evaluations, an understanding of complex engineered systems is acquired (CES Model, Figure 3.1, top left corner). There are many ways to model complex engineered systems, including agent-based models [58], fractals [74], and model-based design techniques [90]. For the purpose of this research, model-based design techniques were chosen because they contain the specific functional and behavioral relationships required to capture degradation-type system failures.
not captured with other techniques [72, 71].

For model-based design techniques, a model of the system is created which is capable of simulating system performance or function. These models are at times segmented and combined to form a single model, although they do not need to be [90]. System models are then simulated in order to gain an understanding of performance envelopes prior to physical implementation of the system (CES Model Evaluation, Figure 4.10, bottom left corner).

At times, models such as this are so large and complex that they are too computationally expensive to create, let alone simulate. In this case, smaller sub-systems are created and simulated to gather information about the systems, leaving a level of uncertainty from the lack of a complete model simulation.

Complex engineered systems display emergent interactions. That is, interactions which are not necessarily modeled or expected couple a set of components or sub-systems together. The degree of this coupling, or complexity as it is sometimes called, is difficult to determine because of the associated ambiguity related to the unknown component couples. Coupling has many different forms and definitions. In this research, coupling is described by the relationship between system variables and functions [88]. Quantifying this assessment of coupling can be difficult; however Höltä-Otto et al. [41] addressed the issue with a Design Structure Matrix (DSM) system model and a modularity analysis. Modularity is defined as a group of components that are more tightly connected within the module and loosely connected to the rest of the system [41]. The locations and size of system modules as well as the off-diagonal DSM entries which define inter-module connectedness then play a role in quantifying how coupled the system is.

After a model has been developed, the resulting model parameters are used to define the node abstraction to be used in a complex network model of the system (CN Model Development, Figure 3.1, top right corner). Additionally, should the model be sufficiently complete, a simulation of system performance can be conducted (CES Model Evaluation, Figure 3.1, bottom left corner).
3.2 Complex Network Models

Once a model of a complex engineered system is developed, a complex network representation of that model is created which describes that system in a framework of nodes and edges (CN Model Development, Figure 3.1, top right corner). Such networks can then be evaluated in any number of ways, including centrality analysis [40, 93, 76, 23, 20], algebraic topology [43, 42], viral propagations [36, 64, 65], and robust topology analysis [76] that are potentially easier to perform and less costly computationally when compared to traditional system approaches. These approaches will be discussed further in the Model Evaluation section.

Networks are typically simple to conceptualize, simple to create, and simple to computationally evaluate. The biggest challenge associated with the utilization of complex network tools for the evaluation of complex engineered systems is the level of abstraction to assign nodes so that a specific modeling goal can be reached [35, 63, 65]. An evaluation of degradation-type failures is not possible at a physical abstraction of system topology (i.e. components as nodes with edge connections defined by the physical connection between two components) [35]. This is because failures to system performance manifest within the behavioral properties of a system. This information is not present with an abstraction of physical interaction. However, physical architecture is useful when examining differing spatial layouts or how faults could propagate through the physical connection of multiple components. Additionally, a network type (such as unipartite, bipartite, etc) should be chosen along with node abstractions such that sufficient detail is present within the network to capture failure behavior.

After a node abstraction is chosen and a network is created, the network should be evaluated (CN Model Evaluation, Figure 3.1, bottom right corner). For a network evaluation of system failures, as is the case in this research, failures are noticeable by changes in network topology. Topological changes can be quantified with standard network evaluation tools. In some cases, it may be necessary to make changes to a network model to ensure the proper information to capture a failure is present.
3.3 Model Evaluation

Design evaluations of topology and performance between complex networks and complex engineered system models provide the insight for making improvements to system design. It is the evaluation of model sets that allow for the comparison between networks and systems. If a model shows a 50% reduction in system performance, the results from the network model should show a corresponding benchmarked result from changes in topology. Because of this, it is very important to identify those metrics or methods which allow for the evaluation of complex network models of complex engineered systems, regardless of the modeling goal (failure modeling, performance, resiliency to failures, etc.). Such network evaluation tools as those discussed in the previous section describe methods in which to analyze a network topology for failures.

3.3.1 CES Model Evaluation

The two manuscripts presented as a result of this research describe the process of converting a complex engineered system model to a complex network [35] and then evaluating that network for failures. After this, design optimization and edge weight studies are conducted to determine the best alternatives to the current models for improved system robustness [35].

For complex engineered system model evaluation (CES Model Evaluation, Figure 3.1, bottom left corner), evaluations of performance allow designers to benchmark results between different designs and then select the design which offers the highest performance, reliability, or robustness. This can be done any number of different ways, but typically in model-based design techniques an evaluation of system torque, horsepower, RPM, fuel efficiency, etc, as well as its performance invariability from input uncertainty can provide an indication of the strength of a design.

Once an evaluation of the complex engineered system model has been produced, results must be compared to complex network model results to determine the validity of the network model (CN Model Evaluation, Figure 3.1, bottom right corner).
3.3.2  CN Model Evaluation

Failures to complex network models of complex engineered systems are represented by changes in network topology. Because of this, metrics which describe the topological makeup of a complex network are utilized to determine the extent of a failure to a complex engineered system.

3.3.2.1  Centrality

Network centrality is a measure of connectedness between nodes. Centrality tracks the topology of a network node-to-node. That is, measures of centrality track the influence an individual node has on the overall topology of the network. There are many types of centrality measures, each quantifying a different aspect of how a network is constructed topologically. Table 3.1 describes a subset of the centralities utilized in this research.

<table>
<thead>
<tr>
<th>Centrality Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Degree</td>
<td>The number of times one node is connected to another node.</td>
</tr>
<tr>
<td>Eigenvector</td>
<td>Describes the amount of influence one node has on the overall topology of a network.</td>
</tr>
<tr>
<td>Closeness</td>
<td>Describes the shortest distance from one node to every other node.</td>
</tr>
<tr>
<td>Betweenness</td>
<td>How often one node acts as a bridge along the shortest path between two other nodes.</td>
</tr>
</tbody>
</table>

There are many additional types of centrality, including Katz centrality and Farness, for example. For the purpose of this research, degree and eigenvector centrality were the most highly utilized measures of network centrality for the direct transfer of topological understanding for the evaluation of complex engineered systems because of their direct applicability towards topological organization and structure [35]. Topologically, this allows
for the identification of important nodes and then for network evaluation in the event of an attack on those nodes [35, 76].

3.3.2.2 Algebraic Topology

Algebraic topology measures provide methodologies for the evaluation of robustly connected topologies [42, 43]. Essentially, these measures track how difficult it is to disconnect a network by removing edges. Such measures as the algebraic connectivity are based on spectral measures of network topology to determine how network connectedness may or may not be natively resistant to attack. It can be shown that as the algebraic connectivity, a spectral measure of topological robustness, increases, the difficulty for a network to become disconnected also increases [65, 63]. This is said to be directly related to network robustness because higher algebraic connectivities generally result in higher average degree centralities, a feature which is directly correlated to robust network topologies.

However, there are problems with this thinking. Because the algebraic connectivity is a degree-based measure of global topology, it is not suitable for identifying robust topology for networks that are highly module [65, 63]. This makes global evaluations of robust topology inaccurate because inter-module (local) clusters carry a larger percentage of more highly connected nodes.

3.3.2.3 Topological Modularity

The topological modularity of networks is a growing area of research. Modularity describes how clustered into groups nodes are. A highly module network is one that has clusters of highly connected nodes. Sparse intermodule edges then provide the relationships between modules. Network modularity provides ideas about how a network is organized which does not necessarily include node-to-node organization. Sarkar et al. [78, 79, 80] developed an approach to identify clustered ‘communities’ within networks. Such ‘communities’ display high amounts of intermodule connectivity and fewer extra-module connections. Further, as many ‘communities’ intersect each other, a level of hierarchical modularity is introduced not generally captured by a typical modularity analysis. This allows
designers to observe the types of dependencies associated with large-scale networks.

By observing how a module network architecture breaks down and distintegrates during random, targeted, or sustained attacks, a native resistance to attack can be mapped from the changes in network architecture. Applied to a complex engineered system, this type of ‘community’ detection provides an avenue to explore the emergent interactions caused by the interdependent engineered systems which make up a complex engineered system.

3.3.2.4 Viral Propagations

Network viral propagation tools are used to determine how network topology can influence the spread of a virus. From the literature on network theory [64, 4], attacks on nodes in networks can be viewed as a viral infection transmittable to neighboring nodes on the premise of connected edge propagation. Because of this, researchers evaluate network topologies for the ease of propagation along topological features [42]. This has been done multiple ways; most noticeably through models of network infection rate and Susceptible-Failed-Fixed (SFF) models of node densities [64]. These models allow for the evaluation of a network in terms of the topological properties of node clusters, specifically those highly connected nodes which transmit attacks rapidly and the lowly connected, yet highly redundant nodes which bottleneck attacks from propagation through the topology of a network.

When applied to complex network models of complex engineered systems which are modeled with physical connections between system components, topology has a profound affect on how a fault is transmitted through a system from an initially faulted node (component) [64, 65]. While there is some debate on how an initially faulted component in a system is capable of transmitting such component-specific faults through topology, the evaluations of such propagation patterns provide insight into how robust a system is to the spread of faults and how to better design system architecture to resist those fault propagation patterns.
3.3.2.5 Robust Topology

As an extension of viral propagation metrics, many researchers have begun to explore and quantify how the topology of a network disintegrates under sustained attack (node deletion, in this case). Piraveenan et. al. [76] developed the robustness coefficient as a means of quantifying topological resistance to network disintegration under sustained attack. Measures such as this examine topology locally so that highly concentrated areas of the network do not skew the result when examined globally.

Other distance related topology metrics can provide insight into the robustness of network topology. Network diameter provides a global evaluation of network size. Average shortest path length evaluates the network considering local topology as well. While each does not necessarily provide a direct indication of topological robustness, variations of the metrics between nominal and faulted cases can provide insights into the robustness of complex engineered systems.
Chapter 4: Creating Faultable Network Models of Complex Engineered Systems

Authors
Brandon M. Haley
102 Dearborn Hall
Email: haleybr@onid.oregonstate.edu

Andy Dong
University of Sydney
Email: andy.dong@sydney.edu.au

Irem Y. Tumer
Covell 116
Email: irem.tumer@oregonstate.edu

Proceedings of the 2014 ASME International Design & Engineering Technical Conferences
40th Design Automation Conference (DAC)
IDETC 2014
August 17-20, 2014, Buffalo, NY, United States of America

Journal version in preparation to be submitted Summer of 2014
4.1 Abstract

This paper presents a new methodology for modeling complex engineered systems using complex networks for failure analysis. Many existing network-based modeling approaches for complex engineered systems “abstract away” the functional details to focus on the topological configuration of the system and thus do not provide adequate insight into system behavior. To model failures more adequately, we present two types of network representations of a complex engineered system: a uni-partite architectural network and a weighted bi-partite behavioral network. Whereas the architectural network describes physical inter-connectivity, the behavioral network represents the interaction between functions and variables in mathematical models of the system and its constituent components. The levels of abstraction for nodes in both network types affords the evaluation of failures involving morphology or behavior, respectively. The approach is shown with respect to a drivetrain model. Architectural and behavioral networks are compared with respect to the types of faults that can be described. We conclude with considerations that should be employed when modeling complex engineered systems as networks for the purpose of failure analysis.

Keywords: Complex Systems, Complex Networks, Product Architecture
4.2 Introduction

A basic problem in the design of complex engineered systems is to engineer and monitor the behavior of components so that their failures do not produce adverse interactions with other components or sub-systems. It is the type of recommendation that underscores a key finding in complex network research: system failure is at least equally dependent on system topology and intermediate relationships as a (weak) component [4].

In order to understand the behavior of complex systems, and in particular the relation between structure and failure behavior, researchers have modeled them as networks [2, 67]. Applied studies on the failure of real-world systems include the US air transport network [93] and the North American power grid [54]. Complex engineered systems have also begun to be modeled as networks [80, 83]. Researchers have applied social networks analysis and complex networks theory to analyze the statistical properties of very large scale design products and engineering projects. They have shown that their network models have structural (architectural) properties that are like those of other biological, social, and technological networks [15, 16]. Methods from complex network and graph theory provide intriguing tools to evaluate the robustness and resiliency characteristics of complex engineered systems [3]. Centrality and viral propagation metrics, which have been used to determine the vulnerability of software architecture to the propagation of a virus [43, 42], have been applied to understand the failure tolerance of complex engineered systems due to the propagation of a fault, such as the loss of flow or energy [63, 64]. The underlying idea in the application of complex network methodologies to understand the failure behavior of complex engineered systems is to use network failure metrics to evaluate their system architecture in the early stages of the design process so as to create increasingly robust architectures. A robust system architecture is one that is able to continue operating as intended given the presence of variations to internal or external operating conditions. This may include expected component failures and system-level effects.

In this paper, we question the appropriateness of complex networks as a formalism for modeling complex engineered systems and for understanding behavioral properties such as robustness. In particular, we focus on the assumptions underlying the model. Choosing the appropriate level at which to model a system is an important issue regardless of
the modeling methodology. An appropriate level of detail is required in order to capture the information relevant to the model and to derive behavioral understanding from the model [38].

Because a network model of physical architecture only captures the topological properties of a system, only failures involving this type of abstraction can be usefully modeled. Such failures include broken physical links between mechanical components, electrical short and open circuits, or eliminated flow in fluid systems. A larger variety of failures, such as degraded frictional coefficient, faulty sensor resulting in incorrect system state determination, etc., would require a node abstraction level capable of capturing behavior.

In an effort to expand the application of complex network theory to the evaluation of complex engineered systems, this paper compares and contrasts two types of network representations of the same mechanical system to identify the appropriate uses of the models.

4.3 Related Work

Before proceeding, we begin with a discussion of whether ‘complex’ engineered systems satisfy the requirements of ‘complex systems’ and the type of understanding of the behavior of complex engineered systems that a complex network approach brings.

4.3.1 Complex Systems

Complex systems are a branch of systems with a specific set of defining attributes. The differentiation between complex and complicated systems often leads to confusion about the nature of being ‘complex’. A complex system is a system of systems. Specifically, the interaction or relationship with one system or an aspect of a system to any other aspect of a system is not separable [17]. A complicated system is one that may be large, have many components or parts, or be difficult to build [26]. While most of these attributes are true also of complex systems, this idea of non-separable interacting features or relationships is not present in complicated systems. Complex systems are present in many domains, including inside and outside of the engineering disciplines [60].

While complex engineered systems share properties with other complex systems such
as emergent behaviors and interactions between components that cannot be fully understood and modeled [81, 59], some authors argue that even very complicated systems such as aircraft carriers are not actually complex systems [66]. We use the terminology complex engineered systems to distinguish them from other complex systems (such as biological ones) to emphasize that they are artificial, intended to perform specific functions, and comprised of interdependent engineered systems. Examples include aircraft, process plants, and satellites [12]. Further, we agree that complex engineered systems are indeed complex systems. Although the systems engineering of these systems is thought to be purely rational, properties of self-organization in the design process [66] means that the resulting system design will share an important property with complex networks. The systems will have architectures and behaviors that are neither truly regular nor random, and are dependent upon the pattern of connections.

It is this property that makes complex engineered systems suitable for analysis using a complex network approach. In a complex engineered system, multiple parts, which may be entire sub-systems in their own right, come together to perform a greater function. However, the patterns of connections between components and their physical behaviors could not be reasonably thought of as ‘regular’ or fully predictable. Understanding their behavior requires an understanding of both the constituent components and the patterns of connections between components. Representing the patterns of connections and interactions in a complex engineered system could provide insight into its behavior, because it is the pattern of connections rather than the connections or components themselves that have a significant effect on the behavior of the system [69]. The question that this paper raises, though, is what connections and patterns matter in understanding the failure of complex engineered systems? Before we address this question, we briefly review what is known about how patterns of connections influence the behavior of complex networks with a focus on their failure tolerance.

### 4.3.2 Complex Networks

Complex networks are networks wherein the topological features of the network are non-trivial. That is, connections between nodes are neither random nor are they regu-
lar [70]. There is no predictable pattern governing how a network may be topologically organized. Networks are represented by a series of nodes with associated edge connections representing an interaction or relationship between nodes, physical or otherwise. Importantly, all nodes are considered to be zero-dimensional objects of similar ontological category. That is, in a network model of airports, both ATL and BUF would be modeled as a node even though ATL is clearly a much larger airport in terms of land area than BUF. If a network contains nodes of a single type, it is called a uni-partite network; if the nodes are of two types, it is called a bi-partite network. The premise of network theory is that to understand how a system works, such as the air transport system in the US, we must understand the network structure of the system. We transfer this basic premise toward understanding the behavior of complex engineered systems.

Network representations of complex engineered systems are becoming increasingly popular amongst researchers looking for new ways to identify and design against weak aspects [51, 67] of systems. In theory, the idea is promising. Complex network theory examines the topological architecture of a network in order to gain functional information related to structure and morphology. Computationally tractable metrics describing network topology such as node degree (the number of nodes to which a node is connected) can provide insight into the overall network robustness [3]. This information has already been used to identify the weak aspects of a system and to redesign the system in an effort to increase its quality [83] and to understand the degree of coupling (and hence ‘complexity’) of the system [62, 6]. However, in practice, the differentiation between a complex network and a complex engineered system presents challenges, because a complex engineered system is not a complex network per se. Complex engineered systems are neither constructed as a network because there are many heterogeneous parts (compared to the air transport network wherein all nodes are conceptually the same) nor are they typically represented in the same manner as complex networks. Some researchers have frowned on using complex networks because of these deficiencies [38].
4.4 Mis-Use of Complex Network Models of Complex Engineered Systems

When modeling complex engineered systems, many researchers have not stopped to consider the validity of their modeling choices. Assumptions have been made to apply the network modeling methodology to the physical architecture when, in fact, physical architecture does not always capture enough information to draw informed conclusions about system-level behavior.

In terms of physical architecture, Figure 4.1 displays a hierarchy of the possible levels of granularity at which to model a complex engineered system as a network. Within reason, the physical architecture of a complex engineered system could be modeled at every level except the molecular or sub-atomic levels. This is especially important as it is unlikely that useful detail related to system-level and performance failures would be obtained from such a low granularity modeling layer, except in MEMS or nanosystems.

Figure 4.1: Modelling hierarchy for networks

One of the major issues with modeling complex systems (complex engineered systems,
especially) is the level of abstraction in node declaration [38]. At the moment, the predominant level of granularity is the component level, that is, components as nodes [78, 80, 83, 64]. No rules have been provided, though, to determine whether to model, e.g., a bolt and the plate to which it is connected as two nodes or a single node, for example. Similarly, relations between components are modeled as (binary) edges, but a fixed connection or a frictional connection are modeled identically, despite their differing behaviors. Choosing the wrong level of granularity in the representation distorts the results of architectural analysis [19]. It also fails to model the right kind of failure.

A node as component abstraction is not necessarily the correct approach to model failures. For example, in a network representation of an aircraft, the entire wing could be represented as a single node, as could the fuselage, landing gears, and other ‘components’, which are entire sub-systems in and of themselves. Clearly, this model is unlikely to yield any profound failure insights other than that the failure of any of these nodes would be catastrophic at a system level. An automobile is not effectively modeled with system components chosen as nodes if the desire is to model potential component failures, either. A network representation that includes a differential, axle, wheel bearing, wheel, and tire is not sufficient for modeling failures to the bearing because the model lacks the internal physics-based or geometric relations governing bearing behavior [37]. The most this model can do is describe the physical architecture of the system. Should the network be modeled such that behavior information for the bearing were present, such as fluid dynamics for lubrication and heat dissipation, a network model of the automobile may then be appropriate. Conversely, power grids are more aptly modeled with component-level abstractions because power generation sub-stations could be considered commensurate entities [54]. A complex network representation can thus be easily mis-used to model potential component and system-level failures.

Because networks are modeled based upon the underlying principle that all nodes are equivalent, it is important to note that the model should be abstracted sufficiently for both network architecture purposes and so that sufficient information is incorporated into the model to fulfil a modeling goal, such as the modeling of failures. In the examples above, the network models were mis-used because the nodes in the aircraft and automobile were
not commensurate. To be commensurate, nodes must share ontological, categorical, or functional similarities. In addition, nodes may be commensurate if they share a commonality according to the goal of the model (for instance, to show connected components or spatial layout). Failures to these systems are described with dislocated edges in networks because the physical architecture describing the grid is of highest interest.

While the component level of abstraction is appropriate for modeling the failure of components when they ‘break’ and lose physical separation, many failures manifest at a behavioral level.

An example of a failure at a behavioral level is torque loss when a bolt becomes loose. This may lead to other problems such as vibration in the system. Degraded performance failures can manifest themselves as material wear, among other things [72, 37]. While material wear is caused as a result of a physical connection between two components or parts, the change in that property is of more interest than the connection itself. In essence, the mechanisms for which failures are produced must be included in the model. Examples include parameters for viscosity in fluid friction failures or friction coefficients in mechanical slipping failures. Otherwise, the information describing the failure is not present and the failure cannot be modeled.

In these cases, a modeling decision should be made that captures enough information so that the failure and the effect on the system can be captured. To accomplish this, the component or part level behavior must be modeled so that it can be altered to reflect the existence of a failure. In the next two sections, we describe two network models of a complex engineered system, an architectural network and a behavioral network, the type of faults that can be identified by each model, and the type of metrics that can be applied to identify failures.

It is important to note that other research has been conducted on the development of complex engineered system to complex network modeling relations. As noted by Alfaris et al. [5] and Hamraz et al. [36], the development of multi-layer Domain Mapping Matrices (DMM) and Design Structure Matrices (DSM) provides an avenue to represent physical architecture, behavior, and an array of input parameters within the same layered network. As a result, off-diagonal matrix blocks would represent the relationships present between
the various layers and could provide information about the system. However, in the case of behavioral failure identification and analysis, networks including extraneous model layers represent an unneeded computational expense.

4.5 Modeling Failures in Complex Engineered Systems with Networks

We present a methodology to model complex engineered systems as complex networks for the purpose of analyzing their failure properties. Uni-partite architectural networks and bi-partite behavioral networks are first represented as adjacency matrices before a network model is produced. In the determination of the proper network type to implement, consider the following:

1. Is the analysis of morphology or function?
2. What constitutes commensurate node abstraction?
3. Will failures be tested?

The type of analysis is most important and will guide network determination for each additional item. If structure and morphology information is desired, an architectural network (uni-partite) must be used. Architectural networks are easily mapped to existing model-based designs. This allows for the evaluation of designs for spatial considerations to determine highly connected, therefore highly important, system components. Identifying those components helps designers mitigate risks associated with the potential failure of that highly utilized component. If functional or behavioral information is required (information beyond the existence of a component), a behavioral network (bi-partite) must be used. Some model-based design software implementations (e.g., Modelica® [90]) provide model instantiation features allowing for the access of model behavior equations. Behavioral networks model components based on the mathematical description of the underlying physics governing their behavior. This is useful because while the location of a failure is often within a component, some variables, such as a heat transfer coefficient, ‘cross’ component boundaries.
In some cases, a weighted network may be useful if the network has a connection or set of connections wherein the strength of connectivity is relevant. For instance, the adjacency matrix indices for the network representation of a transportation system could be weighted with the distance between cities (nodes) or the amount of traffic on a route (edge) between two cities. The formulation of un-weighted and weighted versions of uni-partite and bi-partite graphs is presented next.

4.5.1 Uni-partite networks

Network representations involving physical architecture have typically been constructed using uni-partite networks; we call these architectural networks. Figure 4.2 is a visualization of a sample seven node uni-partite network. The nodes, X1 through X7, are represented by uni-partite relationships. Each node is its own entity and has edge connections determined by physical connections within the system or some other relationships. These networks are represented by square symmetric adjacency matrices that can be manipulated by edge properties. Figure 4.3 is the un-weighted adjacency matrix associated with the network in Figure 4.2. The following rules are applicable when using a uni-partite network representation of a complex engineered system:

1. A node (using whichever selected hierarchy from Figure 4.1) must not be connected to itself. This means that the adjacency matrix will have a diagonal of zeros.

2. A system must be connected as a single large component. However, it need not be
fully connected (every node need not be connected to every other node). Every node should have at least one connection to at least one other node.

3. There can be no isolated groups of connected nodes.

![Figure 4.3: Adjacency matrix (A) for sample 7 node uni-partite network.](image)

The adjacency matrix, $A$, in Figure 4.3 is defined as follows:

$$A_{ij} = \begin{cases} 1 & \text{if there is a connection between node } i \text{ and } j \\ 0 & \text{otherwise} \end{cases}$$ (4.1)

For the weighted case, $A$ is defined as:

$$A_{ij} = \begin{cases} w_{ij} & \text{if there is a connection between node } i \text{ and } j \\ 0 & \text{otherwise} \end{cases}$$ (4.2)

To test the failure tolerance of a network, the basic strategy is to remove a component (node) from the system (network) and then measure changes in topological properties [3]. Researchers often employ specific ‘attack’ strategies such as randomly removing a node (random attack) or removing the node having the ‘importance’ (centrality attack).

The disruption to the system caused by the failed node is measured by various metrics of topological properties such as the network diameter (the average length of the shortest paths between any two nodes in the network) and the size of the largest component (the number
of connected nodes). In addition, evaluations of node degree, eigenvector, and various path length centrality metrics are commonly used. Many metrics have been proposed to measure the connectedness of the network after the removal of a node. These metrics include algebraic connectivity [43] and a topological robustness coefficient [76], and each has limitations. Our point here is not to review the limitations of the topological properties, but to question the type of faults that a uni-partite network can and cannot model, especially with respect to complex engineered systems.

The strategy for testing the failure tolerance of complex networks may overstate the failure likelihood of a complex engineered system. This modeling assumption would fail to capture the situation when the flow is not actually disrupted. Rather, there is only a degradation in the nominal flow between the two nodes. Likewise, it would completely underestimate failure in situations when flows cannot be stopped, such as a valve that is stuck open. An internal component failure could not be handled the same way, either. The failure of a component does not necessarily mean that the component is suddenly ‘lost’. Rather, the component no longer functions as intended, which could manifest itself in various ways. An example of this is a stuck valve refusing to open or close or a sensor that transmits incorrect data causing the control logic to issue an incorrect directive. In these cases, it would be incorrect to model the failed node by the disappearance of that node. It is not that the network methodology is incorrect; rather, the architectural network fails to model these types of faults. For these types of faults, a bi-partite network, or behavioral network, is more appropriate.

4.5.2 Bi-partite Networks

Network representations involving relationships between two distinct node types are represented with a bi-partite network.

Differing from uni-partite architectural networks, bi-partite behavioral networks show the relationship between two distinctly separate types of nodes. An example of a bi-partite network is displayed in Figure 5.1. The example has two node types with three sub-nodes within type one and four sub-nodes within type two. The individual sub-nodes are then connected with edges across each node type. Unlike uni-partite networks, bi-partite net-
Figure 4.4: Bi-partite network. Two node groups, three sub-nodes within type-1 and four sub-nodes within type-2.

Works are represented with rectangular, non-symmetric adjacency matrices. The matrices can be weighted or un-weighted. An adjacency matrix for the example bi-partite network displayed in Figure 5.1 has been provided in Figure 5.2.

\[
\begin{bmatrix}
1 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1
\end{bmatrix}
\]

Figure 4.5: Bi-partite adjacency matrix (A) for network in Figure 5.1.

The following rules are applicable when using a bi-partite network:

1. No sub-node can be connected to any other sub-node from the same node type.
2. A sub-node from one type must be connected to at least one sub-node from the other node type.

In determining the bi-partite adjacency matrix, a rectangular matrix is used with size \( m \times n \).

\[
M = \text{Number of Sub Nodes in Type-1}
\]

(4.3)
\( N = \text{Number of Sub Nodes in Type-2} \) \hfill (4.4)

The adjacency matrix, \( A \), in Figure 5.1 is defined as follows:

\[
A_{ij} = \begin{cases} 
1 & \text{edge connection between sub node } i \text{ and } j \\
0 & \text{otherwise}
\end{cases}
\] \hfill (4.5)

For the weighted case, \( A \) is defined as follows:

\[
A_{ij} = w_{ij} \quad \text{edge connection between sub node } i \text{ and } j
\] \hfill (4.6)

As discussed previously, the proper modeling of a complex engineered system requires an appropriate level of node abstraction to capture sufficient modeling information.

In terms of intermediate system failures, or failures within a system, a level of node abstraction that captures behavioral information is required [72]. This bi-partite behavioral network approach captures the information for systems of all sizes by relating function (type-1 nodes) to variables, parameters, and constants (type-2 nodes) defining that function. The weighted representation allows us to take behavioral properties into account by assigning nominal functional values as edge weights. As we discuss next, the edge weights provide a mechanism to model faultable behavior.

As with uni-partite networks, the failure tolerance of bi-partite networks could be simulated by attacking nodes (removing nodes) and then calculating topology-based metrics such as the topological robustness coefficient.

In relation to behavioral failure, though, failure simulation strategies based on the complete loss of variables or functions are not useful in understanding how the behavior of the system dynamically changes when the operational modes of components fluctuate. Instead, the dynamics of system change are influenced by the behavioral network’s topology (i.e., the pattern of connections between the underlying physics of the system). Fluctuations outside of desired ranges would be considered failures. Of equal interest, though, is the insensitivity of the system to “drifts” despite changes to function or variable values.
This is a measure of system robustness. Many dynamical processes, such as collective out-of-equilibrium dynamics, occurring on complex networks have properties that can be explained via the spectrum (the set of eigenvalues of the network’s adjacency matrix) of the network [11]. Because the dynamic behavior of each component in the architectural network is governed by the relations in the behavioral network, changes to the operational modes of components would be reflected by commensurate changes to variable or function values, which themselves become reflected in the edge weights. Changes in edge weight would manifest themselves as changes to the network’s eigenvalues. Changes to network properties based upon the eigenvalues of the network can thus reflect “drifts” in function or variable values, which in turn are influenced by the behavioral network’s topology.

In addition to examining the change in eigenvalues, failure tolerance of bi-partite behavioral networks can be analyzed through the principle of modularity, that is, isolation of faults in independent modules. As other research has shown, the patterns of connections between functions and variables in the behavioral network, and not the mathematical relations per se, is sufficient to determine the extent to which the system’s behavior can be decomposed into independent modules [79]. Thus, identifying modules in the bi-partite behavioral network can help to identify which functions and variables can be isolated from faults if those functions and variables are not embodied in the same part.

In the case study, we will show the influence of the eigenvector centrality. Eigenvector centrality is a measure of the relative importance of each node in a network relative to surrounding nodes. Tracking topological changes with eigenvector centrality is common practice in the literature on network theory [31, 84, 20, 23]. Changes in importance indicate changes to the network topology due to changes to the edge weights.

4.6 Case Studies

This section presents a drivetrain modeled with OpenModelica, a Modelica® software platform. The model is used to show how a uni-partite network modeling approach at the component level is insufficient to model faulty behavior within the system. For comparison, a weighted bi-partite network is used on the same system to show how behavioral information is implemented. An eigenvector centrality analysis is performed on the bi-partite
network in order to illustrate the “drifts” in network topology (discussed earlier) that may be indicative of a failure. We do not suggest this eigenvector centrality analysis as the sole way of quantifying the effect or magnitude of a failure, but rather a way of identifying changes in edge weight that are potentially indicative of system-wide changes. Following the analysis will be a discussion on the potential methodologies for failure analysis given the restrictions inherent to network analysis tools.

4.6.1 The Drivetrain Model

The drivetrain model illustrated in Figure 5.3 depicts a drivetrain that accepts a constant torque and clutch state input. This information (energy) is then passed through a clutch, a bearing, and two gearing manipulations under an inertial load. The equations governing the behavior in the model have been presented below in Equations 5.12-5.18. Table 5.1 provides the definitions for the variables used in Equations 5.12-5.18.

\[
F_1 : Clutch_{fric} = mu \times cgeo \times Fn
\]  

(4.7)
\[ F_2 : cgeo = N \times \frac{1}{2} \times (r_o + r_i) \]  
\[ F_3 : Clutch_{out} = Torque_{in} - Clutch_{fric} \]  
\[ F_4 : 0 = Clutch_{out} + bearing_{out} - tau - fric_{viscous} \]  
\[ F_5 : fric_{viscous} = 10^{-7} \times f_o \times (nu \times RPM)^{2/3} \times dm^3 \]  
\[ F_6 : 0 = ratio1 \times bearing_{out} + gear_{1_out} \]  
\[ F_7 : 0 = ratio2 \times gear_{1_out} + gear_{2_out} \]

The equations represent faultable behavior, that is, changes in the values of the parameters which may lead to incorrect performance of the drivetrain. This is true for each function aside from Equation 5.13. In this equation, a parameter, \( cgeo \), defining the geometric properties of a clutch has been included. While not strictly information related to behavior, this equation defines clutch properties which make a clutch subject to internal contact mechanism failures, such as clutch slipping. This is an important failure related to clutch behavior not directly captured with typical network modeling techniques [72].

### 4.6.2 Modeling the Drivetrain with a Uni-partite Network

The first step in developing an architectural network is to determine the level of granularity for the architecture of the complex engineered system under analysis. This will determine the proper level of node abstraction to construct a network. As shown in Figure 5.3, the drivetrain model is constructed at the component level with models of the drivetrain’s constituent components. These include a clutch, bearing, and two gears. Equations 5.12-5.18 represent the defining behaviors of each constituent model at a level of node abstraction not accessible in a uni-partite network because there are multiple types and varieties of nodes present at a behavioral level. Because the general premise of network theory is an examination of the pattern of connections between nodes of commensurate value, a comparison between variables such as operational torque values and material parameters.
Table 4.1: Variable Value Definitions [90, 37]

<table>
<thead>
<tr>
<th>Variable</th>
<th>Related Component</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Clutch_{fric}$</td>
<td>Clutch</td>
<td>Amount of Torque from Friction</td>
</tr>
<tr>
<td>mu</td>
<td>Clutch</td>
<td>Static Coefficient of Friction</td>
</tr>
<tr>
<td>cgeo</td>
<td>Clutch</td>
<td>Geometric Constant for Clutch Surface</td>
</tr>
<tr>
<td>$F_n$</td>
<td>Clutch</td>
<td>Normal Force Between Clutch Plates</td>
</tr>
<tr>
<td>N</td>
<td>Clutch</td>
<td>Number of Frictional Surfaces</td>
</tr>
<tr>
<td>$r_o$</td>
<td>Clutch</td>
<td>Outer Surface Diameter</td>
</tr>
<tr>
<td>$r_i$</td>
<td>Clutch</td>
<td>Inner Surface Diameter</td>
</tr>
<tr>
<td>$Clutch_{out}$</td>
<td>Clutch</td>
<td>Torque Output from Clutch</td>
</tr>
<tr>
<td>$Torque_{in}$</td>
<td>Clutch</td>
<td>Torque Input to Clutch</td>
</tr>
<tr>
<td>$bearing_{out}$</td>
<td>Bearing</td>
<td>Torque Output from Bearing</td>
</tr>
<tr>
<td>tau</td>
<td>Bearing</td>
<td>Bearing Torque Loss</td>
</tr>
<tr>
<td>$fric_{viscous}$</td>
<td>Bearing</td>
<td>Bearing Torque from Viscous Friction</td>
</tr>
<tr>
<td>$f_o$</td>
<td>Bearing</td>
<td>Bearing Type Factor</td>
</tr>
<tr>
<td>$nu$</td>
<td>Bearing</td>
<td>Kinematic Viscosity of Lubricant</td>
</tr>
<tr>
<td>RPM</td>
<td>Bearing</td>
<td>Shaft Speed</td>
</tr>
<tr>
<td>dm</td>
<td>Bearing</td>
<td>Bearing Diameter</td>
</tr>
<tr>
<td>ratio1</td>
<td>Gear 1</td>
<td>Gear Ratio for Gear 1</td>
</tr>
<tr>
<td>$gear1_{out}$</td>
<td>Gear 1</td>
<td>Torque Output from Gear 1</td>
</tr>
<tr>
<td>ratio2</td>
<td>Gear 2</td>
<td>Gear Ratio for Gear 2</td>
</tr>
<tr>
<td>$gear2_{out}$</td>
<td>Gear 2</td>
<td>Torque Output from Gear 2</td>
</tr>
</tbody>
</table>

would not be appropriate. This differs from bi-partite networks in that sub-nodes from one node type are not directly connected to any other sub-node in the same node type. The relationship connects variable to function, not variable to variable. The same argument could be made for comparing components as commensurate entities. A clutch and bearing are fundamentally different components. However, because they share a physical connection, they could be seen as commensurate nodes. Figure 4.7 displays the uni-partite network representation for the drivetrain model in Figure 5.3.
4.6.3 Modeling the Drivetrain with a Weighted Bi-partite Network

To develop a weighted bi-partite network, the functional relationships defining component or system behavior are required. In the case of this drivetrain, these relationships have been reproduced in Equations 5.12-5.18 and were available through model instantiation in OpenModelica. Each equation is a function belonging in node type-1, $F_1$ through $F_7$. Each constituent variable belongs in node type-2, $x_1$ through $x_{20}$. Therefore, to construct the bi-partite network for this drivetrain, there are 7 sub-nodes within node type-1 (functions) and 20 sub-nodes within node type-2 (variables). A connection between node types will occur if a type-2 design variable node is present within the type-1 function node. By constructing a network this way, further system information describing the operation of a system is captured. Figure 5.5 displays the bi-partite network for the drivetrain.

To fully utilize network theory, the impact of each design variable should be included within the network. With an unweighted approach, each connection between node type-1 and node type-2 is unweighted according to Equation 5.3. In complex engineered systems, failures involving system performance manifest by altering parameter values present in type-2 nodes [72]. For instance, if the coefficient of friction present in function 1 (Equation 5.12) were to be halved and all remaining variables held constant, the clutch friction would also be halved. In order to capture the fault information, the adjacency matrix which represents the bi-partite graph in Figure 5.5 should be weighted according to the rules identified in Equation 5.4 with the values of the variables. This retains all information necessary to create a faultable network model.

When analyzing a failure in the network model, regardless of architectural or behavioral type, changes in network topology manifest in the adjacency matrix representations of the model. Dissociations between nodes or changes in edge weight describe the failure. For in-
Figure 4.8: Bipartite network of drivetrain

stance, in an architectural model, a removed edge between between two nodes is indicative of a dissociation between those nodes. It is for this reason that connections to highly connected nodes become important. Centrality metrics identify these changes. Additionally, viral propagations are able to spread more rapidly through areas of high connectedness. In
behavioral networks, system failures beyond the physical association between components are possible. Dissociations or changes in edge weight manifest in similar ways as architectural networks. Because edge connections are cross-modal in behavioral networks, the effect on constituent variables or connected functions is observable. Suppose a clutch failure were implemented into the un-weighted architectural network depicted in Figure 4.7. Without any additional information, this failure would result in the dissociation of the “clutch node” with all surrounding nodes. This disconnects the network. Architecturally, this is important. As the components are aligned in series, a failure to the clutch means lost (or reduced) functionality. However, a more descriptive failure mode, such as clutch slipping, is not observable in an architectural network because the information is not present within the network. This is not the case in a behavioral network. From literature, a slipping clutch displays reduced performance according to the frictional capabilities of the rotating disk material [72]. This failure is caused by a reduced frictional coefficient [72], which is included within the behavioral model. The failure is observable as a reduction in internal connectedness local to the clutch. Complete network topology is not lost.

Eigenvector centrality can be used as a way of tracking changes to edge weight (and their impact on network topology). Changes in eigenvector centrality can be used to show how “drifts” in variable value could be indicative of a complex engineered system failure. In order to show how changes in edge weight can be observed, an analysis was performed on the underlying network topology of the drivetrain model. Because the actual operational parameter values of the network are not important for the execution of this exercise, we assign edge weights randomly distributed from one to one hundred on a uniform distribution to the network. While certainly not representative of parameter values for a complex engineered system, the produced variation will suffice. For consistency, we implement a failure within the geometric coefficient of the clutch, $c_{geo}$ by halving the randomly assigned edge weight corresponding with all connections between node group one and the $c_{geo}$ node within node group two.

Below are four plots, two for degree centrality and two for eigenvector centrality (one of each for each node type). One graph of degree centrality and one of eigenvector centrality will display centralities for seven nodes (Fig. 4.10,4.12), while the other set shows
centralities for twenty nodes (Fig. 4.9,4.11). Each figure has a node number (corresponding to the order of nodes presented in Table 5.1) along the horizontal axis versus an eigenvector centrality value on the vertical axis. As degree centrality only shows the local connectedness for a node, the degree centrality figures illustrate the location of the induced failure within the set of variable and function nodes. Blue data sets represent nominal values prior to failure implementation. Red data sets represent centralities after failure.

Figure 4.9: Degree Centrality for Drivetrain Variables
Figure 4.10: Degree Centrality for Drivetrain Functions

Figure 4.11: Eigenvector Centrality for Drivetrain Variables
Figures 4.11 and 4.12, representing eigenvector centrality, are indicative of correlations between nodes. Both the function and variable eigenvector centralities indicate a strong correlation between both node groups. This can be seen by examining Figure 4.12 and noting that the change in edge weight for \( cgeo \) produced a large change in functional (behavioral) network topology. While this does not necessarily provide an indication of the magnitude of failure, these results do confirm traditional network thinking: changes to, or attacks on, highly connected nodes have profound impacts on network topology regardless of architecture type. The same generalization can be made of complex engineered systems because changes to highly integrative aspects of a system can have profound impacts on correlated components and sub-systems.

4.7 Discussion

Uni-partite networks have been shown effective at representing the physical architecture of systems. This is especially true as system and network sizes increase and the patterns of connections become relevant in understanding system behavior. For instance, large scale
models of the Internet and power grids are effectively modeled as uni-partite networks because the representative nodes in each system could be considered commensurate. This is not true of other types of complex engineered systems, especially those that are made up of heterogeneous component and sub-system types. When the aim of the network model is to understand the patterns of connections of the components, such as architectural modularity [80], then a component level abstraction is appropriate. However, when the aim is to understand failure properties, an architectural network may not be the appropriate model. We have presented two modeling approaches based upon considerations associated with the type of system failure the engineer aims to understand.

The underlying idea behind the models is to produce two network representations of the complex engineered system. While we have not shown the following, it is possible to model the interaction between the architectural and behavioral network using the ‘network of networks’ approach [24] and to simulate failures by varying their mutual interdependencies [50]. We note that failure analysis methods such as the FFIP framework [56] model the behavior of the system using a component oriented modeling approach. However, in addition to the component model, high-level, qualitative behavior models of components at various discrete nominal and faulty modes are necessary to model potential system failures. This limits the usefulness of such an approach when the system becomes large and complex such that the qualitative behavior models can no longer capture interaction effects. This is not the case with behavioral or architectural networks. The information required to produce both network types is inherently present within the system models engineers already use. This is an inherent advantage to the methodology presented because the types of performance modeling conducted during early conceptual design stages can be done concurrently with network topology and failure tolerance analysis.

Modeling the system is only part of the problem relating complex engineered systems and complex networks. The utilization of complex network tools must also be appropriate to evaluate failures within complex engineered systems. Two common network tools are centrality and viral propagation metrics. Centrality metrics are measures of node connectivity. The number of times a node is connected to other nodes (called degree centrality) or the degree in which there are shared parameters between multiple functions are examples
of centrality metrics. Eigenvector centrality and eigenvalue spectra can also be used. Analyzing the centrality properties of a network model of a complex engineered system can help to determine highly connected nodes. These have a greater impact on how network topology can change, as was observed in the eigenvector centrality study. By determining the eigenvector centrality of a nominally operating complex engineered system, it is possible to identify the functions and variables of the system that are potentially problematic because they influence the behavior of many parts. These tools could be used to help determine the components of a system that are most susceptible to the effects of a failure and the components that would propagate significant faults should they fail.

4.8 Conclusions and Future Work

Because of the principal advantages associated with complex network theory for the failure analysis of complex engineered systems, this paper presented two issues requiring research:

1. **At what fidelity can a complex engineered system be modeled as a network in order to capture true systemic behavior?**

2. **When is a fidelity choice sufficient and when is it not?**

For a network model to satisfy both issues, a node abstraction level must be used that captures the predominant information required in the complex engineered system model. The fidelity choice is satisfied when a balance is reached between the level of node abstraction chosen and the behavioral simulation goal of the network model. A weighted bi-partite network approach was produced in order to create commensurate nodal relationships between functions and parameters. This network would then be capable of producing faulted network models for the design and evaluation of complex engineered systems.

While the drivetrain model used in this paper is useful for explaining the approach and to justify the methodology, in future work, we will analyze a large scale system to determine if the approach is scalable. A sensitivity analysis will be conducted on large bi-partite weighted adjacency matrices to determine how uncertainty in design variable values (edge
weight) will impact the output and uncertainty of centrality and viral propagation metrics. In addition, future work will also focus on creating network motifs that represent recurrent architectural patterns in complex engineered systems. Such synthetic models induced from previously-modeled complex engineered systems substitute for detailed models when real-world operational ranges of parameter values for large-scale complex engineered systems are difficult to know during early system design.
Chapter 5: Evaluating Complex Engineered System Failures with Complex Networks

Authors
Brandon M. Haley
102 Dearborn Hall
Email: haleybr@onid.oregonstate.edu

Andy Dong
University of Sydney
Email: andy.dong@sydney.edu.au

Irem Y. Tumer
Covell 116
Email: irem.tumer@oregonstate.edu

To Be Submitted to the 2015 ASME International Design & Engineering Technical Conferences
41st Design Automation Conference (DAC)
IDETC 2015
Date and Location To Be Determined

Journal version in preparation to be submitted Summer of 2014
5.1 Abstract

This paper describes a modeling approach derived from complex network theory applied to complex engineered systems. This approach is based on the evaluation of a bipartite ‘behavioral’ network and is shown on a Modelica drive-train model. A network was created based on the model and was evaluated. The goal of the research was to show how a system under the influence of a failure can be modeled as a network and to identify network metrics capable of capturing the failure. Network metrics describing the topological robustness of a network were used to evaluate the system for failure effects. These metrics, average shortest path length, network diameter, and a Robustness Coefficient, would show how a network topologically distintegrates when it is subject to attacks. It was found that as failures are included, average shortest path length and the Robustness Coefficient show consistent topological disintegration; showing the effect of the failure. Network diameter does not show topology changes when the failure is located outside of the cluster containing the diameter.

Keywords: Complex Systems, Complex Engineered Systems, Complex Networks, Product Architecture, Failure Analysis
5.2 Introduction

A basic problem in the design of complex engineered systems is to engineer and monitor the behavior of components so that failures do not produce adverse interactions with other components or sub-systems. It is the type of recommendation that underscores a key finding in complex network research: system failure is at least equally dependent on system topology as a (weak) component [4].

In order to understand the behavior of complex systems, and in particular the relation between structure and fault behavior, researchers have modeled them as networks [2, 67]. Applied studies on the failure of real-world systems include the US air transport network [93] and the North American power grid [54]. Complex engineered systems have also begun to be modeled as networks [80, 83, 63, 64, 35]. Prior to this, researchers had applied social networks analysis and complex networks theory to analyze the statistical properties of very large scale design and engineering projects, and have shown that their network models have structural (architectural) properties that are like those of other biological, social, and technological networks [15, 16]. Methods from complex network and graph theories provide intriguing tools to evaluate the robustness and resiliency characteristics of complex engineered systems [3]. Centrality and viral propagation metrics, which had previously been used to determine the vulnerability of software architecture to virus propagation [43, 42], have been applied to understand the failure tolerance of complex engineered systems due to the propagation of a fault, such as the loss of flow or energy [63, 64].

The underlying idea in the application of complex network methodologies to understand the failure behavior of complex engineered systems is to use network failure metrics to evaluate system architecture in the early stages of the design process to create increasingly robust architectures. A robust system architecture is one that is able to continue operating as intended given the presence of variations to internal or external operating conditions. This includes expected component failures and system-level effects.

In the literature, it has been shown that the use of complex networks for the evaluation of complex engineered systems is largely dependent on the granularity in which information is to be captured and used during model evaluation [35, 78, 64, 63]. In this paper, we utilize a bipartite ‘behavioral’ network to capture the pertinent information defining fallible
system behavior. Without this information, an evaluation of failures would not be possible because the model contains insufficient information to capture a failure [35]. The model lacks a formal definition of the failure and how it appears within the system.

For the analysis and discussion presented in this paper, the authors have utilized the following definitions for evaluating complex engineered system characteristics:

1. A *Function* is the transformation of a flow [87]. A *Function* can be mathematically represented with an equation.

2. *Behavior* is how a *Function* is achieved. For instance, system *Behavior* might be the rotation of a shaft.

3. *Performance* is the manifestation of a *Behavior*. For instance, a shaft rotating at 100 revolutions per minute (RPM).

4. A *Failure* is the degradation of *Performance*. For instance, a shaft rotating at 80 RPM is in a *Failure* state when it should be rotating at 100 RPM.

In an effort to expand the application of complex network theory to the evaluation of complex engineered systems, this paper presents three commonly used network metrics used to quantify topological robustness, average shortest path length, network diameter, and a Robustness Coefficient. This was to determine which metrics are most applicable for the evaluation of failures and which metrics over-magnify or under-represent the size of the failure or the effect on system performance.

5.3 Related Work

Before proceeding, we begin with a discussion of whether ‘complex’ engineered systems satisfy the requirements of ‘complex systems’ and the type of behavioral understanding that a complex network approach brings.
5.3.1 Complex Systems

Complex systems are a branch of systems with a specific set of attributes. The differentiation between complex and complicated systems often leads to confusion about the nature of being ‘complex’. In a complex system, internal system interactions and relationships are not separable so long as system behavior is retained [17]. A complicated system is one that may be large, have many components, or be difficult to build [26]. While these attributes are true also of complex systems, non-separable interacting relationships are not present in complicated systems. Complex systems are present in many domains, including inside and outside of the engineering disciplines [60].

While complex engineered systems share properties with other complex systems, such as emergent behaviors and interactions between components that cannot be fully understood and modeled [81, 59], some authors argue that even very complicated systems, such as aircraft carriers, are not actually complex [66]. We use the terminology, complex engineered systems, to distinguish them from other complex systems (such as biological ones) to emphasize that they are artificial, intended to perform specific functions, and comprised of interdependent engineered systems. Examples include aircraft, process plants, and satellites [12]. Further, we agree that complex engineered systems are indeed complex systems. Although the engineering of these systems is thought to be purely rational, properties of self-organization in the design process [66] means that the resulting system design shares an important property with complex networks. Systems will have architectures and behaviors that are neither truly regular nor random, and are dependent upon connection patterns.

This property makes complex engineered systems suitable for analysis using a complex network approach. Patterns of connections between components and their physical behaviors can not be reasonably thought of as ‘regular’ or fully predictable. Understanding their behavior requires an understanding of both the constituent components and the patterns of connections between those components. It is the pattern of connections, rather than the connections themselves, that have a significant effect on the behavior of the system [69, 35].
5.3.2 Complex Networks

Complex networks are networks wherein the topological features of the network are non-trivial. That is, connections between nodes are neither random nor are they regular [70]. There is no predictable pattern governing how a network may be topologically organized.

Networks are represented by a series of nodes with associated edge connections representing a relationship between nodes. Importantly, all nodes are considered to be zero-dimensional objects of similar ontological category. That is, in a network model of airports, both ATL and BUF would be modeled as a node even though ATL is clearly a much larger airport in terms of land area than BUF. If a network contains nodes of a single type, it is called a uni-partite network. If the nodes are of two types, it is called a bipartite network. The premise of network theory is that to understand how a system works, such as the air transport system in the US, we must understand the network structure of the system. We transfer this basic premise toward understanding the behavior of complex engineered systems.

Network representations of complex engineered systems are becoming increasingly popular amongst researchers looking for new ways to identify and design against fallible [51, 67] systems. Complex network theory examines the topology of a network in order to gain information related to structure. Computationally tractable metrics describing network topology such as node degree (the number of nodes to which a node is connected) can provide insight into overall network robustness [3]. This information has already been used to identify the weak aspects of a system and to redesign the system in an effort to increase its quality [83] and to understand the degree of coupling (hence ‘complexity’) of the system [62, 6]. However, in practice, the differentiation between a complex network and a complex engineered system presents challenges, because a complex engineered system is not a complex network per se. Complex engineered systems are not normally constructed as a network because there are many heterogeneous parts (compared to the air transport network wherein all nodes are conceptually the same). Nor are they typically represented in the same manner as complex networks. Some researchers have frowned on using complex networks because of these deficiencies [38].
Given this reasoning, it is important to recognize the capabilities of a network given a particular modeling desire. As with traditional engineered system modeling, it is important to capture the information required to perform the analysis. An evaluation of failures would not be possible unless the information, properties, and characteristics defining that failure were present. For this reason, it is important to make conscious, accurate decisions regarding network granularity. If a failure is to be modeled, the information must be present and organized with proper ontological identifications for node relationships [63, 64, 78, 79, 80, 35].

The failure modeling and analysis performed in this paper was conducted with a bipartite ‘behavioral’ network [35]. This was done in order to determine the relevant relationships between the behavioral response of the system and the parameters (design variables and noise factors) which define system functionality. Additionally, as functional information is required to properly implement failures [72], a bipartite ‘behavioral’ network provides the necessary information to accomplish this.

While bipartite network modeling is certainly a way to represent complex engineered systems as networks, it is important to note that other research has been conducted on the development of such modeling relations. As noted by Alfaris et. al. [5] and Hamraz et. al. [36], the development of multi-layer Domain Mapping Matrices (DMM’s) and Design Structure Matrices (DSM’s) provides an avenue to represent physical architecture, behavior, and an array of input parameters within the same network. As a result, off-diagonal matrix blocks would represent the relationships present between the various layers and could provide information about the system.

5.3.3 Failure Modeling

The study of complex engineered system failures is heavily focused on the reliability of individual components and on performance simulations conducted under variable input conditions. This was so that an understanding of the system-wide effects of input variability [53, 91] could be gathered.

To maintain operability, redundancy is built into most engineered systems so that the reliability problem becomes one of predicting the effects of component failures. Because
of this, engineers direct their efforts towards the resilience of complex engineered systems. Resiliency describes the ability of a system to recover from failures and continue behaving within expected performance variations [94]. This is different from traditional views of robustness which describe how a system responds to variations of input parameters or from reliability which describes the ability of a system to operate for a specific amount of time. For the purpose of this paper, we will focus on views of robustness for complex systems and complex networks.

To avoid failures of important components, setting aside the problem of identifying these important components, the engineering design literature recommends techniques such as a FMEA [34] or a FFIP [87] method. While these techniques have proven useful where information of failure modes is available or predictable, they can not readily handle the interaction effects of failures. They are also not effective for identification of the most vulnerable components without significant prior knowledge of the system or the specific failure. In both cases, significant expertise of engineers and a knowledge base of operational and historical systems and failures is required.

Network analysis is promising for addressing the problem of designing complex engineered system architecture. There is a growing body of literature investigating the resilience of various real-world and model networks to random and targeted node and edge removal. These sorts of actions are described as an attack. Node removal is a type of attack wherein the node itself, and by association, each edge connected to that node, is removed from the network. For instance, if in an air transport network, one airport were to be closed, the node and every edge describing all arriving and departing flights would be removed, thereby ‘closing’ that airport. The other commonly used type of attack is edge removal. Edge removal is the dissociation of two nodes, but not necessarily the complete removal of either node. For this paper, the failure implementation process utilized is one of edge removal. Essentially, failure modes are implemented into a network as degradations in edge weight, which in an extreme case would result in complete edge removal. These degradations would then have clear parallels to the changes in system performance that are caused by degradations in constituent parameters.

One of the first significant findings in this area found that scale-free networks are robust
to random node removal but are vulnerable to targeted attack [4]. The effects of targeted and random attacks on networks have been described using various topological metrics such as the average inverse geodesic length and the size of the largest connected subgraph [40]. Results show that attacks based on dynamic betweenness centralities (nodes with high betweenness play important roles in connecting other nodes within the network) of nodes have a more harmful effect than strategies based on the initial network [93, 40].

There are challenges in directly translating findings on the failure tolerance of complex networks to complex engineered systems. First, while the behavior of an engineered system is dependent upon inter-connectivity between components, it is not sufficient to describe this behavior in terms of topological properties alone, which is how network robustness is characterised. Generally, the performance of complex engineered systems must be simulated with techniques such as a response surface method, robust optimization, or model-based design simulations to understand its performance when many parameters change [53, 7, 75, 90]. Secondly, producing system architectures based on physical and functional dependencies may produce systems having network topologies that make them vulnerable to attack. Nonetheless, the field of complex networks contains some insights that may be useful in identifying which nodes and edges of complex engineered systems make them vulnerable to attack [16, 63, 64, 35, 78, 79, 80]. Insight from this research would be especially useful in the design of new complex engineered systems for which we have insufficient operational history to know the effect of failure.

5.4 Methodology

Presented is the methodology performed in this paper for the analysis of complex engineered system failures. First, a discussion of how to convert a complex engineered system to a bipartite ‘behavioral’ network is presented. Following is a discussion of the three network metrics utilized, average shortest path length, network diameter, and a Robustness Coefficient. When network topology disintegrates due to an attack, changes in these metrics can describe the severity of the attack through the amount of topological change.

The following sections outline the methodology performed in this paper for use on bipartite networks. While similar to evaluations of uni-partite networks, there are differences
found in literature that can be explored further by the interested reader [63, 64, 35, 50, 76].

5.4.1 Bipartite Network Development

Network representations involving relationships between two distinct node types are represented with a bipartite network. In this paper, the bipartite ‘behavioral’ network describes the relationship between the functions of a complex engineered system that define the performance of the system and the parameters (design variables and quantifiable noise factors) which define each function. Certainly not all functions or parameters could be considered completely commensurate. However, because each function defines system performance, each parameter defines each function, and system failures appear as changes in performance, the nodes are considered to be so.

**Figure 5.1: Bipartite network. Two node groups, three sub-nodes within type-1 and four sub-nodes within type-2.**

Bipartite networks show the relationship between two distinctly separate types of nodes. An example of a bipartite network is displayed in Figure 5.1. The example has two node types with three sub-nodes within type one (Group 1) and four sub-nodes within type two (Group 2). Each node type can be referred to as a mode. The individual sub-nodes are then connected with edges across each mode.

Bipartite networks are represented with rectangular adjacency matrices. The matrices can be weighted or un-weighted. An adjacency matrix for the example bipartite network
displayed in Figure 5.1 has been provided in Figure 5.2.

\[
\begin{pmatrix}
1 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 \\
\end{pmatrix}
\]

Figure 5.2: Bipartite adjacency matrix \((A)\) for network in Figure 5.1.

The following rules are applicable when using a bipartite network:

1. No sub-node can be connected to any other sub-node from the same mode.
2. A sub-node from one mode must be connected to at least one sub-node from the other mode.

In determining the bipartite adjacency matrix, a rectangular matrix is used with size \(N \times M\).

\[
N = \text{Number of Sub Nodes in Type-1} \quad (5.1)
\]

\[
M = \text{Number of Sub Nodes in Type-2} \quad (5.2)
\]

The adjacency matrix, \(A\), in Figure 5.1 is defined as follows:

\[
A_{ij} = \begin{cases} 
1 & \text{edge connection between sub node } i \text{ and } j \\
0 & \text{otherwise}
\end{cases} \quad (5.3)
\]

Where \(i\) goes from 1 to \(N\) and \(j\) goes from 1 to \(M\). For the weighted case, \(A\) is defined as follows:

\[
A_{ij} = w_{ij} \quad \text{edge connection between sub node } i \text{ and } j \\
0 \quad \text{otherwise} \quad (5.4)
\]

Often it is necessary to represent a bipartite rectangular matrix as a block, \((N + M) \times (M + N)\), adjacency matrix. This way, all of the information required for the analysis is
preserved, yet the math required to be performed is simplified for use on a ‘uni-partite’-like network representation [32, 78, 79, 80]. The following adjacency matrix, $B$, is represented as:

$$B = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix}$$  \hspace{1cm} (5.5)$$

Where $A$ is the adjacency matrix from Eq. 5.3 and $A^T$ is the transpose of Eq 5.3. As can be seen, $B$ retains the correlation between both modes with a 1 representing a correlation between nodes and a 0 showing a lack thereof.

### 5.4.2 Average Shortest Path Length

The average shortest path length, sometimes called *average path length*, is a measure of the topological connectedness of a network [13]. The average shortest path length describes the relative efficiency for a flow to travel throughout a network. In this context, the average shortest path length is less a measure of robustness and more a way to quantify how a network disintegrates under sustained attack. When a network is subjected to edge attacks, an edge between two nodes is eliminated. This removal is captured by the metric because the shortest paths between nodes would be altered as a result of the attack, thereby disintegrating the topology of a network.

Consider the block network representation presented in the previous section (Eq. 5.5). Assuming $B$ is a graph with a set of nodes, $K$, the shortest distance $d$ between any two nodes is $d(k_i, k_j)$ where $k_i, k_j \in K$. The average shortest path length, $L_{ASP}$, is defined as:

$$L_{ASP} = \frac{1}{(N + M)(N + M - 1)} \sum_{i \neq j} d(k_i, k_j)$$  \hspace{1cm} (5.6)$$

Where $N$ is the number of nodes in mode 1 and $M$ is the number of nodes in mode 2. Eq. 5.6 has been normalized with the size of the network in order to provide scale and comparability between networks of different sizes [13].
5.4.3 Network Diameter

Network diameter is similar to average shortest path length in its calculation [13]. It is defined as the longest shortest path between any two nodes within the network, the geodesic length [35, 78, 79, 80, 50, 76]. While useful to measure the effect of sustained attack on a network, it does not carry much significance on singular instances of attack, as geodesic distances between nodes must change network wide for effects to be observable. Nonetheless, when under sustained attack, changes to network diameter can represent fundamental shifts or breakdowns in network topology, a fact that may make it easier to identify weak topology and then redesign.

In complex network representations of complex engineered systems, average shortest path length and network diameter provide insight into how interconnected both modes are. By evaluating the network models with these metrics, we are able to obtain a better understanding of how these systems are constructed and what could be done in order to make them more robust.

In this paper, both average shortest path length and network diameter are evaluated with the network representation provided in Eq. 5.5. The next metric, the Robustness Coefficient, is evaluated with an adjacency matrix as presented with Eq. 5.3.

5.4.4 Robustness Coefficient

The Robustness Coefficient has been a widely studied area of research in network theory. The coefficient is a measure of the topological robustness of a network under sustained attack [76, 50]. When a network is fully connected, the coefficient is 100%. As the network disintegrates, it reduces to 0%.

The Robustness Coefficient presented by Piraveenan et. al. [76, 50] is based on the size, or degree, of the largest node after an attack. When a node is removed, this may alter the size of the largest remaining node. Therefore, when evaluated over the total number of nodes, the metric could serve as a network approximation of system stability when subject to attacks, a useful feature as traditional system mechanisms for evaluation are computationally expensive and difficult to perform.
As can be seen in an examination of existing literature, the established coefficient is valid for use on uni-partite representations of networks. As this paper analyzes a complex engineered system with a bipartite network, an alteration must be performed for the metric to remain valid. The specific derivation of the metric is identical to the derivation for unipartite networks; however, the premise behind the metric must change slightly.

From [76], it can be seen that the Robustness Coefficient is a ratio of the areas beneath two network disintegration profiles.

\[
R = \frac{A_1}{A_2}
\]  

(5.7)

Where \( A_1 \) is the area beneath the profile of an expected network and \( A_2 \) is the area beneath the profile of an idealized network. Rather than evaluate the network for the size of the largest connected node, as has been done previously, the presented metric is evaluated with the average size of a node belonging to a specific mode. Average node size is commonly utilized in network analysis for evaluating the general size of a network. This allows the idealized network node size to decrease by unity with the removal of each node belonging to the other mode. Because of this, the derivation continues similarly to the literature. Eq. 5.8 through Eq. 5.10 describe the calculation of the area beneath each profile.

\[
A_1 = \frac{1}{2}(S_o + S_1) + \frac{1}{2}(S_1 + S_2) + \ldots + \frac{1}{2}(S_{N-1} + S_N)
\]  

(5.8)

\[
A_1 = \sum_{k=0}^{N} S_k - \frac{1}{2}S_o
\]  

(5.9)

\[
A_2 = \frac{1}{2}N^2
\]  

(5.10)

Because a bipartite network is represented by an \( N \times M \) adjacency matrix, the robustness coefficient is calculated with respect to the average size of one mode while the other is subjected to sustained failures. Either mode could be chosen for application. For the purpose of this paper, the performed analysis follows the equations as they are presented here.
For a complete derivation of the metric, please consult the literature [76, 50]. The Robustness Coefficient, normalized to a 0-100 scale, is defined as the following:

\[
R = \frac{200 \sum_{k=0}^{N} S_k - 100S_o}{N^2}
\]  

(5.11)

Where \( S_k \) is the average size of the nodes in one mode after \( k \) nodes from the other have been removed. \( S_o \) is the initial average size. \( N \) is the number of nodes in an \( N \times M \) network. It can be verified that the equation has been normalized to result in 100% if the network is fully connected and 0% if it is completely disconnected.

The Robustness Coefficient is valid so long as the network is an unweighted and undirected graph. However, the coefficient remains valid so long as edge weights are between 0 and 1.

In the case study, we will show the performance characteristics of a drive-train system model operating nominally and under the effect of a failure. This model will then be converted to a bipartite network for an evaluation of topological disintegration under targeted attack. This attack will manifest as the same failure presented with the drive-train model simulation to determine the effect of a failure on both network topology and the performance of an engineered system.

5.5 Case Study

Presented is a drive-train modeled with OpenModelica, a Modelica® software platform. The model is used to show how failures related to the parameters which define system behavior and performance can impact network topology for system evaluation. Following the analysis will be a discussion of the relevant results and an evaluation of whether or not failures to complex engineered systems can be adequately modeled with a complex network at this level of granularity.

For both the system and network evaluations, a torque degradation failure was implemented in the clutch. The failure is associated with a degradation of the coefficient of friction, \( \mu \), between rotating clutch disks. This results in reduced torque output to the rest of the drive-train.
5.5.1 The Drivetrain Model Simulation

![Drivetrain model constructed in OpenModelica](image)

The model depicted in Figure 5.3 depicts a drive-train that accepts a constant torque and clutch state input. This torque is passed through a clutch, bearing, and two gearing manipulations under an inertial load (perhaps a shaft). The equations governing model performance have been presented in Eq. 5.12-5.18. Table 5.1 provides the definitions for the parameters used in each equation.

\[ F_1 : Clutch_{fric} = \mu \cdot cgeo \cdot F_n \]  \hspace{1cm} (5.12)

\[ F_2 : cgeo = N \cdot 1/2 \cdot (r_o + r_i) \]  \hspace{1cm} (5.13)

\[ F_3 : Clutch_{out} = Torque_{in} - Clutch_{fric} \]  \hspace{1cm} (5.14)

\[ F_4 : 0 = Clutch_{out} + bearing_{out} - \tau - fric_{viscous} \]  \hspace{1cm} (5.15)

\[ F_5 : fric_{viscous} = 10^{-7} \cdot f_o \cdot (nu \cdot RPM)^{2/3} \cdot dm^3 \]  \hspace{1cm} (5.16)
The equations represent fallible behavior. That is, changes in the values of the parameters may lead to incorrect performance of the drive-train. This is true for each equation aside from Eq. 5.13. In this equation, a parameter defining the geometric properties of a clutch, $c_{geo}$, has been included. While not strictly information related to behavior, this equation defines clutch properties which make a clutch subject to internal contact mechanism failures, such as clutch slipping. This is an important failure related to clutch performance not directly captured with typical network modeling techniques [72].

Significant research has been performed on how to model fault behaviors within complex engineered systems. Much of the work centers around the implementation of fault variables that degrade system performance [49, 72]. Because the implementation of such an approach requires the utilization of a fault variable with regards to an applicable system parameter, a network modeling technique must be chosen so that comparable and faultable nodes are commensurate [63, 64, 35]. For this analysis, a bipartite network was chosen so that comparisons between the functions describing system performance and the system parameters defining those functions would be commensurate. This information was acquired from Eq. 5.12-5.18. The network was then weighted with a fault variable defined between 0 and 1; 1 if no failure exists within the parameter, between 0 and 1 if a degradation failure is present, and 0 otherwise. This scale ensured the applicability of each network metric.

Eq. 5.19 displays the implementation of a fault variable in $F_{1f}$ for use in a traditional system model. The implementation process would be identical for each additional function. In a network, each fault variable, denoted $FV_i$ through $FV_M$ (M is the number of mode 2 nodes), represents the failed state of the parameter for which it is associated, and is applied to an edge (Fig. 5.5). This allows for the dynamic alteration of network topology according to the current state of each relationship.

$$F_{6f} : Clutch_{fric}(FV_1) = \mu(FV_2) * c_{geo}(FV_3) * F_n(FV_4)$$

\[ (5.19) \]
Table 5.1: Variable Value Definitions [90, 37]

<table>
<thead>
<tr>
<th>Variable</th>
<th>Related Component</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Clutch_{fric}$</td>
<td>Clutch</td>
<td>Amount of Torque from Friction</td>
</tr>
<tr>
<td>$\mu$</td>
<td>Clutch</td>
<td>Static Coefficient of Friction</td>
</tr>
<tr>
<td>$c_{geo}$</td>
<td>Clutch</td>
<td>Geometric Constant for Clutch Surface</td>
</tr>
<tr>
<td>$F_n$</td>
<td>Clutch</td>
<td>Normal Force Between Clutch Plates</td>
</tr>
<tr>
<td>$N$</td>
<td>Clutch</td>
<td>Number of Frictional Surfaces</td>
</tr>
<tr>
<td>$r_o$</td>
<td>Clutch</td>
<td>Outer Surface Diameter</td>
</tr>
<tr>
<td>$r_i$</td>
<td>Clutch</td>
<td>Inner Surface Diameter</td>
</tr>
<tr>
<td>$Clutch_{out}$</td>
<td>Clutch</td>
<td>Torque Output from Clutch</td>
</tr>
<tr>
<td>$Torque_{in}$</td>
<td>Clutch</td>
<td>Torque Input to Clutch</td>
</tr>
<tr>
<td>$bearing_{out}$</td>
<td>Bearing</td>
<td>Torque Output from Bearing</td>
</tr>
<tr>
<td>$\tau$</td>
<td>Bearing</td>
<td>Bearing Torque Loss</td>
</tr>
<tr>
<td>$fric_{viscous}$</td>
<td>Bearing</td>
<td>Bearing Torque from Viscous Friction</td>
</tr>
<tr>
<td>$f_o$</td>
<td>Bearing</td>
<td>Bearing Type Factor</td>
</tr>
<tr>
<td>$\nu$</td>
<td>Bearing</td>
<td>Kinematic Viscosity of Lubricant</td>
</tr>
<tr>
<td>RPM</td>
<td>Bearing</td>
<td>Shaft Speed</td>
</tr>
<tr>
<td>$dm$</td>
<td>Bearing</td>
<td>Bearing Diameter</td>
</tr>
<tr>
<td>ratio1</td>
<td>Gear 1</td>
<td>Gear Ratio for Gear 1</td>
</tr>
<tr>
<td>$gear1_{out}$</td>
<td>Gear 1</td>
<td>Torque Output from Gear 1</td>
</tr>
<tr>
<td>ratio2</td>
<td>Gear 2</td>
<td>Gear Ratio for Gear 2</td>
</tr>
<tr>
<td>$gear2_{out}$</td>
<td>Gear 2</td>
<td>Torque Output from Gear 2</td>
</tr>
</tbody>
</table>

Figure 5.4: Simulation Results of a Drive-train. Nominal (Blue). Slipping Clutch (Red)
To simulate a failure in the torque transferability of a clutch, a fault variable, $FV_2$, was applied. The model works by setting a clutch state input, disengaged, partially engaged, or fully engaged. For this simulation, the clutch was set to partially engaged at 75% of an arbitrary maximum. The model was transferred from OpenModelica to Matlab to take advantage of superior plotting controls.

Figure 5.4 shows the simulation results for a slipping clutch. The failure is implemented within the torque transfer capability of the clutch. Therefore, in some cases, a failure may be minor enough for complete transfer of engine torque through the drive-train despite the existence of a failure. This is because a clutch is only able to transfer what it’s plates can transfer. A slipping clutch would result if the input torque from the engine is greater than the frictional torque resisting relative motion between clutch disks [72]. In this case, the clutch is assumed to output torque less than the supplied engine torque but equal to the frictional resistance torque of the rotating clutch disks. For this analysis, in the event of a slipping clutch condition, there would be no relative motion between rotating disks. Only a reduced torque output according to the severity of the failure. For comparison, a fault variable of 0.5 was implemented into Eq. 5.12 for $FV_2$. The fault will be implemented into a bipartite network at the same magnitude, $FV_2 = 0.5$.

5.5.2 Modeling the Drive-train with a Bipartite Network

In the case of this drive-train, the relationships defining system performance have been reproduced in Equations 5.12-5.18 and were available through model instantiation in OpenModelica. Each equation is a function belonging in mode 1, $F1$ through $F7$. Each function’s constituent parameters belong in mode 2. Therefore, to construct the bipartite network for this drive-train, there are 7 sub-nodes within mode 1 (functions, $N$) and 20 sub-nodes within mode 2 (parameters, $M$). A connection between modes will occur if a mode 2 system parameter is present within a mode 1 function.

Figure 5.5 displays the bipartite network for the drive-train.

When analyzing a failure in a network model, changes in topology manifest in the adjacency matrix representations of the model. Dissociations between nodes or changes in edge weight describe an attack, or failure. This type of attack implementation is edge
removal. For each network metric described earlier, two simulations were conducted, one for a nominal drive-train and one for a slipping clutch. It is important to remember that these network metrics describe topological properties and not system performance. They are designed to identify weakly connected network clusters. These weaknesses then outline potential areas for redesign in order to lessen the effect of a failure on system performance.
5.5.3 Network Evaluation of Failure

To evaluate the network model of a complex engineered system for failure, an adjacency matrix must be produced from the graph in Figure 5.5. For the nominal case this has been produced in Figure 5.6. Each row represents the relations of a parameter to a specific function. Each column represents the relations of a function to a specific parameter.

\[
\begin{pmatrix}
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
\end{pmatrix}
\]

Figure 5.6: Adjacency Matrix for Nominally Operating Drive-train

An adjacency matrix with a failed clutch having the same fault variable as the simulation has been produced in Figure 5.7. The second column from the left is representative of the failure by the change in edge weight. Each existing edge associated with \( \mu \) (the failed node) has been altered to \( FV_2 = 0.5 \) in order to reflect the failure.

\[
\begin{pmatrix}
1 & 0.5 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{pmatrix}
\]

Figure 5.7: Adjacency Matrix for Failed Drive-train Clutch

Figure 5.8 shows the results from the network topology analysis. Three subplots represent average shortest path length (left), network diameter (center), and the Robustness Coefficient (right) results.

The results show a direct topology change from average shortest path length and the Robustness Coefficient, yet failed to show a correlation from network diameter. In order to explore this relationship further, another analysis was performed in order to see how
failures to any node within the network would impact the topological disintegration of the bipartite network model. This analysis has been included in Figure 5.9. To compare, the value for each fault variable was set to $F_{Vi} = 0.5$ and was failed independently of all other parameter nodes (i.e., only one node is in a failure state at a time). Therefore, the value of each network metric would be identical for parameter node 2 ($mu$) in this analysis as they were in the analysis from Figure 5.8.
As expected, the network diameter is dependent on the topology of the network more so than the average shortest path length and the robustness coefficient. This is because the network diameter is a metric describing static topology. That is, changes in surrounding topology (failures) may not affect the longest shortest path in the network. The overall diameter may not change while the size of local clusters do. This differs from the other metrics because they are more dynamic. Changes in local cluster topology effect the average cluster size network-wide. No changes from failures are locally bottlenecked.

5.6 Discussion and Conclusion

Typically, uni-partite network representations of complex engineered systems have been utilized in the literature. There are some immediate advantages to this. When structure is important, simply knowing the topological properties of a system is all that is necessary. This is especially true of a group of systems known as cyber-physical or mechatronic software systems. These systems are collections of hardware and software components (or
modules) that interact through connected relationships. Many researchers have looked into the topology that defines the emergent interactions present in such interdependent systems and have found a strong correlation between failures to interconnected modules and the topology which defines the interdependencies [50].

In other cases, though, simply acknowledging the topology present in the complex engineered system is not sufficient to adequately model a failure. This is where models constructed with bipartite networks are useful. The examination of relationships between multiple dependent entities which affect system performance characteristics is important, especially early in the design process when costs for design changes are drastically less. A networks perspective in complex engineered system design provides a method in which to evaluate these systems prior to physical development and identify how a system could break down. This way, redesign would take place prior to the physical manifestation of a device (or a failure) when risk and cost are higher.

Presented in this paper were three metrics used for the topological evaluation of network robustness. These metrics, average shortest path length, network diameter, and a Robustness Coefficient, were evaluated on a bipartite network model of a drive-train to determine how the network disintegrates under the influence of a failure. This evaluation was compared to a simulation of torque response for the system. It is not immediately clear if a direct correlation exists between the change in simulation torque and the change of any one of the network metrics. However, it was shown that two of the metrics, average shortest path length and the Robustness Coefficient, showed topological disintegration patterns which differed between nominal and failed cases, regardless of failure implementation location. The network diameter is not sufficiently dependent on local cluster topology to consistently show changes in topology when under the influence of a failure (Fig. 5.9). However, between average shortest path length and the Robustness Coefficient, while a failure certainly is shown in the evaluation of each metric, they are currently most useful in identifying where a failure originates from, rather than the actual negative impact provided as a result of failure. This is still beneficial because local and global topology can be optimized in such a way so that failures related to any one parameter do not cripple a system.
5.7 Future Work

While the drive-train model used in this paper is useful for explaining the approach and to justify the methodology, in future work, a large scale system will be analyzed to determine if the approach is scalable. This will be performed against a wide array of different failure modes. Additionally, research is required to map simulation results of complex engineered system models and network topology metrics to determine the relevant correlations and manifestations of system performance within network analysis tools.
Chapter 6: Conclusion

As complex engineered systems continue to grow in size and complexity, finding new ways to model their structures and evaluate their failure properties is of vital importance. Complex networks provide one avenue for such exploration. The issue with using complex networks, as many researchers have pointed out, is that comparing such dissimilar entities as system components with networks is not appropriate for anything other than spatial layout and organizational analysis. This is especially true when conducting an analysis of system behavior and performance when a component, or group of components, consists of significantly different characteristics compared to that of other components within the same system.

However, the usefulness of complex networks is undeniable. Topological patterns of connectedness among different modeling motifs can suggest alternative ways to design a system without looking at a specific component solution. Essentially, identify what causes specific behavior or performance from a system and then design that system with resistance to known problems. Additionally, the emergent properties associated with interacting system elements may be observable within a greater design space so that these unknowns may not continuously present problems during the design process.

Within the first manuscript, methodologies for creating complex network models of complex engineered systems were introduced. According to a guideline of identifying commensurate node relationships, a behavioral network was developed for use with evaluating failure properties of complex systems. With the network, the relationships between function and design parameter which define system behavior and performance were mapped within a bipartite network. This way, system faults that manifest within the design parameters, are observable through a commensurate analysis of system behavior at the functional level. A degree centrality and eigenvector centrality analysis was then performed with the hope of identifying where a fault could manifest within the network model and evaluating the extent of topology change caused by interacting connection patterns within the net-
work. The results were promising, changes in local degree centrality identified which parameter was causing the fault and which functions contained that parameter. An evaluation of eigenvector centrality then displayed how changes to the connections within the functional relationships affect the global topology. The connections between functions caused by shared parameters became observable through shifts in topology.

Within the second manuscript, research related to the accurate portrayal of relationships between bipartite groups and methodologies for the global evaluation of failure topology was presented. For edge weights, failure flows were presented. Each design parameter is assigned a value describing its current health state [49, 72, 71], between 0 and 1, 0 representing completely failed and 1 representing completely healthy. Every edge connected to that parameter was then weighted with that value. This provides a method for which to represent failed parameters while still maintaining the underlying topology which is defined by connections to failed, or failing, nodes. After the network model developed in the first manuscript was scaled with failure flows, network topology was evaluated under attack (system failures). Average shortest path length and the robustness coefficient provided global evaluation of network topology when considering local cluster patterns, while network diameter did not. While the results did not conclusively show a direct correlation between topology disintegration metrics and system performance, they did provide positive indications of patterns which correlate disintegration profiles to system simulations. Additional research is required.

Complex networks offer intriguing tools in which to evaluate complex engineered systems. Emergent interaction detection, computationally inexpensive analysis, and developed evaluation tools, to name a few. This research provides a first look into this potential, while trying to formalize modeling practices for use with complex networks. As additional research is presented, this will help to aide in the design and development of increasingly complex engineered systems.

The current state of the method allows for an evaluation of network topology under ‘representative’ failure. That is, the performance of the system represented with network topology can not yet be conclusively proven to accurately model true system behavior. This makes the analysis of reliability and failure effects rather difficult. In order to improve the
current state of the method, a series of future work applications have been provided in the following chapter in order to outline the potential avenues for further research exploration given the current shortcomings.
Chapter 7: Research Shortcomings and Future Applications

The network ideas and methodologies presented here for the evaluation of complex engineered systems are with respect to a simple drivetrain model constructed in OpenModelica [90]. As both manuscripts are to be submitted as journal versions, a larger scale example is to be included in order to show the methodology on a more complex engineered system. Currently, a variable geometry turbo-charged diesel engine will be used.

As with many burgeoning research areas, there are several unexplored avenues in which to continue complex engineered system reliability research, both inside and outside of the context of a network framework. The research methodology presented in this document has some very fundamental benefits compared to the state of existing reliability research, including the potential for emergent interaction detection and analysis. However, there are several shortcomings related to usability and scalability concerns, failure definition, additional modeling approaches, and implementable evaluation frameworks which lead to a series of potential future work opportunities.

7.1 Usability and Scalability

Typically in complex engineered system research, concerns about usability and scalability of a method arise. These include concerns from mathematical constraints and processes perhaps too difficult for a wide array of people to utilize or from the amount of data and detail required to implement a specific method.

As has been mentioned numerous times, complex engineered systems exist in a wide array of sizes, shapes, and functions. When addressing issues with system reliability, one of the major shortcomings of newly developed methods is in relation to the class of engineered system to which the method could be used. That is, many research methods, including the network framework presented here, are so highly dependent on data and very detailed representations of large systems that the method cannot be practically used. Additionally,
because extensive amounts of variation exist between even very similar systems, no two models may be practically used in tandem and applicably compared.

With regards to the network representation and evaluation framework presented in this thesis, even the small drivetrain example which included minimal functional behavior resulted in a network with groups of seven and twenty sub-nodes. As systems become increasingly larger, it is reasonable to expect the size of network representations to expand exponentially as additional components and behaviors are added. It is possible that even the smallest real-world complex engineered systems may require thousands or potentially millions of nodes, putting a larger emphasis on computing power as well as algorithmic processes, resources, and organization.

Future Work

Network templates are designed to emulate the topological patterns of related items without having a full grasp on the nuanced details of system operation. A network template represents a section, or cluster, which provides information about the system without having modeled the system entirely. It has been suggested that types of network templates can be created which represent a complex engineered system. Therefore, a series of generic network representations can be combined in order to glean information about a complex engineered system prior to being constrained by physical structure or defined functionality. This has immediate benefits because the requirement of highly detailed system representations has been removed. There are potential limitations to this, however. As no two complex engineered systems are identical, this approach would require a large number of network templates to be simulated in order to gain information on trends for the system. However, if setup in a formalized framework, these templates can provide useful detail prior to ever implementing a system physically. Network templates will be developed and analyzed in order to develop design rules and guidelines. With this information, design engineers will be better able to produce more highly robust, resilient, and reliable complex engineered systems.
7.2 Failure Definition

As was shown in this research, failures to complex engineered systems which were implemented into corresponding network representations showed correlated changes to network topology. However, the behavioral manifestation of the failure, a failure mode in system terminology, is lost within the network topology. Additionally, because different failures can result in equal shifts in network topology, there is no current way to associate a specific change in network topology to a clearly defined failure within a complex engineered system.

Future Work

To combat this problem, research will be conducted on benchmarked network topology metrics. This way a specific change in network topology can be associated with the specific system failure that it emulates. Additional research is also required to determine the definite relationships between a fault in a complex engineered system (observed through altered performance characteristics) and a corresponding fault in a network (observed through changes in network topology) that is unique, quantifiable, and defined on a specific and measurable scale. This can be done with a variety of pattern matching and clustering algorithms designed to see how a network is topologically organized.

7.3 Additional Modeling

The included network representation method analyzes the functional and behavioral properties of a complex engineered system in order to model system failures without the loss of emergent interaction information. As with typical engineered system modeling techniques, many types of models are sometimes required in order to acquire sufficient detail for a true analysis of system reliability. This may include structural information, spatial information, use-case information, and control directives, to name a few. The current state of the included framework is not able to handle this issue.
Future Work

A layered network is a type of network wherein the topological properties of different, but related, networks are compared. For instance, a network representation of a traffic system may have a physical architecture layer which represents the connections between cities with a series of roads. Other layers within the network may then look at the specific lengths of the roads connecting a set of cities, the designed flow of traffic over those roads, and actual routes preferred by the drivers utilizing the infrastructure. This type of layered network allows designers to glean information related to multiple interacting elements that may not be ontologically equivalent.

With regards to a complex engineered system, a layered network could allow for the analysis of multiple interacting entities simultaneously. Included could be the relationships between function and parameter, system performance, hazard models, physical architecture, etc. This type of multi-layer network would allow for a vast amount of knowledge to be obtained with regards to a complex engineered system.

It should be pointed out that a layered network would suffer from the same debilitating demand on data and detail that the included network representation method does. This makes the development of network templates of even greater importance because these templates could provide a solid foundation for system design prior to the physical development of a device when the costs for design changes and consequences from poor design decisions are lower.

7.4 Implementable Evaluation Frameworks

The topology metrics identified in this research are a good start to the problem of identifying vulnerable locations of a complex engineered system. However, just as complex engineered systems require multiple analysis metrics to be performed simultaneously (power, torque, speed, voltage, etc.), a complex network representation may require multiple topology metrics to identify what is happening within a system. The current state of the research analyzes each metric individually in order to develop a universal evaluation of the system. This, like with complex engineered system models, may not tell the whole story.
Future Work

In order to evaluate a network thoroughly, a framework which looks at each topology metric simultaneously will be created. This framework will look at the optimization of a network representation according to topology metric constraints. Because each topology metric could be analyzed as being good or bad at opposite ranges of their defined scales, this framework will provide a topology optimization scheme which designs a network representation of a complex engineered system for use during early design.
VITA

Brandon Haley earned his Bachelor of Science degree in Mechanical Engineering from Oregon State University. While finishing his Master of Science in Mechanical Engineering, he is eager to explore future endeavors in design and reliability engineering in addition to failure identification, analysis, and risk mitigation.
Bibliography

[1] United/continental merger creates latest global mega-network carrier serving 370 airports in 59 countries; will be 1 for us-europe flights and us-asia flights. anna aero, May 2010.


