ON THE RESILIENCE OF COMMAND AND CONTROL ARCHITECTURES

by

Mark Andrew Pflanz

A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Systems Engineering and Operations Research

Committee:

Alexander Levis	Dr. Alexander H. Levis, Dissertation Director
[illegible]	Dr. Andrew G. Loerch, Committee Member
[illegible]	Dr. Lance Sherry, Committee Member
[illegible]	Dr. Lee E. Wagenhals, Committee Member
[illegible]	Dr. Stephen Nash, Program Director
[illegible]	Dr. Lloyd J. Griffiths, Dean, Volgenau School of Engineering
Date: 12/07/2011	Fall Semester 2011 George Mason University Fairfax, VA

On the Resilience of Command and Control Architectures

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Mark Andrew Pflanz
Master of Science
Virginia Polytechnic Institute and State University, 2005

Director: Alexander H. Levis, University Professor
Electrical, Computer and Systems Engineering

Fall Semester 2011
George Mason University
Fairfax, VA

Copyright 2011 Mark Andrew Pflanz
All Rights Reserved

DEDICATION

This is dedicated to my best friend and wife Shareen, and our two children: Allison and Katherine. During the course of my years in graduate school, Shareen has been unselfishly loving and supportive. Without her patience and support, none of my graduate school progress would have been possible.

ACKNOWLEDGEMENTS

I would like to thank Dr. Alex Levis for the opportunity to learn under his mentorship. In 2004, while completing my Master of Science degree, I had the distinct pleasure of attending one of Dr. Levis' three day courses on architecture development and analysis. I arrived at the course being very interested in the subject. I left the course knowing exactly what I wanted to study during my planned doctoral work. I was lucky that Dr. Levis agreed to be my advisor and accepted me as a research student. Since then, I have been constantly amazed at his generosity in time to support my research, as well as his wealth of knowledge and good ideas to expand my intellectual horizons. I am truly grateful. I would also like to thank the other members of my advisory committee: Drs. Loerch, Sherry, and Wagenhals. Each of them provided insightful comments and helpful recommendations during the conduct of this research. Finally, I wish to thank my supervisors and leadership at work, especially Mr. Chris Yunker, whose flexibility, patience and support allowed me to work part time over the past few years.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

ON THE RESILIENCE OF COMMAND AND CONTROL ARCHITECTURES

Mark Andrew Pflanz, PhD

George Mason University, 2011

Dissertation Director: Dr. Alexander H. Levis

From a national defense, as well as a civilian infrastructure perspective, a need exists to design and develop systems resilient to disruption. Systems able to survive and recover from disruption are referred to as resilient. In the case of mission critical systems, such as command and control systems, resilience is a necessary characteristic and must be considered early in the development cycle. This thesis describes a quantitative approach to evaluating the expected resilience of a command and control system. The approach uses a Petri Net based executable model of the system architecture. The rigorous graph-theoretic and executable properties of Petri Nets are leveraged to support structural and behavioral measures of the attributes of resilience. These measures are then combined into a holistic evaluation of resilience for the command and control system under study. The evaluation results can then be used to support selection among alternative candidate

architectures and to identify areas for improvement in the selected architecture. The approach is demonstrated in two different case studies.

CHAPTER 1

INTRODUCTION

1.1. BACKGROUND

From a national defense, as well as a civilian infrastructure perspective, a need exists to design and develop systems resilient to disruption. Disruptions are typically difficult to predict and can potentially be unavoidable. Systems able to avoid, survive and recover from disruption are referred to as resilient systems [Jackson, 2010]. In the case of mission critical systems, such as command and control systems, resilience is a necessary characteristic and must be considered early in the development cycle.

This research describes a quantitative approach to evaluating the expected resilience of a command and control system based on its architecture. The approach can be used to support selection among alternative architectures and to identify areas for improvement in the proposed architecture. The key idea is that resilience can be measured through its attributes, and that these measures may be combined into a holistic evaluation of resilience for the command and control system under study.

1.2. MOTIVATION

Today's society is highly dependent on complex command and control systems. In the civilian sector, examples include systems which control the financial system, the national air traffic control system, and the power distribution and control systems to name a few. In the defense sector, command and control systems represent the heart of our warfighting capabilities, with examples like command and control of strategic (nuclear) forces, command and control of carrier battle groups, and command and control of land formations at the tactical, operational and strategic levels. The normal functioning of these systems can be disrupted due to natural or man-made (unintended or malicious) actions. We view many of these systems as 'mission-critical,' because when they become unavailable due to a disruption, the impact of that loss is felt broadly and accomplishment of the overall mission may no longer be possible.

1.3. PROBLEM STATEMENT

From a national defense, as well as a civilian infrastructure perspective, a need exists to design and develop resilient systems. However, few quantitative means exist to evaluate the resilience of alternative architectures to assist in selection of a preferred architecture or to direct an architect's attention to areas in a design where improvements are needed.

Given a problem, an emergent threat, or a new opportunity, new capabilities are envisioned and development activities initiated. These early activities are crucial to the eventual success of the delivered capability. These early activities include the development of alternative architectures and competitive down-selection to a preferred architecture. They also include improvements made to the architecture based on structural, behavioral, and performance analyses of the architecture. For mission critical systems, these early development activities should include an evaluation of resilience.

1.4. HYPOTHESIS

Resilience can be measured through its attributes. For command and control systems, the attributes can be measured in the architecture before the system is built. In measuring the attributes, one can improve the architecture to include desirable features, or eliminate undesirable features. This information can also be used to assist in alternative architecture down-selection during early system development activities.

1.5. ORIGINAL CONTRIBUTION

No formal quantitative means exist to evaluate the resilience of alternative architectures to assist in design and selection. This approach develops measures and an integrative framework from which to quantitatively evaluate resilience. The approach may be used in alternative architecture down-selection. Additionally, it demonstrates means by which an architect may use this information to make improvements to the design.

1.6. DOCUMENT ORGANIZATION

This dissertation is organized into six chapters. Chapter One introduces the problem providing the background and motivation of the research, as well as defining the problem statement, hypothesis, and original contributions of the work. Chapter Two provides the context of the research, describing related work on resilience, defining key terms, and introducing certain original concepts of this research which underpin subsequent sections. Chapter Two also describes the role of architecture in system development, processes used to develop architectures, and progress in the evaluation of architectures. Chapter Three describes an approach to evaluating the resilience of a system using the architecture. It defines metrics, how to select metrics, and the combination of metrics into a holistic approach to evaluating resilience. Chapter Four describes the overall research approach and methodology. Chapter Five introduces two case studies to demonstrate the approach to evaluating resilience. The first case study is a time sensitive targeting example, with an architecture described using Business Process Modeling Notation (BPMN). The second case study is a military command and control organization with an architecture described using a five stage decision making model. Two different types of systems (a targeting system and an organization) modeled using two different architectural modeling techniques are deliberately selected to show that the approach described in this research is domain independent and robust against different architectural approaches. Chapter Five also describes how the resilience evaluation approach is applied, and the results of that application both in terms of evaluation of the design and demonstration of how the approach can highlight areas for design improvement. Chapter Six concludes this research by summarizing the results, providing conclusions, and making recommendations for future work based on this research.

CHAPTER 2

RELATED WORK

2.1. DEFINING RESILIENCE

2.1.1. Resilience Defined in Various Domains

Resilience as a concept is defined in different ways in various scientific disciplines. The word ‘resilience’ has its origins in Latin, where ‘resilire’ and ‘resilio’ meant: “the ability to rebound or jump-back.” In general, one’s definition of resilience can depend upon the context of the domain being considered. Regardless of the domain, various definitions of resilience tend to include common themes: disruption, avoidance, survival and recovery. Therefore, we will use the following definition of resilience from Jackson [2010]: *the ability to avoid, survive and recover from disruptions*.

Other disciplines, such as civil engineering, or material sciences define the term ‘resilience’ in different, but highly related ways. The International Council on Systems Engineering (INCOSE) defines resilience as “the ability of organizational, hardware and software systems to mitigate the severity and likelihood of failures or losses, to adapt to changing conditions, and to respond appropriately after the fact” [INCOSE Resilient Systems Working Group]. INCOSE hosts the Resilient Systems Working Group (RSWG) as a node within the international Resilience Engineering Network and actively works in support of The Infrastructure Security Partnership (TISP) to develop more resilient infrastructure. The Resilience Engineering Network defines resilience as the "ability of a system to adjust its functioning, prior to or following changes and disturbances, so that it can sustain operations even after a major mishap or in the presence of continuous stress." Resilient systems have the ability to respond quickly and efficiently to disturbances, are able to monitor for potential threats and anticipate threatening changes in the operating environment [Resilience Engineering Network].

The material sciences definition comes closest to the original Latin where resilience is defined as the material's ability to deform and recover under a load [Ashby, 2007]. In civil engineering, resilience is typically viewed from a perspective of the vulnerability to seismic hazards. Specifically, for a given set of peak ground and floor accelerations, the inter-story drift ratios of various longitudinal layers of a building are compared against the assessed drift ratios of given standards (FEMA-351) to determine a likelihood of damage [Ray-Chaudhuri and Shinozuka, 2010]. However, civil engineering also includes non-building related research, where definitions for resilience tend to include the original Latin meaning where resilience is the ability to 'bounce back' after a disturbance [Reed et al, 2009]. In software engineering, resilience is the resistance of the software to events which threaten to cause it to fail, and how quickly the system can be brought back to an acceptable level of functionality [Axelrod, 2009]. In network engineering, resilience is defined as the expected number of communicating node-pairs in a network. This is referred to as a 2 terminal resilience and later expanded to a special case of k-terminal resilience [Farley and Colbourn, 2009]. For larger systems with n nodes, k-terminal resilience is the expected number of communicating k nodes [Farley and Colbourn, 2007].

Reliability engineering considers resilience as the ability to continue functioning in the presence of faults. This is a derived definition from the reliability community. Reliability itself is defined as the ability of the system to perform its intended functions under stated conditions for a stated duration. Reliability engineers have long looked at these types of issues, finding a connected view of reliability and resilience. They describe reliability engineering as focused on failures, their probabilities and effects, where resilience focuses on an ability to recover once those failures occur [Zio, 2009]. Like reliability engineers, safety engineers have also long examined concepts related to resilience, with a particular focus on accidents and accident avoidance. The safety engineering community defines safety as “the sum of all the accidents that don’t occur,” and resilience is referred to as how well the system handles unanticipated variability [Hollnagel, Woods and Leveson, 2006].

The ecology and systems sciences describe qualitatively similar definitions of resilience. Both use the concept of system state space, and the movement of a system into a new state as a sign of change following a disturbance. An ability to return to an original state is thereby an indicator of the recovery aspects of resilience. The ecology community defines resilience as the “ability of a system to return to normal following a disturbance or stress period” [Mitchell et al, 2000]. A short time to return indicates greater resilience. The Resilience Alliance, an academic research group primarily focused on ecological sustainability and climate change, defines resilience as "capacity of a system to absorb disturbance, undergo change and still retain essentially the same function, structure, identity, and feedbacks" [Resilience Alliance]. The systems sciences and system dynamics field views resilience from a tipping point or stability perspective. Using a topology frame of reference, some systems exist in valleys (depressions) and others on hilltops. Systems in valleys are in stable equilibrium (i.e. more resilient), systems on hilltops are in unstable equilibrium (i.e. less resilient) with respect to some force. Resilience is then the amount of 'force' required to move a system from one of those states into another. For example, more force is required to push a marble up out of a valley into another neighboring valley, than for a marble existing on a hilltop to be pushed into a valley [Sterman, 2000].

Infrastructure resilience is an emerging, cross-domain field of research. The importance of assessing and improving the resiliency of domestic infrastructure became markedly apparent after September 11th, 2001. Prior to September 11th, most infrastructure research focused on earthquake resilience for buildings, roads, power-lines, water systems, etc. This includes the National Earthquake Hazards Reduction Program (NEHRP), led by Multidisciplinary Center for Earthquake Engineering Research (MCEER) at the State University of New York at Buffalo. MCEER defines resilience as "ability of social units (e.g., organizations, communities) to mitigate hazards, contain the effects of disasters, and carry out recovery activities in ways that minimize social disruption, while also mitigating the effects of future disasters." MCEER's definition of resilience includes four properties: robustness, redundancy, resourcefulness, and rapidity, and three characteristics: reduced failure probabilities, reduced consequences of failure, and reduced time to recovery [Bruneau and Reinhorn, 2006]. Mathematically, this definition can be calculated using an integral of the loss of infrastructure quality over the recovery time [O'Rourke, 2007].

The attacks of September 11th, 2001 broadened the view of infrastructure resilience. Broad-based non-profit organizations, such as The Infrastructure Security Partnership (TISP) were established, with stated goals centered on improving the resilience of national infrastructure. The Department of Homeland Security developed an Infrastructure Protection Plan [DHS, 2009] and a presidential directive [Homeland Security Presidential Directive (HSPD) 7] established 17 specific critical infrastructure sectors of national interest. These include agriculture and food, banking and finance, chemical, commercial facilities, communications, critical manufacturing, dams, defense industrial base, energy, information technology, national monuments and icons, transportation systems, and water. Universities established research centers such as the George Mason University Center for Infrastructure Protection and Homeland Security (CIP/HS) in coordination with local and national authorities. CIP/HS promotes a multidisciplinary definition of resilience as combining the traditional material sciences definition and the ecological definition, along with a system's rate of return to normal following a perturbation, and an organization's ability to absorb unexpected challenges[Arsenault and Sood, 2007]. Each of these organizations and their research presented updated, more cross-disciplinary viewpoints on resilience.

Survivability engineering may be thought of as subsumed within resilience, or as an outright field in and of itself. In military domains, survivability is often considered its own field, and is defined as the ability to resist the effects of an adversary's attack, an accident, or a hostile environment. In the civilian domain, survivability is defined in Federal Standard 1037C as "A property of a system, subsystem, equipment, process, or procedure that provides a defined degree of assurance that the named entity will continue to function during and after a natural or man-made disturbance; e.g., nuclear burst". Here we see an overlap in definitions of disturbance with attack and damage, along with the ability to continue operating after that attack or accident (i.e. disturbance).

Outside the engineering and mathematics fields, the Psychology community also includes collective descriptions of resilience in its profession. Here, resilience is the ability to adapt to changing life conditions and recover quickly from stressors (disturbances) [Waugh, Fredrickson, Taylor, 2008]. The psychology community also defines resilience as the display of effective functioning despite exposure to stressful circumstances and/or internal distress [Karoly and Ruehlman, 2006]. Typically, qualitative based scales are used to measure resiliency in psychology.

Resilience has also been examined in the field of workflow management, where workflow is the automation of a business process [Weske, 2010]. Tavana, Busch, and Davis [2011] define the term 'robustness' as the ability to avoid failure, and 'resilience' as the ability to recover from failure. Modeling workflow nets as high-level Petri nets, Tavana, Busch, and Davis examine resilience from the perspective of expected workflow system time and expected repair times for the transitions in the workflow. Robustness is considered using a ratio of best to worst case system times minus a probability that the workflow net is rendered non-functional. The robustness and resilience of a given workflow net is then studied using a two dimensional Cartesian coordinate system with robustness and resilience defining the four quadrants. While substantively different from the resilience evaluation framework described in this research, Tavana, Busch, and Davis [2011] present an important advance in how resilience can be modeled and measured.

Research work in resilience has been conducted by the safety engineering community. Since many systems lacking resilience have resulted in catastrophic failure, much research includes a post-mortem investigation detailing why an accident occurred, and the nature of resilience characteristics which might have precluded it.

Finally, we can also consider resilience with regard to what it means to lack resilience. A lack of resilience is typically referred to as 'brittleness.' Brittle systems are those that are "unable to adapt to unanticipated disturbances or disruptions" [Madni and Jackson, 2009].

The above discussion describes only a representative snapshot of resilience related research work ongoing across many diverse fields. A broad array of other definitions for resilience exists from many fields. As these many fields are working independently, the definitions and approaches are not perfectly consistent. However, each of these definitions contains common threads: disruptions, avoidance, survival, and recovery.

2.1.2. DISRUPTIONS

Resilience includes a notion of disruption and, as with the overall topic of resilience, multiple definitions exist, but with the key theme of an event associated with a loss of performance. The INCOSE defines disruptions as “the initiating event of a reduction is performance. A disruption may be either a sudden or sustained event” [INCOSE Resilient Systems Working Group]. Madni and Jackson [2009] define disruptions as “conditions or events that interrupt or impede normal operations by creating discontinuity, confusion, disorder or displacement.”

Jackson [2010] defines disruptions as events which jeopardize a systems ability to perform its intended capabilities, noting that disruptions can be internal or external. External disruptions are changes related to inputs to the system or the environment in which the system operates. For example, on 15 January 2009 the normal airflow (input) to both engines of US Airways flight 1549 was disrupted when it flew through a flock of birds, leading to the aircraft’s spectacular emergency landing in the Hudson River [NTSB/AAR-10/03, 2009]. The wind-related environment of the Tacoma Narrows Bridge changed on November 7th, 1940 leading to significant oscillation and collapse. While the bridge had previously endured higher wind speeds, the 40-50mph winds on the day of the collapse induced torsional oscillations beyond the capacity of the bridge, leading to its failure [Plaut, 2007].

Internal disruptions are related to the functionality of the system. They can occur at the system level or at the component level. They may be caused by unintended interactions, or by unreliability. The NASA Mars Polar Lander crash is an example of a system level disruption of unintended interactions leading to catastrophic failure. The craft's descent engines had to be shut down nearly immediately upon landing to ensure a stable landing. However, the engine controls received a false positive signal from the vibrating landing leg components incorrectly indicating touch-down. The flight control system prematurely shut down the engines, resulting in an uncontrolled descent and crash [JPL, 2000]. The Apollo 13 spaceflight mission is an example of unreliability. During the Apollo 13 mission, a component level disruption (an oxygen tank explosion) induced a loss of system power and near loss of the mission and crew [Apollo 13 Review Board, 1970]. The spacecraft power system is an example of a mission critical system, whose failure jeopardizes the entire mission. On November 19th, 2009, a router failed in the Salt Lake City location of the FAA flight planning system. The ripple effect of this failure forced manual filings of all flight plans and the widespread delays of flights across the entire country [USA Today, 2009]. On May 6th, 2010, an event known as the "Flash Crash" occurred when US financial markets inexplicably tumbled almost 6% within a matter of minutes but then quickly recovered. The SEC assessed that the built in functions of electronic trading systems actually caused this decline, where some trades were performed at more than 60% away from their value minutes before [SEC, 2010].

Disruptions can be naturally precipitated, unintended, maliciously intended, or non-maliciously intended. Examples of naturally precipitated disruptions include natural events, such as Hurricane Katrina, or the wind induced oscillation responsible for the destruction of the Tacoma Narrows Bridge. An unintended disruption typically occurs as a result of human error or poor human-machine interfaces. The Three Mile Island accident is an example where a combination of human error and mechanical failure played a role in the uncovering of the reactor core and eventual loss of the reactor facility. In this case, the operators were unable to resolve a maintenance issue due to human difficulties in operating the control systems of reactor along with other errors [Kemeny, 1994]. The Chernobyl disaster is an even more compelling example of the role the humans can play in disruptions. In the case of Chernobyl, a series of human errors and non-maliciously intended actions led to the worst accident in civilian nuclear history. The disruption in this case was an unauthorized system test along with a series of deliberate unsafe acts by the controlling operators causing the resultant nuclear explosion [Reason, 1988].

In contrast to naturally precipitated and unintended disruptions, intended disruption are malicious deliberate acts by human agents seeking to cause damage. Intended disruptions are often perpetrated with a specific goal in mind. For example, an internet cyber-weapon computer virus, Stuxnet, targeted and infiltrated the Iranian nuclear facilities in Natanz and Bushehr. The virus is believed to have caused both physical damage to plant equipment as well as overall delays to the Iranian nuclear program. The virus disrupted the centrifuge facilities control processes, forcing them to damage themselves without revealing the damage to operators until too late [New York Times 29September 2010; CBS News, 2010].

2.1.3. The Resilience 'Of What, To What, and Under What Conditions'

The concept of ‘resilience’ is meaningful only if one considers the resilience ‘of what, to what, and under what conditions.’ Carpenter [Carpenter et al, 2001] introduces a notion ‘of what to what’ when describing the magnitude of disturbance an ecosystem can withstand prior to changing states. The work in this research extends the Carpenter et al notion into the systems engineering and development field to include application of threshold requirements and the resilience of capabilities to disruptions. Consider the following examples. A building’s structure may be resilient to an earthquake up to magnitude 5.0 on the Richter scale. A power system may be resilient to a fluctuation in voltage up to +/- 10 volts. A human body may be resilient to blast induced accelerations up to 30g for less than 5ms. Notice that in each case, the concept of resilience included a statement ‘of what’ (building structure, power system, human body) ‘to what’ (earthquake, voltage fluctuation, blast induced acceleration). Further notice that the ‘to what’ statement included threshold parameters (5.0 Richter, +/-10 volts, 30g for less than 5ms). Implicit in these statements is the notion of “under what conditions.’

In addition to Carpenter’s ‘of what, to what’ concept, the evaluation of resilience should also consider ‘under what conditions.’ Here we are concerned with resilience in what situations or in which scenarios. For example the resilience of a command and control system may be different if the command and control system experiences a disruption during peak operations, rather than a disruption that occurs during periods of less intense activities. In the human body acceleration example above, resilience may be possible up to 30g for less than 5ms ‘when the person is properly restrained using a seat-belt.’

In this research, we will consider the resilience of a capability to a disruption. From DoDAF 2.0, a capability is “the ability to achieve a desired effect underspecified performance standards and conditions, thru combinations of ways and means [activities and resources] to perform a set of activities.” Capabilities are a key consideration in this research because disruptions inherently affect the capabilities of a system or an organization. For example, in the case of the disrupted FAA router discussed above, affected airlines lost the capability to file automated flight plans. This loss affected the speed of entry into the air traffic control system, thereby delaying flights nationwide.

Westrum introduces furthers the notion of the resilience ‘of what, to what’ and ‘in what scenario’ through a typology of resilience situations. In regular threat situations, a disruption (threat) occurs often enough that it can be predicted and therefore preparations for handling it can be developed. Improvised Explosive Device (IED) attacks during Operation Iraqi Freedom are an example of regular threat situations where the disruption is almost certain to occur and preparations can be made. In irregular threat situations, the low probability of occurrence of many different types of threats makes preparation nearly impossible. Westrum cites the 2004 suicide bombing of an American mess tent in Forward Operating Base Marez in Iraq as an example of a one-off irregular threat. The final situation type is that of the unexampled event, where the disruption is so unexpected or unimaginable that the response requires an entire change in mental framework. The 9/11 attacks are an instance of unexampled event situations [Westrum, 2006].

2.1.4. Resilience and Time

All dynamical systems, such as the ones considered here, include a temporal aspect. Therefore, an evaluation of resilience must also include time. Timescales vary based upon the system under consideration. However, the timescale can be normalized to allow for more equivalent comparisons. Figure 1 illustrates the significance of time when examining resilience. The evaluation begins at some initial time, defined as time t_0. A disruption occurs at time t_d. The system reaches some minimum operating performance at time t_{min}, and returns to a pre-disruption state at time t_{ret}. The avoidance phase of resilience runs from time t_0 to time t_d, the survival phase runs from time t_d to t_{min}, and the recovery phase runs from t_{min} to t_{ret}. *The work in this research is primarily concerned with the survival phase.*

Performance is evaluated using a Measure of Performance (MoP) for a single capability of the system as described by the architecture. In this research, the evaluation is restricted to a single capability of the system to avoid value judgments about the relative importance and relative contributions of certain capabilities with respect to the operation of some single system. During the avoidance phase, a system is operating at some normal operating level of capability, defined above as $Value_2$ (V_2). When a disruption occurs at time t_d, the level of capability decreases to some minimum value, V_1, at time t_{min}. The system has a minimum threshold level of capability, V_T, below which, performance is deemed un-acceptable, or below which a catastrophic failure could result.

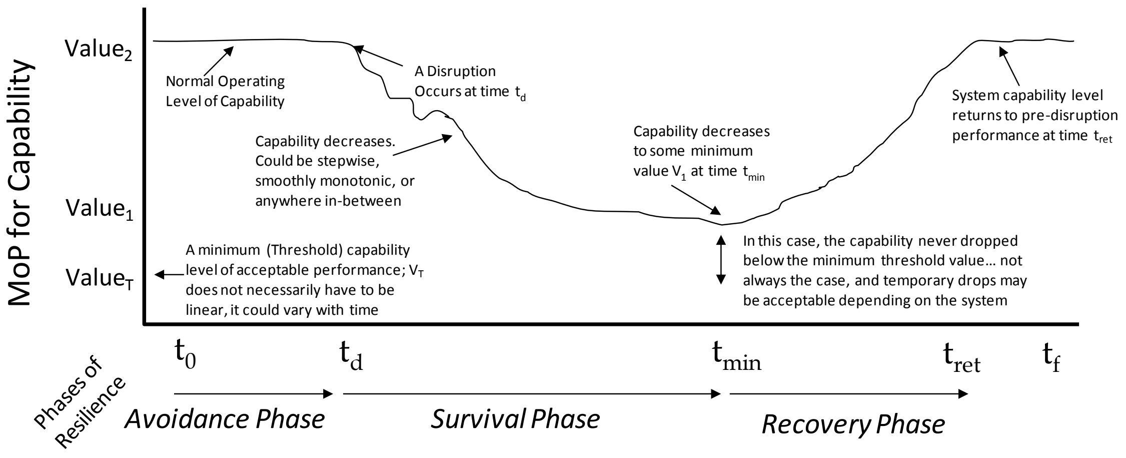

MoP for Capability
Value₂
Value₁
Value_T
Normal Operating Level of Capability
A Disruption Occurs at time t_d
Capability decreases. Could be stepwise, smoothly monotonic, or anywhere in-between
Capability decreases to some minimum value V₁ at time t_min
System capability level returns to pre-disruption performance at time t_ret
A minimum (Threshold) capability level of acceptable performance; V_T does not necessarily have to be linear, it could vary with time
In this case, the capability never dropped below the minimum threshold value… not always the case, and temporary drops may be acceptable depending on the system
t₀
t_d
t_min
t_ret
t_f
Phases of Resilience
Avoidance Phase
Survival Phase
Recovery Phase

Fig. 1: Temporal Aspects in Evaluating Resilience

2.2. THE ATTRIBUTES OF RESILIENCE

On the basis of the existing body of resilience knowledge, Jackson [2010] defines four primary attributes which characterize resilience: capacity, tolerance, flexibility, and inter-element collaboration. This research uses Jackson's resilience attribute descriptions as a foundation and extends them to support the overall evaluation of command and control architecture resilience. Each of these attributes is further defined in Chapter 3.

2.2.1. Capacity

Jackson [2006] defines capacity as "the ability of a system to absorb or adapt to a disturbance without a total loss of performance or structure." Capacity is further defined here as the ability to operate at a certain level as defined by a given measure. In the context of resilience, capacity is the available capability margin between current operating levels and minimum threshold operating levels. It includes the notions of residual capacity and reactive (spare) capacity. Capacity includes an inherent time element because its characteristics may change over time or as a disruption runs its course.

2.2.2. Tolerance

A tolerant system is defined by Jackson [2006] as “one that exhibits graceful degradation near the boundary of its performance.” Tolerance is the ability to degrade gracefully after a disruption or attack. Behaviorally, graceful degradation is the continued operation (potentially at reduced levels) in the presence of faults. From a performance perspective, ‘graceful’ could mean a stepwise downward change of state, or smoothly monotonic degradation in performance within acceptable reaction times.

2.2.3. Flexibility

Citing Westrum [2006], Jackson [2010] defines flexibility as “the system’s ability to restructure itself in response to disruptions.” Flexibility is the ability of a system to reorganize its elements to maintain its capabilities at degraded or even pre-disruption levels. Flexibility is also desired in the case of a change of mission or a change of state.

2.2.4. Inter-Element Collaboration

Jackson [2006] describes inter-element collaboration as communication and collaboration among the elements. Inter-Element Collaboration is the human aspect. It describes unplanned cooperation within a system (typically an organization) to share resources or work together in new ways. Inter-element collaboration accounts for the emergent properties of many systems. Note that while humans tend to be one of the largest initiating sources of disruptions, they are also typically the most resilient part of any system or organization due to the human’s ability to adapt and collaborate in new ways. Due to the difficulties in evaluating emergent properties, inter-element collaboration is not considered in this approach.

2.3. THE ROLE OF ARCHITECTURE

An early definition of the term ‘architecture’ is presented in IEEE 610.12 [1990] as “the organizational structure of a system or component.” This early definition corresponds to the traditional, hardware-based perspective of physical systems. However, hardware, software and human systems are now more closely embedded than ever before, requiring a broader view of the term ‘architecture.’ This trend is especially true in command and control systems. ISO 42010 [2007] expands this definition to “the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution.” Systems architecting is described by Maier and Rechtin [2009] as the art and science of creating and building systems, striving for fit, balance and compromise among competing client needs, resources, technology, timing, and stakeholder interests.

In simple terms, we can view the architecture as the high-level design of the system. Architects develop the overall design, while engineers design and deliver systems which conform to that architecture. By representing the design of a system in a rigorous way, one can analyze the design for key properties, and simulate the design to examine for desired behavior and performance aspects. By evaluating the design in this manner, one can make decisions and improvements far earlier in the process, saving time, money and ultimately delivering better products.

In government and industry, the importance of architectures has increased in response to a host of challenges: the need for increasingly capable systems often with greater complexity; time pressure to deliver new capabilities more quickly, and increased cost pressures. Moreover, requirements tend to change mid-stream in development as a result of changing operational environments, rapid evolutions in technology, and the emerging policy goals of higher institutions. These challenges are especially prominent in command and control systems, which are often considered mission critical and where the technology and innovation cycle is extraordinarily fast when compared to other domains. Architecture-based systems engineering has evolved to cope with these challenges.

The architecture can be analyzed to answer key questions prior to building the system. These include: whether the architecture is logically correct, if it exhibits the intended behavior, whether instantiations of the architecture will meet given performance requirements and provide the desired capability, and in comparison of alternative architectures [Levis, 2008]. Static and dynamic methods have been developed and demonstrated to address these questions. These methods require that the architecture be described in a rigorous manner.

Architectures can be developed using several different design methodologies based on Object Orientation (OO) or Structure Analysis (SA); they can embrace certain architecture patterns such as Service Orientated Architectures (SOA), and they can be expressed using different languages such as the Unified Modeling Language (UML) or a specialized profile such as the Systems Modeling Language (SysML). Structured analysis is a common approach used by many systems engineers. It is based on four primary topics: process or activity modeling, data modeling, rule modeling, and dynamics modeling. Object Orientation is a design methodology based around objects, where objects are unique items with states, properties and behavior which belong to a defined class [Booch, Rumbaugh and Jacobson, 1998]. The Unified Modeling Language, UML, is a widely adopted language used to support primarily an object oriented approach. An extension of UML, SysML is a "graphical modeling language with a semantic foundation for representing requirements, behavior, structure, and properties of the system and its components" [Friedenthal, Moore, and Steiner, 2008]. Service Orientation is an architecture pattern used to challenges in interoperability and is based on "loosely coupled, standards-based, and protocol independent distributed computing" using the concept of services, which are "well defined, self-contained modules that provide standard business functionality and are independent of the state or context of other services" [Papazoglou and Heuvel, 2007].

The executable model, derived from the architecture description of the architecture, is developed using the operational concept. Building the executable model requires that the architecture have progressed sufficiently to generate functional, physical, dynamical and organizational views of the proposed system. If built in a rigorous way, the executable model can support the use of structural and behavioral analyses to address the key questions identified by Levis [2008]. Therefore, generation and analysis of an executable form of an architecture is an essential part of developing complex mission critical systems. Petri nets are one such rigorous approach to developing executable architectures supporting the design and development of complex systems.

2.3.1. Petri Net Use in Architecture Modeling

Petri nets were first developed by Carl Adam Petri in 1962 as a part of his PhD thesis describing communication with automata, where automata are abstract machines. In the following years, Petri net theory has been expanded and widely adopted in a diverse array of science, mathematics and engineering fields. Girault and Valk [2003] describe the advantages of Petri nets for modeling complex system as:

- Rigorous: a formal mathematical model underlying all aspects of Petri net theory

- Graph-theoretical: Petri nets include an interpretation that follows graph theory, allowing for both visual and mathematical explanations with equivalent meanings

- Support Abstraction and Hierarchy: Petri nets include mechanisms for abstraction and decomposition to allow for hierarchical modeling

- Support analysis via defined algorithms: structural evaluation methodologies exist to determine important characteristics of systems to be examined without resorting to simulation.

- Executable: Petri nets support the use of simulation to assess performance and dynamic behavior.

Formally, Petri nets are bipartite directed multi-graphs. Bipartite, means that the graph's vertices can be partitioned into two types of nodes (places and transitions) and the arcs many not connect two nodes of the same type; directed, means that arcs between nodes have directionality associated to them; and multi-graph means that multiple parallel edged may exist between pairs of nodes. Since its inception in 1962, Petri net theory has been significantly expanded. Excellent descriptions of updated Petri net theory and its applications are available in [Jensen and Kristensen, 2007] and [Girault and Valk, 2003].

2.3.2. Architecture Frameworks

Architecture frameworks provide a systemic approach to describing an architecture for a given domain (e.g. defense, commercial, etc.) using a set of inter-related viewpoints (i.e. perspectives) [Tang, Han, and Chen, 2004]. Each viewpoint addresses different aspects of the system to answer different questions; the framework ensures that these products are concordant. Frameworks do not usually include a specific architecture development methodology (e.g. object orientation or structured analysis). A number of well-established architecture frameworks exist, including frameworks from the US Department of Defense (DoDAF), The Open Group (TOGAF), Zachman Framework for Enterprise Architecture, and others. The two case studies used in this research are defense related; therefore further information is presented on the DoDAF.

Recognizing the importance of architectures in system development, the US DoD established the DoD Architecture Framework [DoDAF, 2009], now in version 2.0. DoDAF2.0 provides overarching architecture concepts, guidance, best practices, and methodologies, to enable architecture development in support of DoD weapon systems or transformation endeavors. Because all DoD systems of complexity must include a DoDAF compliant architecture, synergies are beginning to be realized in the inter-relation and interoperation of these systems at the enterprise level. DoDAF 2.0 is an improvement because it focuses on the architectural data, rather than the architecture modeling products, and allows the content to be fit for purpose as opposed to being normative. The DoDAF definition of terms, including the term 'architecture' aligns with ISO 42010:2007. The collection of models shown as viewpoints is referred to by DoDAF 2.0 as an architectural description. The DoDAF does not specify a particular methodology, however DoDAF compliant architectures are very often developed using structured analysis, a UML form of object orientation, or Business Process Model and Notation (BPMN).

2.3.3. Business Process Modeling

Business Process Management (BPM) has evolved in response to a need to automate the processes used to execute a line of business; to automate what is referred to as the 'workflow' [Weske, 2010]. BPM uses UML to develop models of business activities and processes. The BPM extension of UML is referred to as Business Process Modeling Notation, or BPMN. BPMN is a rigorous graphical notation for modeling business processes, with the goal of providing an intuitive approach for all users, yet rigorous enough to be executable. A detailed specification for BPMN was developed in coordination with the Object Management Group, who has subsequently adopted BPMN as a standard [White and Meiers, 2008]. The execution language for BPMN is Business

Process Execution Language (BPEL), with the detailed translation of BPMN to a BPEL simulation engine specified in the OMG BPMN approved specification. Well defined means of translating BPMN to Petri net form are also available including those presented in Raedts et al., [2007], Dijkman, Dumas, and Ouyang, [2008], Stahl, [2005], and Weske, [2010].Further details and an example BPMN model are provided in Chapter 5, in the targeting architecture case study.

2.3.4. Organizational Architectures

To truly examine the resilience of command and control architectures, one must also consider architectures which describe human-centered organizations. Organizational models consider the topics common to other forms of architecture, e.g., structure and inter-relations, but must also consider unique topics such as decision making and workload. Remy, Levis, and Jin [1988] first described decision making organizations in Petri net form using a four stage model and a Lattice algorithm. Damael and Levis [1994] expanded this approach for distributed information systems with variable structure. These initial models evolved over time to a five stage model of decision makers making up an organization, with defined rules for interaction between the decision making nodes [Kansal, AbuSharekh, and Levis, 2007] supporting the design and analysis of a broad range of organizational architectures. A software implementation, CAESAR III (Computer Aided Evaluation of System ARchitectures), was developed at the George Mason University System Architectures Laboratory to assist in the design and analysis of command and control organizational architectures that include variable and adaptive structures as well cultural constraints. Further details and an example decision making organizational architecture are provided in Chapter 5, in the organizational case study.

2.3.5. Progress in Architecture Evaluation

Architecture evaluation describes a process to answer the key questions posed by Levis [2008]: whether the architecture is logically correct, if it exhibits the intended behavior, whether instantiations of the architecture will meet given performance requirements and provide the desired capability, and the comparison of alternative architectures. The goal is to develop a model that can be analyzed from a structural or behavioral perspective. Logical analyses include a check of whether a given input yields the intended (correct) outputs, or whether functions occur in correct sequences. Behavioral analysis can be used to ensure the architecture is reversible, bounded, and to check for deadlocks, or conditions where certain activities (represented as transitions) cannot function. Structural analysis enables connecting structure to behavior by examining the invariants of the architecture. Behavioral analysis includes methods of examining the state space of the architecture via reach-ability and occurrence graphs that depend on initial conditions (initial state). State space analysis allows for a complete examination of all possible states of a given system as well as detailed looks at individual states or the transitions between states. Execution via simulation can be used to assess the performance of an instantiation of the architecture with regard to defined Measures of Performance (MOPs) from which Measures of Effectiveness can be obtained when performance is compared to the given user requirements. The key is to ensure that the executable model is derived in a traceable, rigorous way. The results of these structural and behavioral means of evaluating an architecture can be compared to user requirements and lead to improvements in the design, or to support the down-selection of alternative candidate architectures.

Architecture evaluation is possible independently of the methodology used to design the architecture. This has been demonstrated in Wagenhals et al, [2000], Wagenhals, Haider, and Levis, [2003], and Wagenhals and Levis, [2008]. This methodology relies on the translation of the architecture into Petri net form. This translation is necessary to take advantage of the many strengths of Petri nets described in Section 2.3.1 in support of the overall architecture evaluation. The translation must occur in a traceable, defined, and repeatable manner to ensure that the insights gained during the evaluation of the executable model are relevant to the architecture in its original format. Liles [2008] demonstrated an automatic means of translating UML based architectures into Petri net form. Methods of translating BPMN-based architectures are described in Section 2.3.3.

CHAPTER 3

EVALUATING RESILIENCE

The approach developed in this research uses the architecture of a command and control system to evaluate resilience. When developed, the architecture is translated into Petri net form and evaluated using an approach founded on the attributes of resilience. Multiple measures are developed for each attribute of resilience. The architect and development team selects the appropriate measure and maps the architecture's performance with regard to that metric. This portion of the approach is further described in Section 3.1. Section 3.2 describes a holistic approach of comparing the architecture's performance with respect to the selected metrics against a required performance level for each attribute of resilience. This comparison will be further defined in Section 3.1 as the intersection of requirements and performance. Resilience-related improvements to the design can now be quantified and alternative architectures can be compared. The idea is to evaluate the resilience performance of the baseline architecture against the resilience requirements established by the system developers. Then either compare the baseline against alternative architectures, or make improvements to the baseline to move its performance into a desired range.

3.1. MEASURING THE ATTRIBUTES OF RESILIENCE

As discussed in Chapter 2, Jackson [2010] identifies and defines four attributes which characterize resilient systems: capacity, tolerance, flexibility and inter-element collaboration. The methodology developed as a part of this research begins with the attributes identified by Jackson and then considerably extends each of them to make them more specific and to examine multiple facets of each. Multiple metrics are proposed for two reasons. First, not all C2 systems are the same; therefore, distinct choices in metrics are needed. Second, metrics are also required to direct an architect's attention to areas in the architecture where resilience-related improvements may be warranted. Table 1 summarizes the resilience metrics associated with each attribute of resilience. These metrics are individually described in detail in the remainder of Section 3.1.

3.1.1. Capacity: Buffering, Reactive ad , and Residual

Capacity is the ability to operate at a given level as described by a given measure. Absorptive capacity is the available capability margin between current operating levels and a defined minimum threshold operating level. Absorptive capacity can be considered from three perspectives: buffering, reactive, and residual. The primary question with regard to capacity is whether sufficient unused capacity exists, or can be marshaled in a timely manner in order to buffer the adverse effects of the disruption. Under capacity, the secondary questions to address and measure include: 'can the disruption be absorbed within existing capacity?', 'can new capacity be brought on-line in time?', and (given survival) 'what residual absorptive capacity remains' after a disruption induces its full effect. The associated capacity measures are 'buffering capacity at time t_d', 'reactive capacity at time t_d+t_{rc}', and 'residual capacity at time t_{min}.' t_{rc} is defined as the time required to bring spare (reactive) capacity on-line.

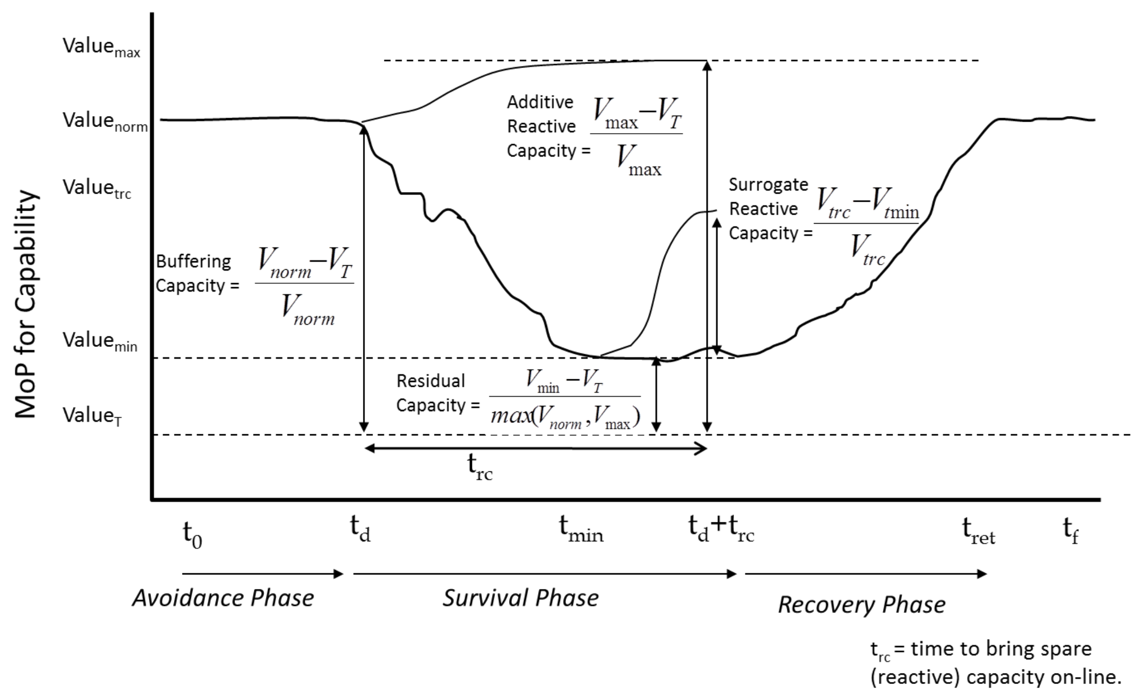
Table 1: Resilience Attribute Metrics

Attribute	Metric	Measures	Question Answered
Capacity: *"the ability to operate at a certain level as defined by a given measure."*	Buffering Capacity	Available capability margin between current operating levels and a defined minimum threshold operating level at the time preceding a disruption.	Can a disruption be absorbed with immediately available (on-hand) resources?
	Reactive Capacity	Available capability margin between maximum operating levels (i.e. including any spare capacity) and a defined minimum threshold operating level.	Can a disruption be absorbed with the addition of spare capacity?
	Residual Capacity	Available capability margin between operating levels at the end of the survival phase and a defined minimum threshold operating level.	Given survival, how vulnerable is the system to a follow-on disruption that occurs before the system can recover?
Tolerance: *"the ability to degrade gracefully after a disruption"*	Rate of Departure	Rate of change in system performance with respect to its requirements (ie rate of loss of effectiveness) after a disruption.	What level of capability is lost per unit of time during the survival phase?
	Fault Tolerance	The ratio of simple functionalities which may be disrupted without a loss of capability to the total number of simple functionalities.	How many simple functionalities can be disrupted prior to losing the capability. Primarily a tool to draw architects attention to key areas in the design.
	Point of Failure Tolerance	Relatedness of failures at the element level to an overall loss of capability	Are element level failures relatively localized, or do failures incur broad system-level effects? Primarily a tool to draw the architect's attention to key design areas.
Flexibility: *"the ability of a system to reorganize its elements to maintain its capabilities"*	Cohesion	Relatedness of the elements within a node or module which support a given capability	How difficult is it to reorganize the system at the node / module level?
	Common Use	Extent of common use of the elements among the simple functionalities which support the overall capability.	Can a system execute multiple functionalities concurrently, or is it limited by competition for resources?
	Proportion of Use	The fraction of the total elements used by any given simple functionality to deliver the overall capability	Are most of the elements needed for a given functionality, making it more difficult to reorganize?

There are three primary means of addressing capacity when time is also considered. Buffering Capacity is the capability margin available immediately at the time of disruption or attack. Reactive Capacity accounts for the fact that certain systems are able to bring additional capacity on line after a given reaction time, defined as t_{rc}. This allows for the system to increase capacity to some improved value. Given a system survives a disruption, Residual Capacity describes the remaining capacity above the threshold requirements and captures system vulnerability to a follow-on disruption that might occur in quick succession to the original disruption. See Fig. 2 for details.

Buffering capacity is the capability margin available immediately at the time of disruption or attack. Buffering Capacity can be calculated, as shown in Fig.2, as a proportion of the difference between the normal operating levels of the capability and the minimum threshold.

Reactive capacity accounts for the fact that certain systems are able to bring additional capacity on line after a given reaction time, defined as t_{rc}. Reactive capacity can be additive or surrogate. In the case of additive reactive capacity, this allows for the system to increase capacity to some maximum value, shown as V_{max} below. This applies when the additional capacity can work in concert with the existing capacity to improve performance above normal levels. Surrogate capacity occurs when the additional capacity does not add value or cannot be combined with existing capacity. Both cases of reactive capacity (Additive and Surrogate) can be calculated in a similar manner to buffering capacity and are also shown in Fig. 2. Chapter 5 includes examples of additive capacity, in the MOC case study, and surrogate capacity, in the Targeting case study. Given a system survives a disruption, residual capacity describes the remaining capacity above the threshold requirements and captures system vulnerability to a follow-on disruption that might occur in quick succession to the original disruption.



Value$_{max}$
Value$_{norm}$
Value$_{trc}$
Value$_{min}$
Value$_T$
MoP for Capability
Additive Reactive Capacity = $\frac{V_{max}-V_T}{V_{max}}$
Surrogate Reactive Capacity = $\frac{V_{trc}-V_{tmin}}{V_{trc}}$
Buffering Capacity = $\frac{V_{norm}-V_T}{V_{norm}}$
Residual Capacity = $\frac{V_{min}-V_T}{max(V_{norm},V_{max})}$
t_{rc}
t_0
t_d
t_{min}
t_d+t_{rc}
t_{ret}
t_f
Avoidance Phase
Survival Phase
Recovery Phase
t_{rc} = time to bring spare (reactive) capacity on-line.

Fig. 2: Measures of Capacity for a Single Capability

Figure 3 shows an example for calculating aspects of the capacity attribute. In this example, the system's normal operating capacity is defined by V_{norm} as 80, and additive capacity can be brought on-line up to a maximum (V_{max}) of 100 following a reaction time of t_{rc}. Following a disruption, performance degrades to a level V_{min}, defined here as 60, but remains above a minimum threshold level of V_T, defined here as 50. In the case of surrogate reactive capacity, spare capacity can be brought on line after a reaction time of t_{rc} to increase performance to a value of 70. The thick line shown in 'bath-tub' curve format represents idealized system performance and the two thin lines arcing upwards show how additive or surrogate reactive capacity might be introduced to improve performance.

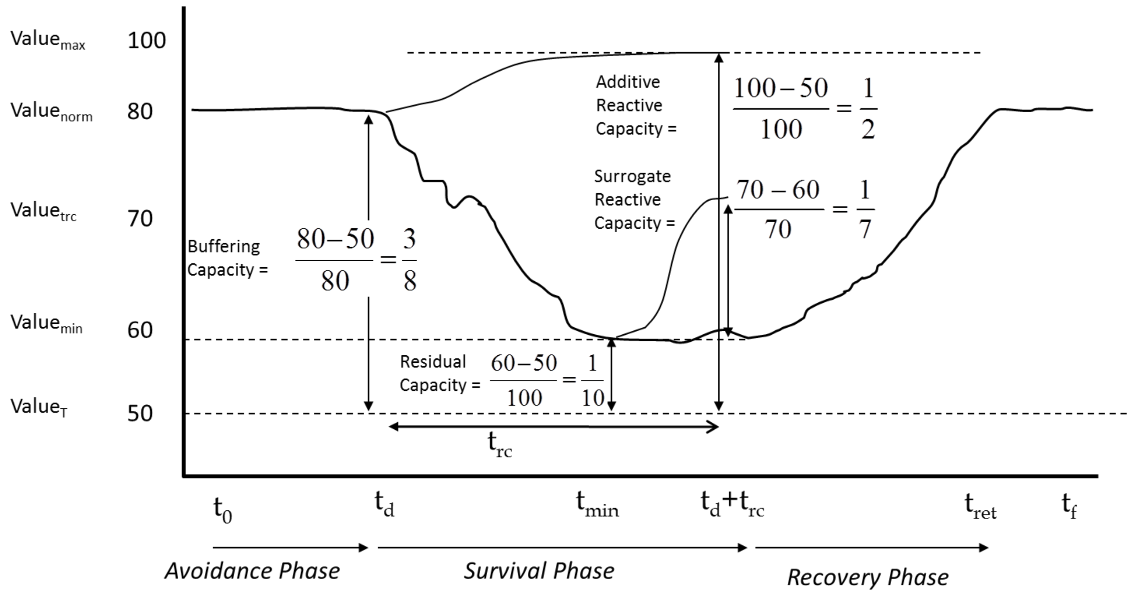


Fig. 3: Example for Calculating Capacity

Note that here we are determining capacity with respect to some minimum threshold value, and not with respect to some absolute maximum capacity. Examining capacity in this manner is related to, but somewhat different than typically considered. For example, we typically say that a given system operates at 80% capacity, meaning with respect to some absolute maximum level of performance of the system as-built. In this framework, we are concerned with a minimum threshold level of performance, below which is deemed unacceptable with respect to a given capability. This is an important change to the equation and is necessary because, in the development phase, we are interested in system performance versus the requirements, and not necessarily versus an instantiated system's absolute capacity.

3.1.2. Tolerance

In addition to capacity, resilience must also consider whether the changing system operating characteristics are tolerable. Tolerance is the ability to degrade gracefully after a disruption or attack. Graceful degradation is the continued operation at reduced levels in the presence of faults, mitigating the effect of the original disruption. A fault is an element-level failure that may cause a reduction in or loss of capability for a given portion of a system. In terms of graceful degradation, we can consider the rate of departure (Tol_{RD}) from normal operating conditions as the rate of change of system effectiveness over time in meeting its requirements. We can also consider fault tolerance, where fault tolerance describes the ability of a system to continue performing its functions in the presence of faults[Rausand and Hoyland, 2004]. Here we are interested in both the fraction of elements that can individually fail prior to a loss of capability (Tol_{FT}), as well as the relatedness of failures to a loss of overall capability (Tol_{PF}).Fault Tolerance (Tol_{FT}) and Point of Failure Tolerance (Tol_{PF}) connect structure to behavior. They are useful tools to draw an architect's attention to areas in the architecture where redesign may be required. The following paragraphs examine all three of these aspects.

3.1.2.1. Tolerance: Rate of Departure

As stated, rate of departure is the rate of change over time in system effectiveness in meeting its requirements. This encapsulates both the temporal aspects of resilience (t_d

and t_{min}), as well as the effectiveness aspects of how the system performs with respect to its requirements and how effectiveness changes during the survival phase (post disruption).Effectiveness can be measured by comparing the system performance with respect to defined Measures of Performance (MoP) against the corresponding requirements. Cothier and Levis [1986] and Bouthonnier and Levis [1984] describe a methodology of comparing system performance to system requirements as the intersection of the locus of performance (L_p) and the locus of requirements (L_r). The two loci, L_p and L_r, are then depicted in a common reference frame. System effectiveness at meeting the established requirements is determined by measuring the intersection of the two loci in the common reference frame. Their metric is shown in Eq. 1.

$$\frac{L_p \cap L_r}{L_p} \tag{1}$$

System performance is characterized by the applicable MoP selected by the system development team. The performance locus (L_p) describes the range of system performance in the defined MoP space as the parameters of various situations are varied according to expected conditions. The requirements locus (L_r) defines the required system performance levels over the same MoP space. This coincidence of the performance and requirements locus has been demonstrated as an architecture evaluation technique to show how effectively a proposed architecture (if built) would meet the stated requirements [Wagenhals and Levis, 2009].

Where the Cothier and Levis [1986] approach is static, this approach adds time. Specifically, the intersection of L_p and L_r is measured at pre-disruption (prior to t_d) and post disruption (at t_{min}) time periods, and computed as shown in Eq. 2, yielding a change of effectiveness per unit of time between those two points (t_d and t_{min}). Rate of departure applies between the pre- and post-disruption periods, and does not extrapolate beyond those areas.

$$Tol_{RD} = \frac{\left[\frac{L_p \cap L_r}{L_p}, t_d\right] - \left[\frac{L_p \cap L_r}{L_p}, t_{\min}\right]}{t_{\min} - t_d} \qquad (2)$$

Rate of departure may also be visualized abstractly, as shown in Fig. 4. In essence we are sampling Eq. 1 at both t_{min} and t_d, and then examine the average marginal rate of departure by dividing by the duration of the survival phase as shown in Fig. 3 and Eq. 2. This provides a measure of the rate of departure over the interval t_d to t_{min} with respect the system's performance against the stated requirements.

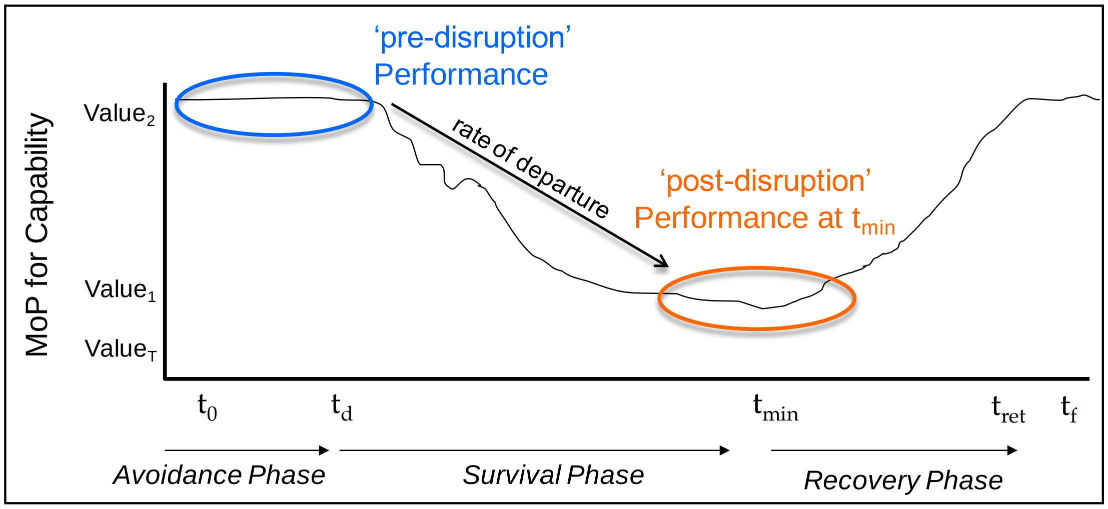


Fig. 4: Abstract Visualization of Rate of Departure

To examine the intersection of the performance and requirements locus, a scenario is required. Parameters of interest (e.g. response time, or inter-arrival time) are varied to form a parameter locus. The executable architecture is simulated at each point in the parameter locus to determine a locus of performance (L_p). L_p is overlaid on L_r to determine system effectiveness at meeting the established requirements.

Figure 5 demonstrates a notional example of how to compute the rate of departure. On the left side the intersection of the locus of performance and requirements is shown at time t_d, essentially capturing normal system performance before the disruption has had any effect. On the right side, the intersection of the locus of performance and requirements is shown at time t_{min}, the end of the survival phase. The overlap in L_r and L_p at time t_d is 0.4 and reduced to 0.05 at time t_{min}. Using Eq. 2 in Fig. 5, we can calculate the rate of departure (Tol_{RD}) for this example; it is 0.035. A lower value is considered better. What this Tol_{RD} means is that for every unit of time between t_d and t_{min}, on average, 0.035 of the overlap in L_r and L_p is lost. More generally, the system's effectiveness is reduced by 0.035 for each unit of time during the survival phase.

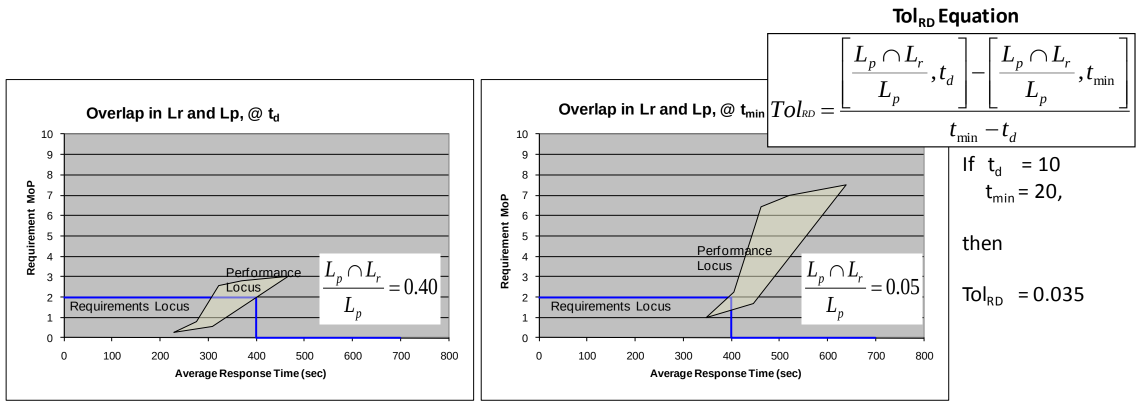


Fig. 5: Rate of Departure Example

3.1.2.2. Tolerance: Fault Tolerance

Resilient systems typically exhibit high fault tolerance: they continue providing their main functionality despite the occurrence of one or more element-level failures. From a structural perspective, one wants to understand how many faults (failures) can occur in the system (as represented by the architecture) prior to a loss of a given capability. A second measure of tolerance, fault tolerance (Tol_{FT}) examines the fraction of elements that can fail prior to a loss of capability.

Applying the graph-theoretical properties of Petri nets, Valraud and Levis [1989] demonstrated the use of Petri net minimum support place invariants with their associated components to describe information flow paths and functionalities in an architecture. In their approach, a simple information flow path corresponds to a simple functionality of the system described by the architecture. A complete information flow path is obtained by coalescing all of the simple information flow paths terminating in a common sink. A complete information flow path corresponds to a complete functionality described by the architecture. A complete functionality is the partially ordered set of functions that generate a specific output. A capability is then the instantiation of one or more related complete functionalities. Petri Net theory subsumes elements of graph theory and provides an opportunity to evaluate resilience using an already developed graph-theoretical foundation.

Fault Tolerance (Tol_{FT}) can now be defined in this case as the ratio of simple information flow paths which may be disrupted prior to the loss of the capability to the total number of simple information flow paths. A capability is said to be 'lost' when the sink or set of sinks (output places) associated with that capability can never have tokens. Relying on the graph-theoretical properties of Petri nets, we are looking for vertices of the net which, when removed, either disconnect the sub-graph or, with regards to a complete functionality, prevent a source input from arriving at its designated sink in the complete functionality, thereby eliminating a capability. From graph theory, such vertices are referred to as "cut vertices." Those elements of the sub-graph (vertices) which can be removed without disconnecting the sub-graph or eliminating the complete functionality (capability) are those that may be disrupted. An overview of the use of graph theory used in this research is available in Chartrand, [1997], Chartrand and Lesniak, [1986], and Grassman and Tremblay, [1996].Equation 3 defines how to compute fault tolerance.

$$\mathrm{Tol}_{\mathrm{FT}} = \frac{x}{r} = \frac{\sum_{i=1}^{\mathrm{r}} x_i}{r} \tag{3}$$

where:

x = the number of disrupted information flow paths

r = total number of information flow paths ℓ

V_{nc} = the set of non-cut vertices

V_c = the set of cut vertices

$$\text{For each } \ell \qquad x_i = \begin{cases} 1 & \text{if } \ell_i \text{ contains a non-cut vertex } (V_{nc}) \\ 0 & \text{if } \ell_i \text{ does not contain a cut vertex } (V_{nc}) \end{cases}$$

Two methods of finding cut vertices exist. The first is based on depth first search, while the second is based on comparison of information flow paths. Both methods produce equivalent results. Returning to graph theory, consider the following lemma [Chartrand and Lesniak, 1986]. For a connected sub-graph *H* with a set of sources *S* and sinks *U* and a set of vertices *V:*

Lemma 1:A vertex *v* is a cut vertex if *H* - *v* increases the number of connected components in *H*.

Lemma 2: A vertex *v*, where *v* is not contained within *U* or *S*, is a cut vertex in a connected sub-graph *H* if and only if for each order pair *(s,u)*, *v* exists on every path from *s* to *u*.

The first method of determining Fault Tolerance (TolFT), uses Lemma 1, and the following algorithm is used. Given a Petri Net *D*, which represents the part of the architecture describing the capability:

1) Determine the minimum support place invariants, *K*, of the architecture and use the pre-set and post-set to construct the components,ℓ,corresponding to each minimum support invariant. (the minimum support components correspond to the simple information flow paths, and analogously, simple functionalities); such that $K = \{k_1, k_2,k_3, \ldots k_r\}$ and $\ell = \{\ell_1, \ell_2, \ell_3, \ldots \ell_r\}$

2) Coalesce the simple information flow paths into a complete information flow path, *H*. *H* is therefore a connected subgraph of *D*, consists of one or more information flow pathsℓ, and represents a single capability. *H* has a set of input sources *S*, as set of output sinks *U*, and a set of vertices *V*, where *S, V*, and *U* are disjoint.

3) Use a Depth First Search (DFS) algorithm as defined in Cormen et al [1990] to determine which vertices of V contained within H will disconnect a source S from a sink U. For each source S:

   a. Let V_{nc} be the set of non-cut vertices within H. Initially, $V_{nc} = \emptyset$

   b. Let tree T be the result of a depth-first search, where the root is a member of S and U is contained within T.

   c. Remove a vertex v_i from T and repeat step 3.a using each root S. If U is contained with the updated tree T, then v_i is not a cut vertex.

   d. If vertex v_i is not a cut vertex, then add v_i to V_{nc} and move to step 3.e, else move directly to step 3.e.

   e. Replace vertex v_i

4) Iterate step 3 for each vertex $v_1 \ldots v_i$ contained within T.

Those vertices which are not cut vertices represent elements of a functionality which may be disrupted in some manner without guaranteeing loss of the capability. (Those which do not disconnect the source and sink are those elements which may be disrupted.) Identify every simple information flow pathℓ which contains a non-cut vertex V_{nc} as a member of ℓ. These represent information flow paths which may be disrupted in some manner without losing the capability under study.

Constructed in this manner, Tol_{FT} will vary between 0 and 1, where higher values are considered better. For example, a Tol_{FT} of zero indicates nearly every element in the design is essential for functionality. This implies very little redundancy, low interconnectivity, and potentially numerous points of failure. A Tol_{FT} of 1 indicates high levels of redundancy, high levels of interconnectivity and possibly no single points of failure.

The following example demonstrates each corresponding step of the algorithm. Petri Net *D* is shown in Fig. 6 representing a capability (or complete functionality).

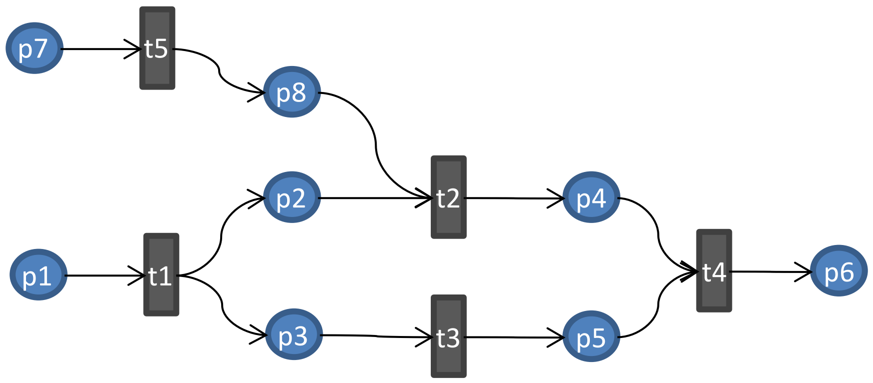


Fig. 6: Petri Net *D*

Step 1) The Farkas [1902] algorithm provides a methodology using Gaussian elimination to solve systems of linear equations with natural number solutions. Martinez and Silva [1982] adapted the Farkas algorithm to solve for the minimal support (linearly independent) place invariants of ordinary Petri nets. Applying the Martinez and Silva adaptation of the Farkas algorithm to Petri Net *D* yields three place invariants:

$K = \{k_1, k_2, k_3\}$so that r is now equal to 3:

k_1 = p1, p2, p4, p6 with component ℓ_1 = p1, t1, p2, t2, p4, t4, p6

k_2 = p1, p3, p5, p6 with component ℓ_2 = p1, t1, p3, t3, p5, t4, p6

k_3 = p7, p8, p4, p6 with component ℓ_3 = p7, t5, p8, t2, p4, t4, p6

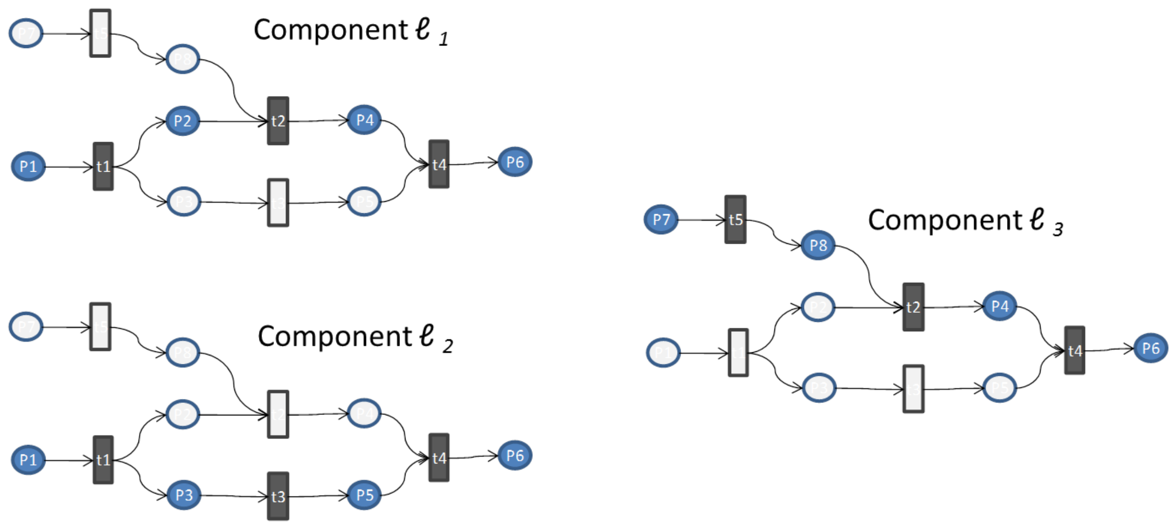


Fig. 7:Minimum Support Components of Petri Net *D*in Graphical Form

Step 2) Coalescing the simple information flow paths, $\{\ell_1, \ell_2, \ell_3\}$, reveals that *D* is included in the union of all three minimum support components.

Therefore, in this simple example case, $H = D$

The set of S sources includes *{p1, p7}*

The set of U sinks includes *{p6}*

The set of V vertices includes {*t1,t2,t3,t4,t5,p2,p3,p4,p5,p8*}

S, U, and V are disjoint

Step 3) For each source S, perform a Depth First Search (DFS) of Tree T

Initially, set $V_{nc} = \varnothing$

S includes two elements, therefore tree $T = \{T_{p1}, T_{p7}\}$

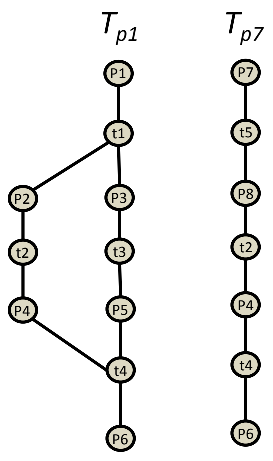


Fig.8: Tree Structure of Petri Net D for Each Source

Note that *T* includes *S*, *V*, and *U*

Remove vertex $t_1$from *T* and perform a DFS for each *S*

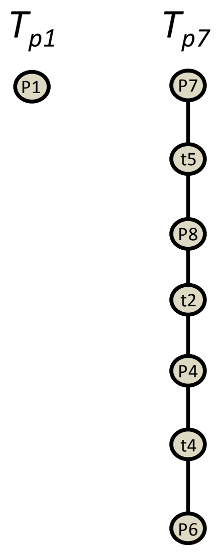


Fig.9: Tree Structure of Petri Net D with Vertex t1 Removed

Note that *U*, *(i.e. {P6})*, is not contained within T_{p1}, therefore, t_1 is a cut vertex because it disconnects the tree (separates *P1* from *P6*)

Replace t_1 and move to the next vertex.

Remove vertex *p2* from *T* and perform a DFS for each *S* in Tree *T*

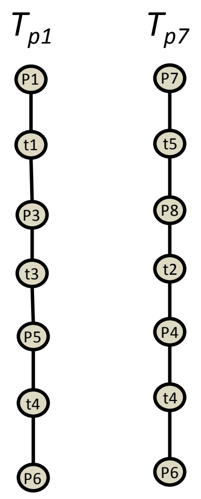


Fig.10: Tree Structure of Petri Net D with Vertex p2 Removed

Note that U is contained within tree $T = \{T_{p1}, T_{p7}\}$, therefore *P2* is not a cut vertex. Add v_i to V_{nc}, where now $V_{nc}= \{P2\}$

Replace P2 and iterate Step 3 for every vertex contained in subgraph H (i.e. check for cut vertices across the entire capability represented in subgraph H)

The results of Step 4 determine the set of non-cut vertices V_{nc}

V_{nc}= {P2,P3,P5,t3}

For each ℓ $$x_i = \begin{cases} 1 & \text{if } \ell_i \text{ contains a non-cut vertex } (V_{nc}) \\ 0 & \text{if } \ell_i \text{ does not contain a cut vertex } (V_{nc}) \end{cases}$$

From Step 4 V_{nc}= {P2,P3,P5,t3}

From Step 2, recall that:

k_1 = p1, p2, p4, p6 with component ℓ_1 = p1, t1, p2, t2, p4, t4, p6

k_2 = p1, p3, p5, p6 with component ℓ_2 = p1, t1, p3, t3, p5, t4, p6

k_3 = p7, p8, p4, p6 with component ℓ_3 = p7, t5, p8, t2, p4, t4, p6

ℓ_1 contains *P2*	$x_1 = 1$
ℓ_2 contains *P3,P5,t3*	$x_2 = 1$
ℓ_3 and V_{nc} are disjoint	$x_3 = 0$

$$\mathrm{Tol}_{\mathrm{FT}} = \frac{x}{r} = \frac{\sum_{i=1}^{r} x_i}{r} = \frac{2}{3}$$

In the above example, two information flow paths may be disrupted in some manner without a loss of capability. A key thrust area of this research is also to provide ways of directing an architect's attention to areas where design improvements maybe needed. One can look at the above results and note that if information flow path (simple functionality) ℓ_3 is disrupted in any way, a loss of capability is more likely than in the case of the other two simple functionalities.

The second method to identify cut vertices uses Lemma 2. Recall Petri Net *D* representing a capability. The Farkas algorithm determined that *D* includes three invariants: $K = \{k_1, k_2, k_3\}$ and we found that that *r* is equal to 3.

k_1=p1, p2, p4, p6 with component ℓ_1=p1, t1, p2, t2, p4, t4, p6

k_2= p1, p3, p5, p6 with component ℓ_2= p1, t1, p3, t3, p5, t4, p6

k_3= p7, p8, p4, p6 with component ℓ_3= p7, t5, p8, t2, p4, t4, p6

Table 2: Cut Vertices

		Elements												
	Invariant	**p1**	**p2**	**p3**	**p4**	**p5**	**p6**	**p7**	**p8**	**t1**	**t2**	**t3**	**t4**	**t5**
Source p1	1	1	1		1		1			1	1		1	
	2	1		1		1	1			1		1	1	
Source p7	3				1		1	1	1		1		1	1

Sinks & Sources
Cut Vertices

From Lemma 2, we know that any vertex that is on every path from sources S_i to Sink U_j is a cut vertex. Elements t_1 and t_4 are on every path from source p1 to sink p6. Elements t_5, p_8, t_2, p_4, t_4 are on every path from source p7 to sink p6 (there is only one path, so all elements are cut vertices in that regard). Therefore, the non-cut vertices, (V_{nc}) are

$V_{nc}= \{p2,p3,p5,t3\}$, (as found in the previous methodology).

$$\text{For each } \ell \qquad x_i = \begin{cases} 1 & \text{if } \ell_i \text{ contains a member of } V_{nc} \\ 0 & \text{if } \ell_i \text{ does not contain a member of } V_{nc} \end{cases}$$

r = total number of information flow paths

ℓ_1 contains p2	$x_1 = 1$
ℓ_2 contains p3,p5,t3	$x_2 = 1$
ℓ_3 and V_{nc} are disjoint	$x_3 = 0$

$$\text{Tol}_{\text{FT}} = \frac{x}{r} = \frac{\sum_{i=1}^{r} x_i}{r} = \frac{2}{3}$$

The results found using the two methodologies are equivalent.

3.1.2.3. Tolerance: Point of Failure Tolerance

While Fault Tolerance (Tol_{FT}) examines the fraction of elements that can fail individually prior to a loss of capability, Point of Failure Tolerance (Tol_{PF}) examines the relatedness of individual failures to a loss of overall capability. When considering faults, it is important to understand the relatedness of failures at the element level to a loss of functionality or a loss of capability; whether single element level failures tend to induce a failure of the entire system or large portions of the system. In engineering systems, we refer to these as points of failure, and at the extreme, single points of failure in the design. The design for any system whose goal is high availability attempts to avoid these points of failure. A system with high point of failure tolerance means that any given fault at the element level is less likely to cause a broad system failure because the effects of its failure are limited to its local simple functionality. Here we define Point of Failure Tolerance ($\mathrm{Tol}_{\mathrm{PF}}$) as the ratio of elements whose effects are limited to their local simple functionality to the total number of elements in that capability.

Here again we use the relationship of information flow paths to simple functionalities, and coalescence of related simple functionalities to complete functionalities as originally described in Valraud and Levis and reviewed in the Fault Tolerance metrics section. Point of Failure Tolerance is defined as the ratio of elements whose effects are limited to their local simple functionality to the total number of elements in that capability.

$$\mathrm{Tol}_{\mathrm{PF}} = \frac{\sum_{j=1}^{E} q_j}{E} = \frac{q}{E} \tag{4}$$

where:

E = total number of elements

q = number of elements with localized failure effects

A localized failure effect means that the element is a member of <u>*only one*</u> simple functionality, information flow path ℓ_i.

$$\text{For each element } e_j, \quad \begin{cases} 1 & \text{if } e_j \text{ is a member of one and only one flow path } \ell_i \\ 0 & \text{otherwise} \end{cases}$$

Returning to our simple example in Petri Net D, we find that seven elements have localized failure effects. Elements *{p2, p3, t3, p5, p7, t5, p8}* are each a member of only one simple functionality. Therefore, q = 7. Visual inspection of D show there are 13 elements, so E = 13. Refer to Fig. 11.

Element	# flow paths associated w/ element	q_j =
p1	2	0
p2	1	1
p3	1	1
p4	2	0
p5	1	1
p6	3	0
p7	1	1
p8	1	1
t1	2	0
t2	2	0
t3	1	1
t4	3	0
t5	1	1
13	21	7

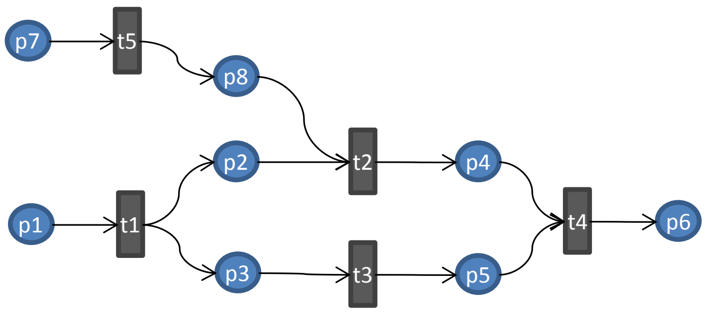


Fig. 11 Determining Localized Failure Effects, q_j

$$\mathrm{Tol}_{\mathrm{PF}} = \frac{\sum_{j=1}^{E} q_j}{E} = \frac{q}{E} = \frac{7}{13}$$

What this means is that 7 of 13 elements are contained within only one information flow path. Therefore, the effects of their failure are localized to that simple functionality. As Point of Failure Tolerance increases, a single fault is less likely to have broad effects, and conversely, as $\mathrm{Tol}_{\mathrm{PF}}$ decreases, a single failure is more likely to have broad system level effects.

A primary thrust of this research is also to enable a methodology to improve design outcomes. This aspect of the research directs an architect's attention to those point failures of the design, especially focusing attention on those elements in the design with the highest number of associated information flow paths. For example, element *t4* in the above example is involved in all three simple functionalities. The capability is highly vulnerable to disruption within element *t4*. Special attention should be paid to this aspect of the design.

3.1.3. Flexibility:

In contrast to tolerance, flexibility is the ability of a system to reorganize and adapt itself to changing conditions. Flexibility is an enabler of adjustment used by many systems to maintain their functionality during the changing conditions which follow a disruption. Flexible systems often have multiple means of providing a given capability, are able to reorganize, and are able to localize failures so that they can be repaired while the remainder of the system continues to function. This may include aspects of redundancy, loose coupling, and adaptable constituent components. Liles [2008] defines cohesion as a measure of the relatedness of the elements within a node. A node is defined by Liles as groupings of elements with internal communication structures, where each element is a member of only one node. Nodes with low cohesion are easier to reorganize than nodes with higher cohesion. Liles also introduces a measure to determine the extent to which the elements of a given node support multiple capabilities, called Degree of Reuse. In this research, we have renamed Degree of Reuse as Common Use, to emphasize the idea of multiple functions relying on the same elements. This renaming is made because the term Degree of Reuse tends to imply a sequencing or temporal aspect which may not be present in many cases. Common Use, or CU, reflects the ability of a node to perform multiple capabilities concurrently. High values for Common Use imply competition for resources, and a low ability to support multiple capabilities concurrently. Low values for Common Use imply less competition for resources and greater ability to execute multiple capabilities concurrently. Departing from Liles, Proportion of Use (PoU) normalizes the Common Use measure. Proportion of Use reflects the relative proportion of the total elements used by any given simple functionality to deliver the overall capability. For example, does the average functionality use 10% of the elements, or 80% of the elements supporting that capability?

3.1.3.1. Flexibility: Cohesion

Liles develops means of measuring the adaptability and agility of system of systems architectures. To address system of systems issues, Liles introduces the concept of a

System Of Systems Instance (SOSI). A SOSI is a subset of the elements available to an enterprise which are organized together for a particular purpose and deliver a specific capability. Liles describes measures for Cohesion and Coupling and uses the inverse of the Cobb-Douglas production function as a measure for Adaptability. Cohesion describes how tightly bound the elements within a node are (their relatedness), and is measured by calculating the number of information flow paths (minimum support s-components) within a node along with its input and outputs. From Liles [2008], Cohesion within a Node (n) is calculated as:

$$\mathrm{Coh}(\mathrm{n}_{\mathrm{ki}}) = \frac{z_{ki}}{x_{ki}} \tag{5}$$

where,

k = the specific system of system instance (SOSI)

z_{ki} = number of paths in node n_{ki}

$\mathrm{x}_{\mathrm{ki}} = \mathrm{I}_{\mathrm{ki}} * \mathrm{Q}_{\mathrm{ki}}$

and where,

I_{ki}= number of inputs for the node

Q_{ki} = number of outputs for the node

From our example in the Fault Tolerance section, if we let Petri Net D = node n_{ki}; then:

$I_{ki} = 2$; $Q_{ki} = 1$; and $z_{ki} = 3$;

Cohesion may then be calculated as

$$\text{Coh}(n_{ki}) = \frac{z_{ki}}{x_{ki}} = \frac{3}{2}$$

When a capability (SOSI f_k) involves more than one node, then the overall cohesion is measured as:

$$\text{Coh}(fk) = \frac{\sum_{i=1}^{m} \text{Coh}(n_{ki})}{m} \qquad (6)$$

where, m = number of Nodes in SOSI f_k

3.1.3.2. Flexibility: Common Use

Common use reflects the ability of a SOSI to support multiple capabilities concurrently. As common use increases, Liles notes that a SOSI's ability to conduct multiple capabilities decreases due to a competition for resources among the elements. Equation 7 below is an equivalent restatement of Liles' equation for computing degree of reuse. Common Use (CU):

$$\text{CU} = \frac{\sum_{j=1}^{E} A_j}{E} = \frac{A}{E} \tag{7}$$

where:

A= number of information flow paths containing element e_j

E = total number of elements

Returning to our prior example using Petri Net *D*:

Element	# flow paths associated w/ element
p1	2
p2	1
p3	1
p4	2
p5	1
p6	3
p7	1
p8	1
t1	2
t2	2
t3	1
t4	3
t5	1
13	21

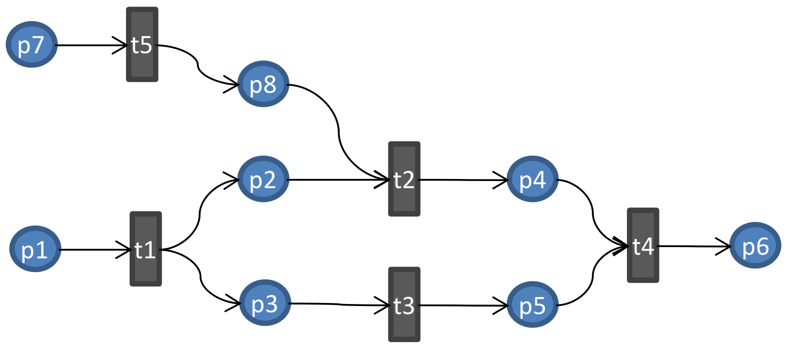


Fig. 12: Associating Information Flow Paths with Each Element

$$\text{CU} = \frac{\sum_{j=1}^{E} A_j}{E} = \frac{A}{E} = \frac{21}{13} = 1.6$$

This means that each element is involved in, on average, about 1.6 information flow paths. Stated another way, each element contributes on average to 1.6 of the simple functionalities associated with that capability. As Common Use increases, competition for resources may increase in the case of concurrent operations, where a particular node is attempting to execute more than one capability. Common Use also indicates that a particular system is able to support more than one functionality with a given element. Departing from Liles, it is important to understand whether the calculated result for CU is a 'low' or 'high,' (good or bad) number. Proportion of Use, defined in the next section, allows us to consider CU in this perspective.

3.1.3.3. Flexibility: Proportion of Use

It's also useful to look at the proportion of the elements within a given capability that are used by any single simple functionality. Proportion of Use (PoU) reflects the fraction of the total elements used by any given simple functionality to deliver the overall capability. For example, does the average functionality use 10% of the elements, or 80% of the elements supporting that capability? Systems with low proportion of use are more resilient to a disruption, since each element is involved in comparatively fewer simple functionalities, and easier to reorganize, because elements are less extensively used in the capability. Systems with high proportions of use are less resilient to disruption, since elements tend to be involved in comparatively more simple functionalities for a given capability, and more difficult to reorganize, because each element is extensively involved in the simple functionalities needed to deliver the overall capability.

$$\text{PoU} = \frac{\dfrac{\sum_{i=1}^{r} B_i}{E}}{r} = \frac{\sum_{i=1}^{r} B_i}{rE} \tag{8}$$

where:

r = total number of information flow paths

B_i= number of elements e contained by path ℓ_i

E = total number of elements

Returning to our simple example in Petri Net D we find:

Information Flow Path ℓ	Elements Contained by ℓ
ℓ 1	7
ℓ 2	7
ℓ 3	7
r = 3	21
E =13	

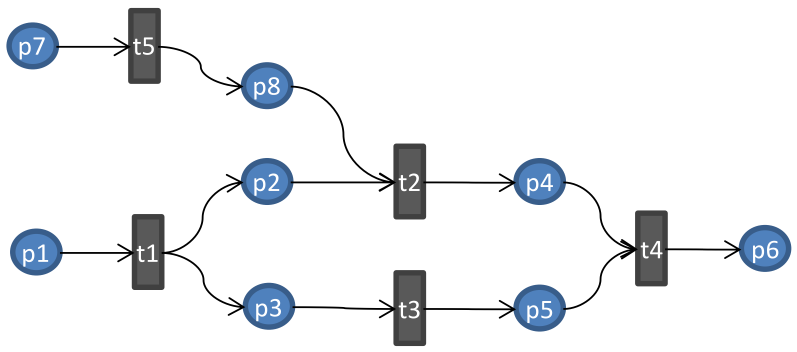


Fig. 13: Calculating Elements Within Each Information Flow Path

$$\mathrm{PoU} = \frac{\frac{21}{13}}{3} = \frac{21}{39} = 0.54$$

This means that each simple functionality uses just over half (0.54) of the elements contained within the architecture representing that capability. Stated another way, the loss of any given element due to a disruption would imply an effect on just over half of the simple functionalities associated with that capability.

This is essentially the same as normalizing Common Use as a proportion of the number of simple functionalities associated with that node. This helps determine whether CU is comparatively low or high with regard to the total number of simple functionalities in that capability. Recall from the previous section on Common Use that it is difficult to determine whether a particular value for Common Use is 'good' or 'bad.' This approach therefore addresses that concern. In this example, we are saying that each element is contained within 1.6 out of 3 simple functionalities, or just over half. A disruption of the average element would therefore affect 0.54 (or about 50%) of the simple functionalities delivering that capability.

Systems with low Proportion of Use are easier to reorganize because the elements are less extensively used in simple functionalities needed to deliver the capability. Systems with high proportions of use are more difficult to reorganize because the elements are more extensively involved in the simple functionalities.

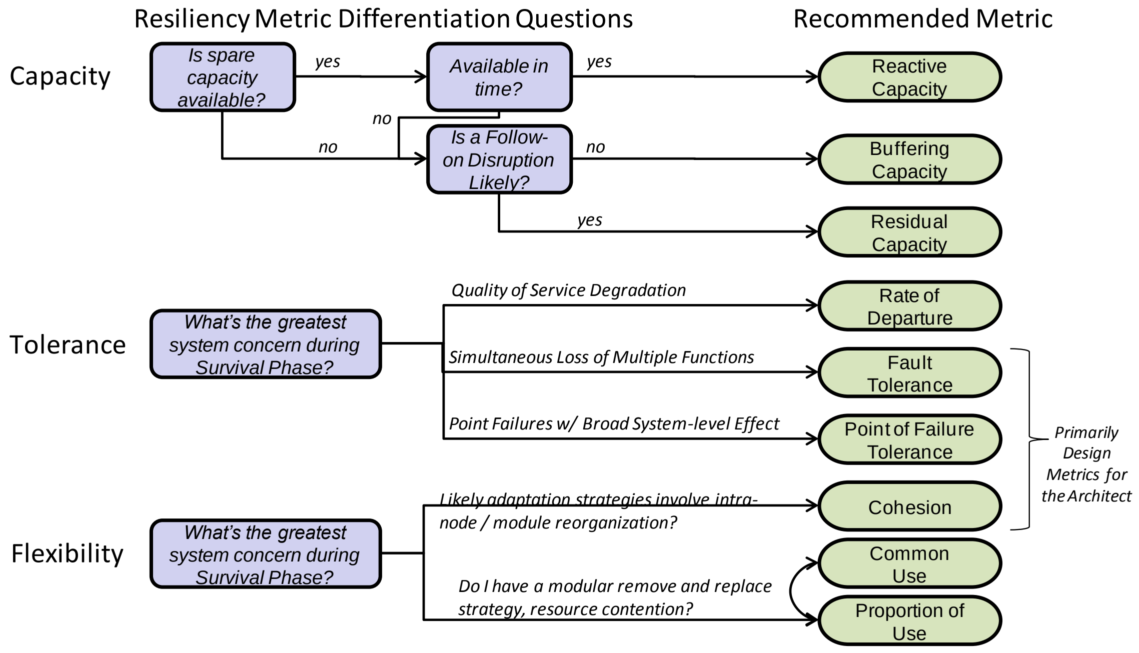
The relationship allows us to examine element usage as a number between 0 and 1, where numbers close to 1 mean a disruption to any element can be expected to have broad reaching effect, and a number close to zero implies a disruption will have a limited effect. Systems with high proportions of use will be less resilient to disruption, since elements tend to be involved in comparatively more simple functionalities for a given capability. Therefore a loss of functionality in a certain element will have more broad effects at the system level. Systems with low Proportion of Use will be more resilient to a disruption, since each element is involved in comparatively fewer simple functionalities. Therefore a loss of functionality will have more isolated effects.

3.2. HOLISTIC APPROACH TO RESILIENCE

Section 3.1 defined each of the individual measures of the attributes of resilience. While each of the individual measures for resilience is useful in assessing one aspect of performance after a disruption, evaluating the attributes of resilience (Capacity, Tolerance, and Flexibility) into a single holistic approach is the goal of this research. The construct presented in this section allows one to map the performance of the architecture as implemented in the executable model into one such approach.

Resilience can be evaluated using the architecture by: identifying the appropriate measure for each attribute; mapping the architecture performance for each measure; and overlaying a requirements locus to assess performance against requirements. Resilience-related improvements to the design can now be quantified and alternative candidate architectures can be compared.

To assist the architect and the overall system development team, a metric selection process is presented in Fig. 14. The architect can use a series of questions regarding each aspect of resilience to gauge in how to proceed with the analysis. For the purposes of improving the design, all metrics may be applicable. However, it's likely that only one of each major area, Capacity, Tolerance, and Flexibility, will be of concern.



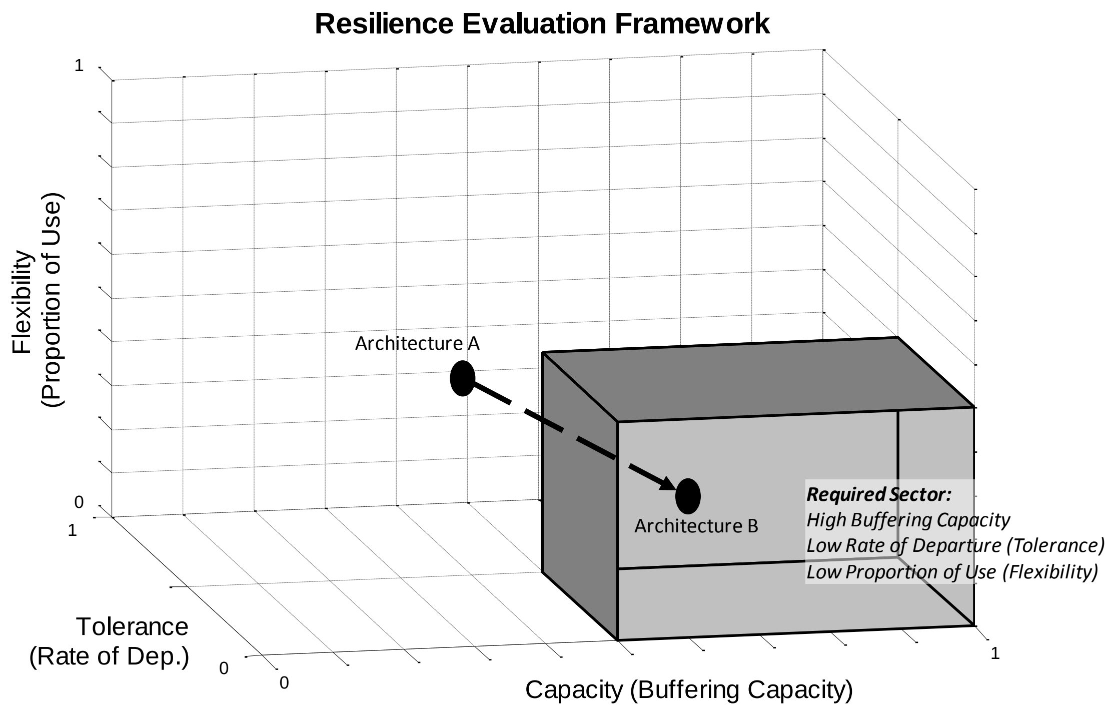
Resiliency Metric Differentiation Questions
Recommended Metric
Capacity
Is spare capacity available?
yes
no
Available in time?
yes
no
Is a Follow-on Disruption Likely?
no
yes
Reactive Capacity
Buffering Capacity
Residual Capacity
Tolerance
What's the greatest system concern during Survival Phase?
Quality of Service Degradation
Simultaneous Loss of Multiple Functions
Point Failures w/ Broad System-level Effect
Rate of Departure
Fault Tolerance
Point of Failure Tolerance
Flexibility
What's the greatest system concern during Survival Phase?
Likely adaptation strategies involve intra-node / module reorganization?
Do I have a modular remove and replace strategy, resource contention?
Cohesion
Common Use
Proportion of Use
Primarily Design Metrics for the Architect

Fig. 14: Resilience Metric Selection

As discussed in Section 3.1, certain metrics are identified in Fig. 14 primarily to help draw an architect's attention to areas in the design where greater focus may be required. For example, in terms of Point of Failure Tolerance, if a given element is used in a majority of the simple functionalities supporting a given capability, the architect may need to investigate whether this situation will be a problem or not. Perhaps the typical components that might instantiate those aspects of the design have a high failure rate. In that case, a design change may be warranted. In terms of Fault Tolerance, where cut vertices are found, the design team may wish to consider alternative architecture designs that minimize their number or location to minimize potential effects. These metrics take advantage of one of the benefits of Petri net analysis techniques, where the structure of the design may be related to behavior. They provide evidence for potential design changes without resorting to simulation or progressing too far in the design phase.

Under the architect's judgment, one measure is chosen from each attribute, such that each axis includes a single measure. Currently, Buffering Capacity, Rate of Departure and Proportion of Use are shown in Fig. 15. The idea is to map the performance of the architecture, and then overlay a resilience requirements locus. Determining the resilience requirements locus requires value judgment and subject expertise. For example, systems with volatility in demanded capacity may require a large buffering capacity to ensure capacity resilience is maintained. Or, for systems where quality of service is paramount, the required tolerance may call for a very low rate of departure to allow for sufficient reaction times by maintenance crews.

Once the architecture's performance is mapped against the metrics and the resilience requirements locus, the architect can either attempt to modify the design to move it into an ideal sector as defined by the resilience requirements or use this framework to compare alternative architectures in a down-selection process. This approach allows us to fulfill a goal of this research: evaluating resilience to assist in selection among alternative architectures or to improve outcomes of design. Figure 15 demonstrates the approach.



Resilience Evaluation Framework
Flexibility (Proportion of Use)
1
0
Tolerance (Rate of Dep.)
1
0
0
1
Capacity (Buffering Capacity)
Architecture A
Architecture B
Required Sector:
High Buffering Capacity
Low Rate of Departure (Tolerance)
Low Proportion of Use (Flexibility)

Fig. 15: Evaluating the Resilience of the System Described by the Architecture

CHAPTER 4

THE RESEARCH APPROACH

4.1. SCOPE

Of the three phases of resilience identified by Jackson [2010] and included on Fig. 1, this methodology is focused on the survival phase. The reason for this is simple: disruptions can be difficult to predict, and even if predictable, difficult to completely prevent. Therefore, avoidance is important, but not always possible or in some cases cost effective. For example, the command and control portion of the US electrical grid is known to be susceptible to malicious disruption via cyber-attack [Nicol, 2011]. However, the funding and manpower resources needed to avoid these potential disruptions are not often available. Furthermore, this work is focused on survival because survival is necessary in order to recover. Additional research is required to determine methods for assessing resilience during recovery phases.

This research considers the resilience of a command and control system’s ability to implement a capability to a disruption. Many command and control systems can execute multiple capabilities simultaneously. For example, a Naval Command and Control system may have the ability to coordinate and control logistical operations, fire support operations, and expeditionary operations. This research does not simultaneously assess all capabilities a command and control system may have. Doing so would require a mechanism to compare the relative value of each capability within the context of the overall mission.

The disruption is defined a priori to the analysis. In the case studies that follow, the disruption is not an amorphous incident selected during the course of simulation or analysis. Rather, it is defined ahead of time by the architect based on subject matter expertise. This allows for a more focused analysis, with more meaningful results. Additionally, the evaluation occurs in the perspective of a given scenario, providing a context for the overall evaluation.

These two ideas, ‘the resilience of a capability to a disruption,’ and ‘defining the disruption a priori’ are in keeping with the concept stated in Section 2.1.3 by Carpenter et al. [2001]. It is important to consider the resilience ‘of what,’ ‘to what’ to have a meaningful evaluation of resilience and this research is scoped accordingly.

4.2. TWO CASE STUDIES

Two case studies are presented in Chapter 5, a time sensitive targeting case and a decision making organization case, each of which is developed using different architecture styles. The decision making (DM) organization is a US Navy Maritime Operations Center (MOC). The time sensitive targeting case is a fire support organization operating in a US Marine Corps ground maneuver unit ashore. Selecting two disparate case studies to demonstrate the research is important for two reasons. First, it shows that the approach is adaptable to the architectural style used to develop the architecture. The approach can be used whether the architecture was developed using Structured Analysis, Object Orientation or other styles. Second, it shows that the approach is agnostic to the type of command and control system under study. The approach can be used for various types of command and control systems, from time sensitive cases to military decision making organizational cases.

4.3. METHOD

This research applies the George Mason University (GMU) System Architectures Laboratory (SAL)architecture evaluation process (see Fig. 16). This process was originally developed by Wagenhals, Haider, and Levis [2003], and provides a macro framework for developing architecture descriptions with rigorous evaluation. The process contains feedback loops and may be iterated to progressively elaborate details in the architecture and in the evaluation.

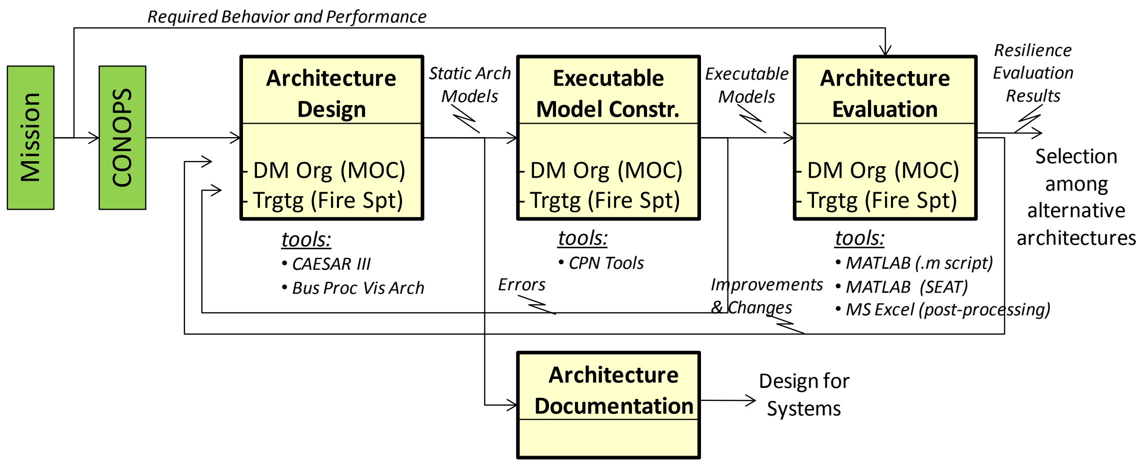


Fig. 16: The Architecture Evaluation Process

From a given mission statement, a CONOPS is developed describing how the organization and assets would accomplish the mission, or in system specific cases, it states how the user would employ the system to conduct the mission. The architecture design is developed based on the CONOPS and other sources, such as doctrine, operating procedures, and other factors. This step generates static views of the architecture and eventually delivers the models and artifacts needed to document the architecture and design relevant systems. The architecture design is transformed into a Petri net – based executable model to allow for logical, behavioral, and performance analyses. Errors identified during the construction of the executable model are fed back into the design. Construction of the executable model is not in itself an evaluation, although its construction may result in a better architecture assuming any errors discovered are fed back into the design. The executable model of the architecture is then evaluated in support of the analysis needs particular to the system or organization being developed. In this research, that assessment consists of the resilience evaluation described in Chapter 3. Improvements are then fed back into the architecture design.

Different tools are used for each case study, and at various stages in the process. The decision making organization architecture is developed using a tool called CAESAR III (Computer Aided Evaluation of System ARchitectures). Developed in the SAL at GMU, CAESAR III is a suite of tools based on Petri net theory to design, analyze, and evaluate command and control organizations and processes. The time sensitive targeting architecture is developed in BPMN using a commercial architecture tool called Business

Process Visual Architect, v4.0, Modeler Edition, under license to the GMU SAL from Visual Paradigm ®.

The Petri net-based executable model of the architecture is derived from the original architecture. In the case of CAESAR III, the executable model is automatically generated in Petri net form. In the case of Business Process Visual Architect, the Petri net executable model is generated per the process described by Raedts et al., [2007]. CPN Tools, is a software program developed by the CPN Group at Aarhus University in Denmark for the development, simulation and analysis of Petri nets. Detailed information on the modeling and analysis of concurrent systems using CPN Tools is presented in Jensen and Kristensen [2007].

MATLAB was used to assist in the evaluation of the executable model with respect to the attributes of resilience involving invariants (simple information flow paths). A MATLAB script, available on MATLAB Central (www.mathworks.com/matlabcentral/fileexchange/6501) and attributed to Hanzálek was used to solve for the invariants of the Petri net executable model. This script, silva.m, takes the incidence matrix as an input and uses the Martinez and Silva adaptation of the Farkas algorithm to solve for the invariants of the Petri net model.

A MATLAB add-in tool, called SEAT (Systems Effectiveness Analysis Tool) version 2.0, [1999], was used to execute the analyses required as a part of the Rate of Departure metric within the tolerance attribute of resilience. The GMU SAL developed SEAT as a part of a suite of architecture analysis and evaluation tools. SEAT allows for the visualization simulation performance results, and for quantifying the intersection of the performance and requirements locus.

Post processing of analysis data from the CPN Tools model was accomplished using MS Excel. The integration of resilience performance with respect to each of the attributes with the overall resilience requirements locus requires three dimensional graphing capability and was accomplished again using SEAT.

CHAPTER 5

CASE STUDIES

This chapter presents two case studies demonstrating an application of the approach described in this research: a targeting case study, and a decision making organization case study. The targeting case study introduces the ideas using a simple example. An excursion is presented in the targeting case study to show the effects of improvements to the architecture. The decision making organization case study shows the comparison of two candidate architectures from a resilience perspective. Each case is introduced with background information, key aspects of the architecture are provided, and the results of the approach are demonstrated.

5.1. TARGETING SYSTEM CASE STUDY

5.1.1. Background

The US Army and Marine Corps are replacing portions of the High Mobility Multi-purpose Wheeled Vehicle (HMMWV) based light tactical fleet with new vehicles. One of the mission role capabilities for these vehicles is that of fire support coordination and synchronization, as conducted by the maneuver unit Fire Support Officer (FSO). This is a time sensitive capability, in which the FSO coordinates fires in support of the maneuver unit as a part of a fire support team. We use the term C2 FSO Vehicle (C2FSOV) in this case study. The C2FSOV is a part of the larger fire support architecture under study. In this case study, we will consider the resilience of the targeting architecture to implement the capability of fire support coordination and synchronization to the disruption (loss) of the GPS network (for example due to cyber-attack or jamming). This forces the fires team to revert to manual geo-positioning means for the target and friendly forces. A high-level operational view of the targeting case study is presented in Fig. 17.

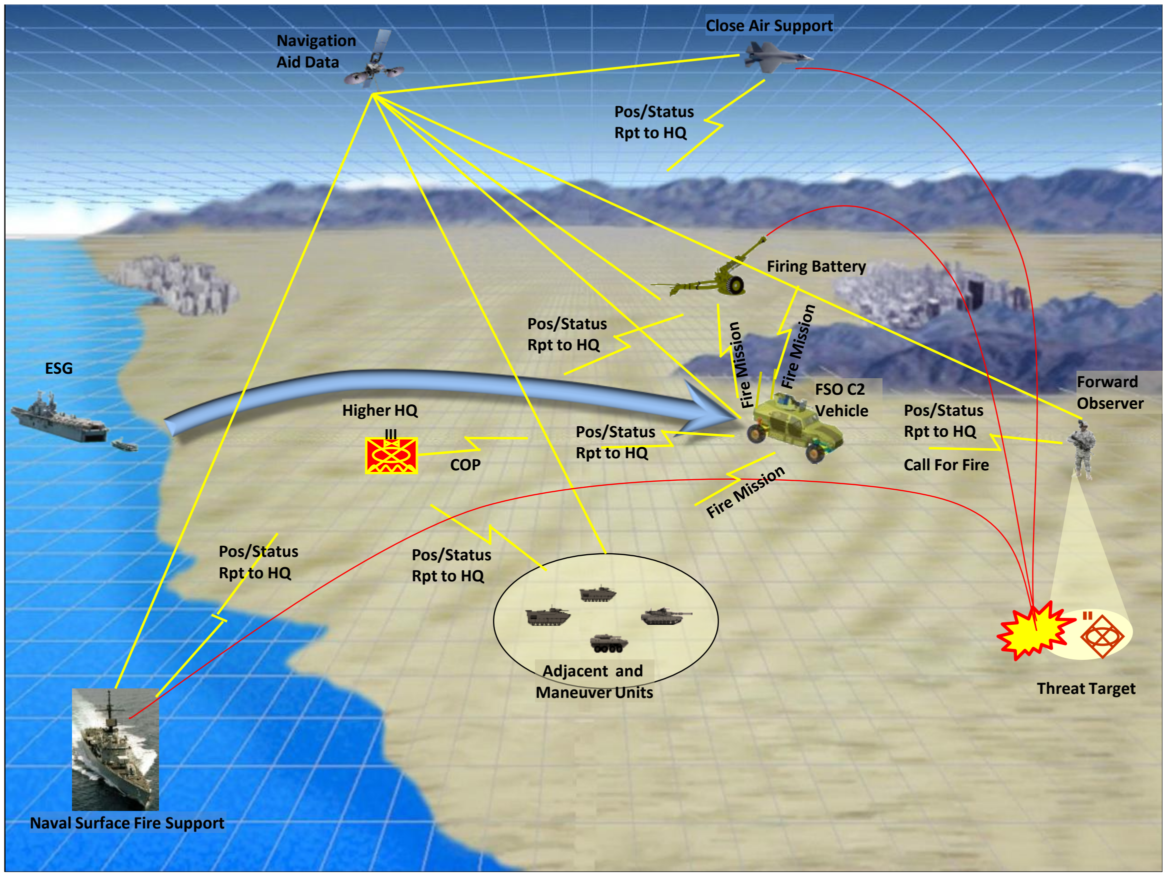


Fig. 17: Targeting Case Study Operational View

In this case study scenario, illustrated in Fig. 17, a Forward Observer (FO) identifies a threat in his sector. The FO sends a Call For Fire (CFF) request to the FSO. The FSO clears the mission using the common operational picture (COP), and assigns a weapon system to fire the mission using fire support software. A fire mission may be assigned to ground artillery/mortars, close air support, or naval surface firing units. These firing units receive the mission, orient their weapons and fire the requested mission. The FO observes the fire mission's effects on the target threat and determines whether to adjust fire based on those effects. Battle command capabilities such as global positioning / navigational aids (GPS), common operational picture and advanced fire support software support this capability.

In this case study, we are examining the resilience of the capability to coordinate and synchronize fire support to disruption of the geo-positioning navigation aid signals (GPS). During early entry operations, cyber-attack to the GPS satellite constellation scrambled the GPS signal, rendering it useless. Loss of GPS affects the fire support process from end to end (FO self-location, target location, clearance of fires, and allocation of weapons). Each portion of the fire support team can still complete the process, but the process transitions to pre-GPS era methods which require much longer times to complete. In this case, soldier common task standards for times to manually complete tasks, versus GPS enabled times are used. No backup is available to offset this loss beyond the older manual approaches (i.e., no reactive capacity).

5.1.2. Architecture

An architectural description of the fire support coordination capability using Business Process Model and Notation (BPMN Specification Version 2.0, 2010) was developed as shown in Fig. 18. BPMN is a standardized Object Management Group (OMG) graphical modeling representation for modeling business processes. It is intuitive to use, and importantly, is mathematically underpinned to allow for translation into executable forms. Critical functions in the fire support coordination capability are fulfilled via navigation aids from the Global Positioning System (GPS) and fire support information from the Advanced Field Artillery Tactical Data System (AFATDS), both of which may be vulnerable to electronic or network attack. In this case study, the approach will evaluate the effect of disruptions to the critical GPS function with regard to the overall fire support coordination capability. In Fig. 18, the portions of the capability most directly affected by a disruption to GPS are colored in light gray.

The BPMN based architecture may be translated into Petri Net form to allow for structural, behavioral and performance analyses. Figure 19 shows the BMPN model in Fig. 18 translated into ordinary Petri net format. To execute the architecture, instrumentation places are added to the model to collect data in support of appropriate analysis needs. Additionally, timing and stochastic features can be added to represent how the system would operate in a given command and control scenario.

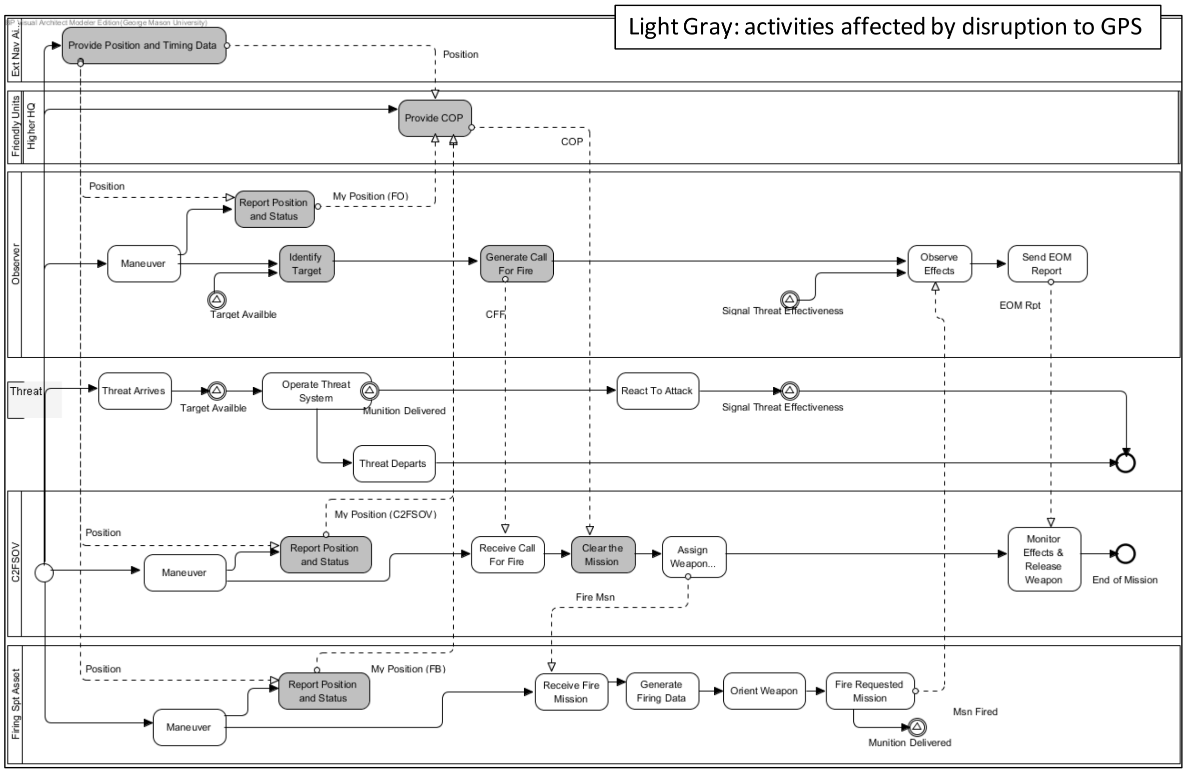


Fig. 18: BPMN Architectural Description of Targeting Case

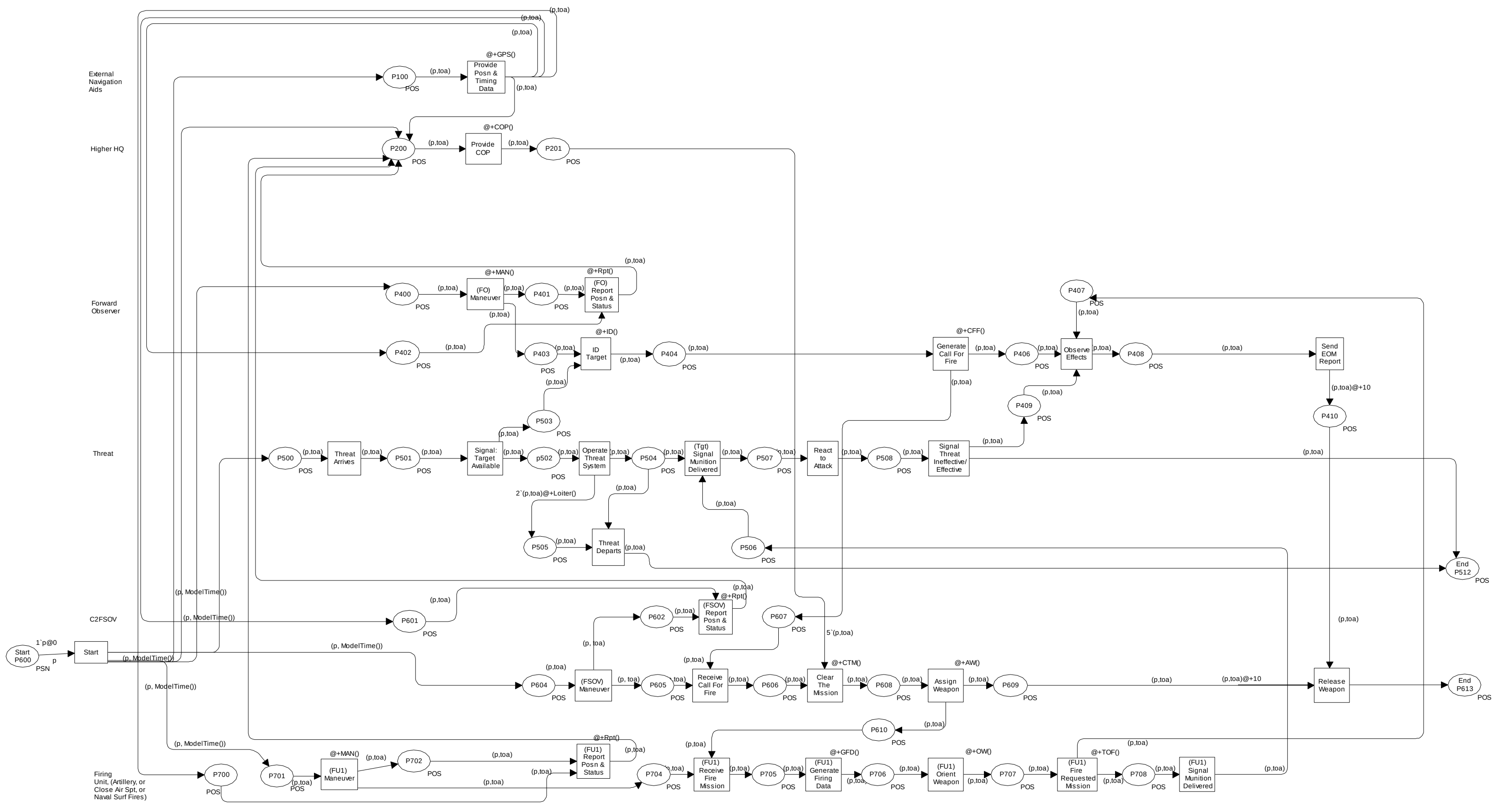


Fig. 19: Petri Net Model Translated from BPMN Architecture

5.1.3. Targeting Case Study Results

Once the architecture is developed, sufficiently verified, and any errors / revisions are addressed, it may be used to support the analyses described in Chapter 3. Figure 14 and Table 1 in Chapter 3 assist the architect to determine which aspects of resilience are most applicable to the architecture definition and resilience issues at hand. In this case study, buffering capacity was determined as the most applicable with respect to capacity. Reactive capacity is not appropriate since no backup (spare) capacity exists. Residual capacity is not appropriate since follow-on disruptions were not anticipated. As a time sensitive targeting case study, Rate of Departure (Tol_{RD}) is the most appropriate aspect of the Tolerance attribute since it addresses quality of service degradation during the survival phase. Proportion of Use (PoU) is the most appropriate aspect of Flexibility, because it addresses how widespread (non-localized) the effects of the disruption may be and assesses the difficulty in reorganizing the system.

5.1.3.1. Targeting Case Study Results: Capacity

To measure capacity in a time-sensitive targeting architecture, the rate at which targets can be serviced is a critical measure of the fire support capability from the FO all the way through effects (rounds impact). This is often considered in terms of a 'window of opportunity' as illustrated in Fig. 20. The remaining window of opportunity to engage the target is a measure of capacity, from the perspective of how quickly a target may be serviced.

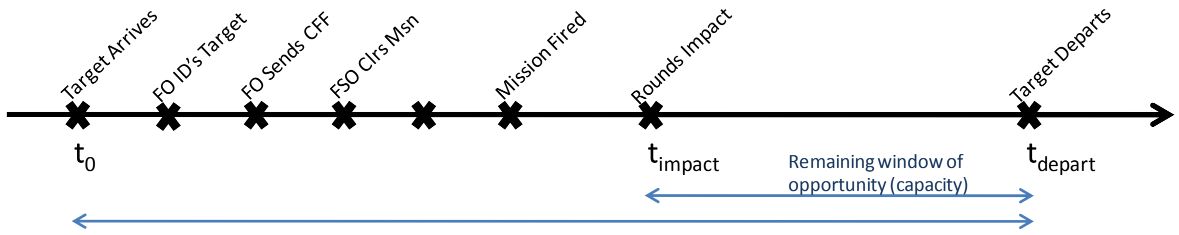
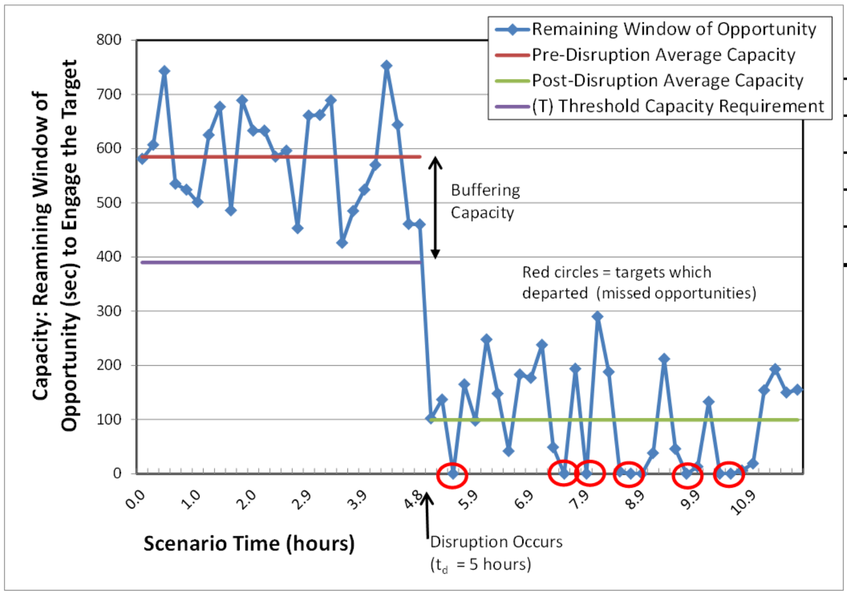


Fig. 20: Capacity as a Window of Opportunity

In this scenario, capacity can be measured as t_{depart} - t_{impact}. When a target departs prior to rounds impact, capacity = zero, meaning that the fire support team could not deliver within the window of opportunity. In executing the Petri Net architecture in this scenario, we find the following results. Recall that the disruption is a loss of the GPS network, causing many formerly automated fire support coordination methods to revert to manual methods. This disruption occurs at scenario time t_d = 5 hours.

Figure 21 shows the results of the capacity analysis for the targeting architecture. In this scenario, the capacity metric is the remaining window of opportunity to engage the target prior to its departure. The threshold shown in Fig. 21 was established in conferral with fire support subject matter experts, and represents a community standard of engaging targets within a certain period following their identification. Buffering capacity was calculated as 33%. Since the post disruption performance dropped below the threshold, residual capacity is effectively zero. Also, since there is no backup (spare) capacity for the GPS system, no reactive capacity exists. The circles indicate missed opportunities, where the target departed prior to rounds impacting.



(T) Threshold Req	390
Pre-Disruption Avg	585
Buffering Capacity	33%
Residual Capacity	0%
Reactive Capacity	None

Buffering Capacity:

$$\frac{V_2 - V_T}{V_2} = \frac{585 - 390}{585} = 0.33$$

Fig. 21: Measuring Capacity in the Targeting Case Study

5.1.3.2. Targeting Case Study Results: Tolerance

Having examined the capacity attribute of resilience, tolerance is also considered. Recall from Chapter 3, we described Rate of Departure (Tol_{RD})as the rate of change over time in system effectiveness in meeting its requirements after a disruption occurs. Effectiveness can be measured by comparing the system performance with respect to defined MoP against the requirements. A parameter locus is generated to determine system performance across the varied parameters it may encounter in a scenario. Given the importance of the window of opportunity in time sensitive cases, the average target loiter time is considered in the parameter locus. Inter-Arrival time between threats is varied to allow for simultaneous engagement of multiple threats. In this case, the difficulty of the task increases as the inter-arrival time decreases and the loiter time increases, meaning effectively more targets. The parameter locus (Fig. 22) allows investigation of the most time sensitive and the least time sensitive situations to generate a system performance response surface. In the capacity chart, the middle point was measured.

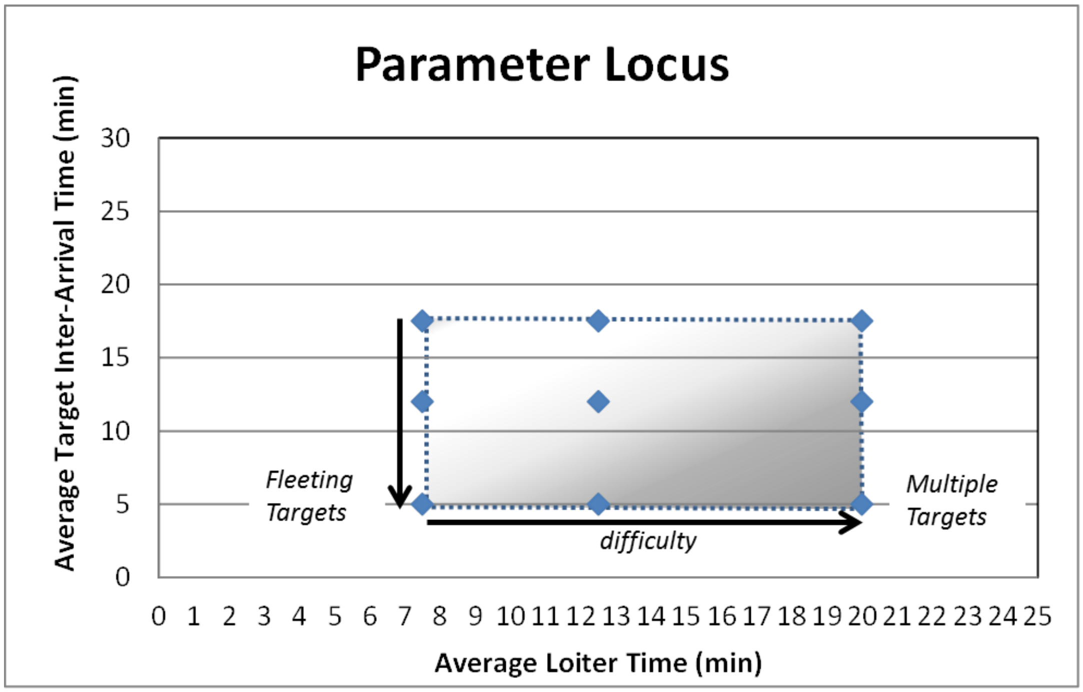


Fig. 22: Targeting Case Study Parameter Locus

Executing the architecture at each point in the parameter locus yields a locus of performance. The performance locus is generated prior to disruption, and again at the minimum point in performance during the survival phase. Recall that rate of departure is described earlier as the rate of loss of effectiveness in meeting the stated requirements during the survival phase. Figure 23 displays pre-disruption performance data, where performance data is collected before the disruption occurs. In this base case, response times are low (i.e. fast) and almost all targets are engaged (i.e., very few missed opportunities or 'leakers'). SEAT was used to generate Figs.23-26and to support the calculations within the rate of departure measure of tolerance.

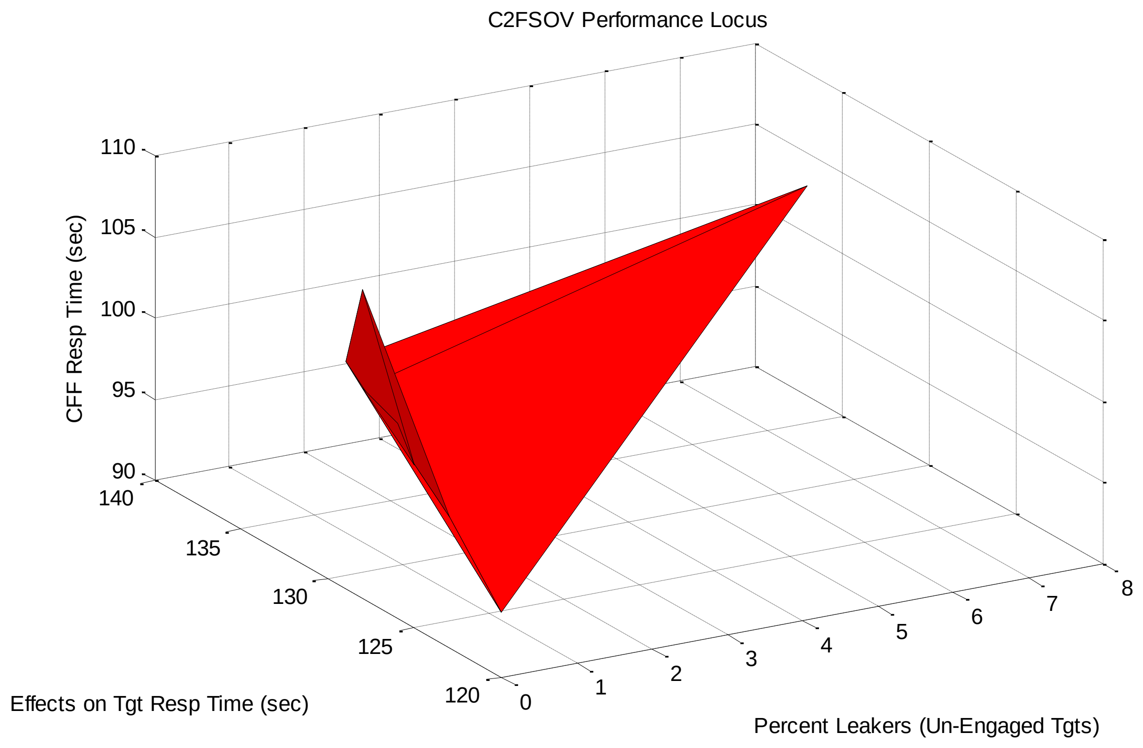


Fig. 23: Pre-Disruption Performance in the Targeting Architecture

Executing the architecture again at each point in the parameter locus, but *after* a disruption, yields a second locus of performance. Figure 24 displays post disruption performance (data is collected after the disruption occurs). In the post-disruption case, as the loiter time decreased in the parameter locus, the fire support team could not move quickly enough without being enabled by GPS. In the most challenging cases of the parameter locus, all targets departed prior to being engaged (i.e. 100% leakers).

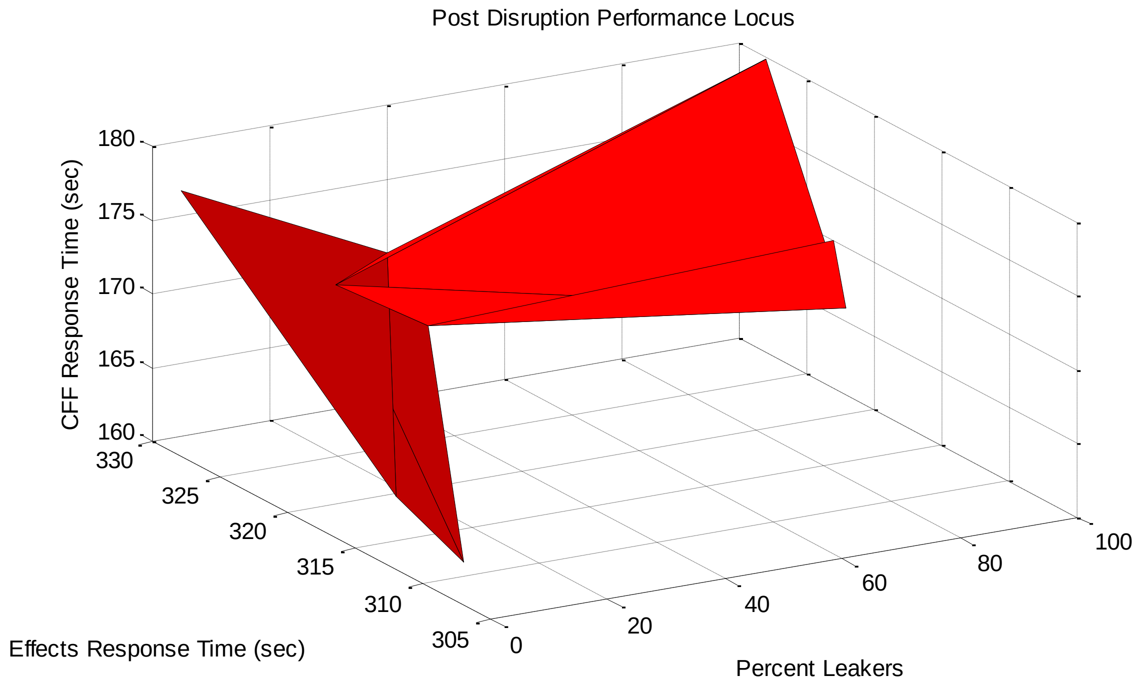


Fig. 24: Post Disruption Performance in the Targeting Architecture

The requirements locus (see Fig. 25) is generated based on the particular requirements for the system. In this time sensitive targeting case, the number of targets departing prior to being engaged (referred to here as 'leakers'), the effects response time, and the call for fire response time are most important.

- Leakers: Requirement is to engage 3 of 4 targets once identified, or < 25% leakers. The criterion is to deliver fires prior to a target's departure.

- Effects Response Time: Time between a target being identified and a munition being delivered. Should be less than 360 seconds.

- Call For Fire (CFF) Response Time: Time used by the fire support team to deliver fires once a call for fire is received from the forward observer (FO). Should be less than 180 seconds.

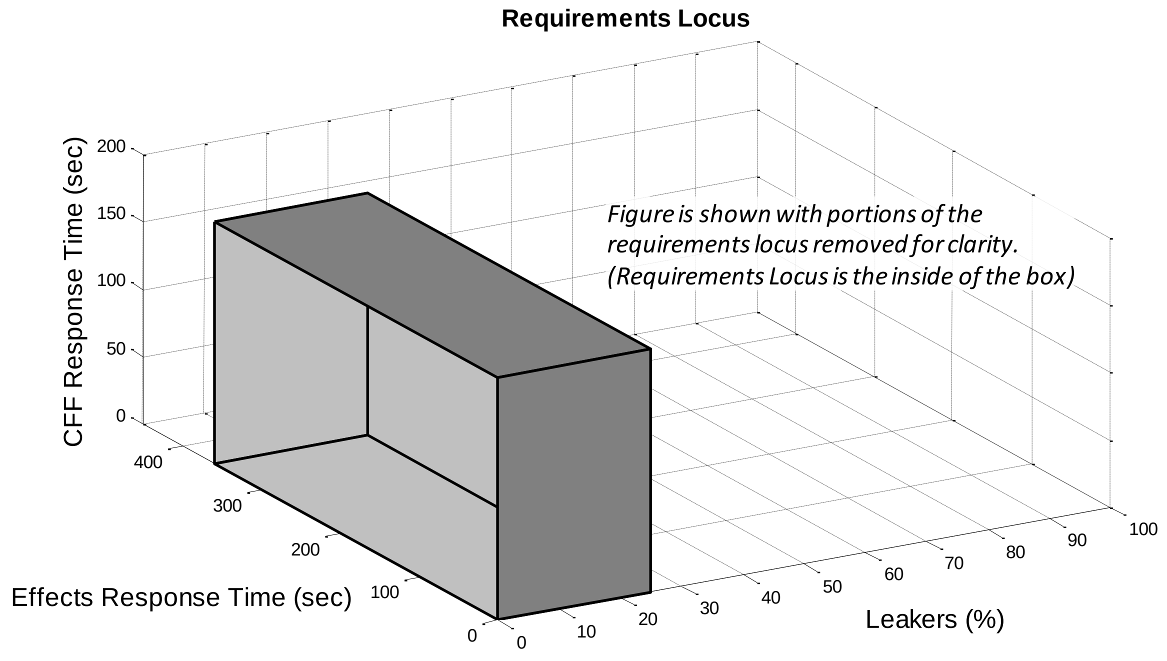
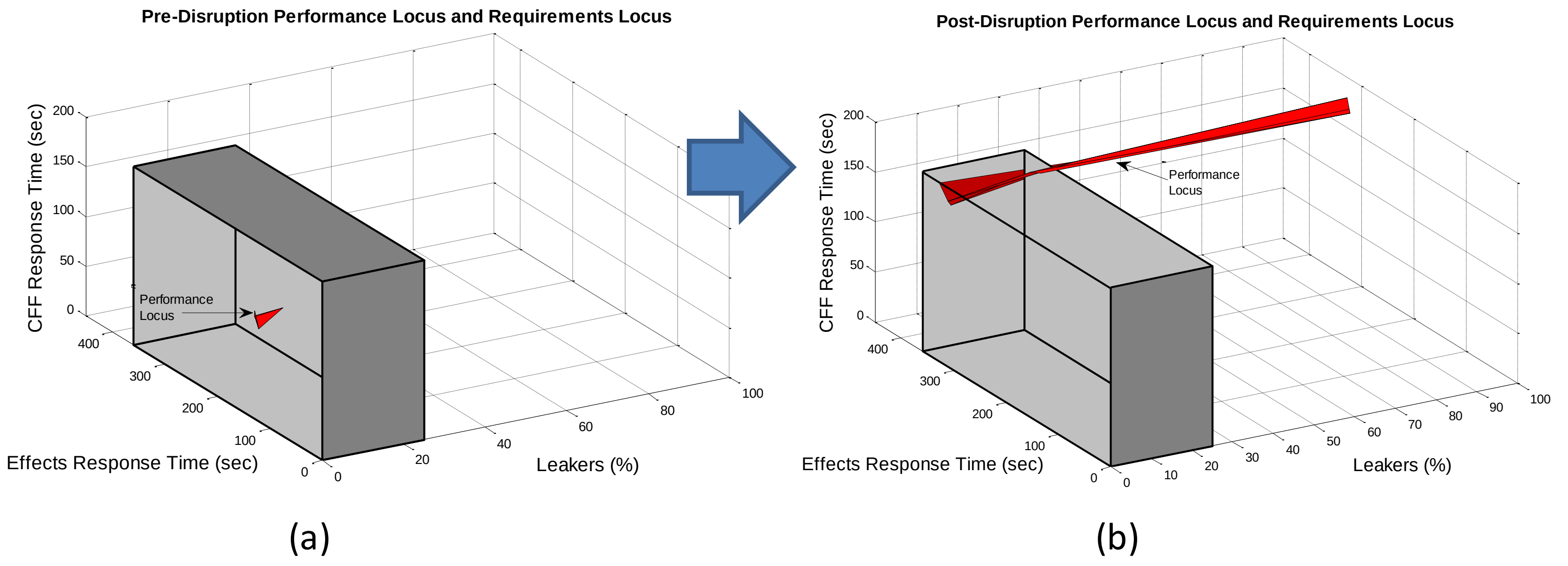


Fig. 25: Requirements Locus

By executing the architecture at each point in the parameter locus, we can generate a performance locus. Figure 26a shows pre-disruption performance. Prior to any disruption, we can see how effective the Fire Support (FST) Capability is when enabled by GPS. The FST capability is able to handle even the multiple, fleeting threats case in the parameter locus. (i.e., very few 'leakers,' fast CFF response times and fast Effects response times). Fig. 26b shows post disruption performance. Here, the simulation results correspond to reality: fire support operations are highly dependent on GPS to engage targets in time sensitive missions. Loss of GPS causes a reversion to manual methods with much longer processing times to geo-locate themselves, targets and to clear fires. This is seen as missed opportunities varies between 0% in the 'long loiter, single target' parameter, to 100% missed opportunities in the 'multiple, fleeting targets' case in the parameter locus. The time sensitive targeting architecture performance degraded from 100% to 66% effectiveness over a course of 17 minutes on average. While the event was instantaneous, its effects took time to occur fully. The rate of departure is ~2% per minute loss of effectiveness.

$$\mathrm{Tol}_{RD} = \frac{\left[\dfrac{L_p \cap L_r}{L_p}, t_d\right] - \left[\dfrac{L_p \cap L_r}{L_p}, t_{\min}\right]}{t_{\min} - t_d}$$

$$\mathrm{Tol}_{RD} = \frac{1.0 - 0.66}{315.1 - 298.9} = 0.02$$



Figures shown with portions of the requirements locus removed for clarity. (The Requirements Locus is the inside of the box)

Fig. 26: Measuring Rate of Departure (Tol_{RD})

In addition to being executable (supporting simulation), Petri Nets include a graph theoretic interpretation, supporting the analysis of properties. The identical model used in the simulations above (see Fig. 19) was also analyzed in static form to assess other aspects of Tolerance and as well as Flexibility. As described in Chapter 3, examining these other aspects of Tolerance and Flexibility requires an ability to determine the information flow paths which form the simple functionalities describing the overall capability under study. The information flow paths are derived from the place invariants in the architecture. The minimum support place invariants of the Petri Net form of the architecture were determined using a MATLAB script (silva.m, available on the MATLAB Central file exchange) which automates the Martinez and Silva [1983] algorithm. Once the minimum support place invariants are known, the minimum support s-components can be determined using the pre-set or post-set of the net.

Fault Tolerance (Tol_{FT}) is one such measure which uses the graph-theoretic properties of Petri nets. Recall that Fault Tolerance (Tol_{FT}) is the ratio of simple information flow paths which may be disrupted prior to the loss of the capability to the total number of simple information flow paths. Those elements of the sub-graph (vertices) which can be removed without disconnecting the sub-graph or eliminating the complete functionality (capability) are those that may be disrupted.

The Martinez and Silva [1983] algorithm finds that the Petri net-based architecture (see Fig. 19) contains 39 simple information flow paths with one source (p600) and two sinks (p512, p613). Following Lemma 2 (See Section 3.2.2.2), we find that:

- sink p512 includes 11 cut vertices: {p500,p501,p502,p507,p508,transition“Threat Arrives”, transition “Signal Target Available”, transition “Operate Threat System”, transition “Threat Departs”, transition “React to Attack”, transition “Signal Threat Effectiveness”}

- sink p613 includes 1 cut vertex: {transition “release weapon”}

- 39 information flow paths are found, therefore, r = 39

Every information flow path contains many non-cut vertices, meaning multiple flow paths exist to connect each source to its corresponding sink.

$\ell_{1\ldots}\ell_{39}$ each contain members of V_{nc} therefore$x_{1..39} = 1$

$$\mathrm{Tol}_{\mathrm{FT}} = \frac{x}{r} = \frac{\sum_{i=1}^{\mathrm{r}} x_i}{r} = \frac{39}{39}$$

Recall that Fault Tolerance is intended to help draw an architect’s attention to areas in the design where improvements or analysis is warranted. In more simple terms, when the architect analyzes the net and finds a series of cut vertices, he or she should begin to analyze these locations. If that cut vertex is associated with a critical simple functionality, or is likely to be allocated to a component with known high vulnerabilities or high failure rates, then alternative designs could be considered to minimize the impact of a disruption occurring at that point. In the targeting architecture, each of the cut vertices associated with sink *p512* are related to activities of the targeted threat. The activities of the targeted threat are important, but not a part of a critical simple functionality. For example, if the target did not show up (cut vertex: *transition "Threat Arrives" is disrupted)*, then the fire coordination capability is not affected. In this particular case, by examining each of the cut vertices found, we can conclude that their presence does not indicate a potential resilience related weakness in the overall design.

Like Fault Tolerance, Point of Failure Tolerance is intended to draw an architect's attention to areas in the design where greater focus may be required. However, where Fault Tolerance looks at the resilience of the capability from the perspective of disruption to the simple information flow paths (simple functionalities)supporting a capability, Point of Failure Tolerance (Tol_{PF}) examines the relatedness of individual failures at the element level to a loss of overall capability. Additionally, for a given disruption it provide a ratio of simple functionalities to the total required in the capability affected by a disruption. Table 3 identifies the number of information flow paths associated with each element in the Petri net architecture of the targeting system (See Fig. 19). It also identifies (in red) the elements associated with the GPS system.

Table 3: Association of Elements to Information Flow Paths

Element Name	# Inf Flow Paths Associated with Element	q =	Element Name	# Inf Flow Paths Associated with Element	q =
P100	24	0	P702	12	0
P200	24	0	P704	2	0
P201	24	0	P705	24	0
P400	16	0	P706	24	0
P401	12	0	P707	24	0
P402	12	0	P708	12	0
P403	4	0	Provide Posn & Timing	24	0
P404	8	0	Provide COP	24	0
P406	2	0	(FO) Maneuver	16	0
P407	12	0	(FO) Report Position and Status	24	0
P408	26	0	ID Target	8	0
P409	12	0	Generate Call for Fire	8	0
P410	26	0	Observe Effects	26	0
P500	6	0	Send EOM Report	26	0
P501	6	0	Threat Arrives	6	0
P502	2	0	Signal Target Available	6	0
P503	4	0	Operate Threat System	2	0
P504	1	1	Threat Departs	1	1
P505	1	1	(Tgt) Signal Munition Delivered	13	0
P506	12	0	React to Attack	14	0
P507	14	0	Signal Threat Ineffective/Effecti	14	0
P508	14	0	Start	39	0
P512	2	0	(FSOV) Maneuver	15	0
P600	39	0	(FSOV) Report Position and Statu	24	0
P601	12	0	Receive Call For Fire	9	0
P602	12	0	Clear the Mission	17	0
P604	15	0	Assign Weapon	33	0
P605	3	0	Release Weapon	37	0
P606	9	0	(FB) Maneuver	14	0
P607	6	0	(FB) Report Posn & Status	24	0
P608	33	0	(FB) Receive Fire Mission	24	0
P609	11	0	(FB) Generate Firing Data	24	0
P610	22	0	(FB) Orient Weapon	24	0
P613	37	0	(FB) Fire Requested Mission	24	0
P700	12	0	(FB) Signal Munition Delivered	12	0
P701	14	0	E = 71	1119	q = 3

From Table 3, we can see that only 3 of the 71 total elements are associated with only one information flow path. Recall that elements associated with only one information flow path have localized failure effects, meaning that a disruption to that element only affects that single simple functionality represented by the flow path. Table 3 shows that

E = 71, and the sum of the q_j = 3. These results imply highly non-localized failures. This interconnectivity means that a failure in one portion has wider spread effects. Only about 4% of the elements are associated with one information flow path

$$\mathrm{Tol}_{\mathrm{PF}} = \frac{\sum_{j=1}^{E} q_j}{E} = \frac{q}{E} = \frac{3}{71}$$

In this case study, we are examining a disruption to the GPS network and its effect on the capability. The Point of Failure Tolerance metric gives us a means to examine this disruption in a specific manner. We can see from Table 3 that GPS related activities (shown in red) play a role in average of 19/39 (~50%) of the simple functionalities in the targeting architecture, meaning that a disruption to the positioning system has widespread implications, affecting about half of the simple functionalities that support the overall capability. These tolerance results correlate well to the behavior shown in the rate of departure analysis. The loss of GPS sharply affected performance of the capability against its requirement. In essence, we are linking structure to behavior. The architect should closely examine whether having fewer simple functionalities rely on GPS might increase the resilience found in terms of Point of Failure Tolerance. An excursion is presented after this case study, using an aerostat navigation beacon, to address potential solutions to this issue of GPS vulnerability.

5.1.3.3. Targeting Case Study Results: Flexibility

In contrast to tolerance, flexibility is the ability of a system to reorganize and adapt itself to changing conditions. From Chapter 3, one measure of flexibility is Cohesion, as defined by Liles [2008]. The Cohesion metric assesses the average cohesion of the individual nodes. A set of more cohesive nodes is more tightly bound (it's inputs and outputs are highly related) and therefore less flexible and less resilient.

In this case, the nodes are reflected by the swim lanes shown in Fig. 18: Navigation Aids; Headquarters and Adjacent Units; Forward Observer, Threat, C2FSOV, and the Firing Unit. Figure 27 demonstrates how to solve for Liles' metric of Cohesion in the Targeting Architecture. The cohesion of each node, *Coh(n$_{ki}$)*, is determined by the number of information flow paths respective to each node/(Inputs x Outputs) of that node. Cohesion for a capability with multiple nodes, *Coh(f$_k$)*, is the average of cohesion calculated for each of the nodes.

$$Coh(n_{ki}) = \frac{z_{ki}}{x_{ki}}$$

$$Coh(\mathrm{f}_k) = \frac{\sum_{i=1}^{\mathrm{m}} \mathrm{Coh}(\mathrm{n}_{\mathrm{ki}})}{m}$$

Cohesion (Multiple Nodes) Targeting Architecture

Node	Inputs	Outputs	Paths	Coh(nki)
Nav Aid	1	4	4	1
HQ	5	1	5	1
FO	6	3	10	0.56
Threat	2	4	6	0.75
C2FSOV	5	2	9	0.9
Firing Unit	3	2	6	1

m= 6 $\quad$ Coh(f$_k$) = 0.87

Fig. 27: Cohesion in the Targeting Architecture

Note, however, that this is really measuring average cohesion of the nodes, and not the entire capability. What it is saying is that the individual nodes themselves are harder to reorganize because they are more cohesive. It is silent on connections between nodes.

A second measure of flexibility is Common Use, also introduced by Liles [2008]. Common Use (CU) measures the extent of common use of the elements to support multiple capabilities. As common use increases, the ability to conduct multiple capabilities simultaneously decreases due to a competition for resources. Table 4 identifies the number of information flow paths associated with each element in the targeting architecture.

Executing Eq. (7) we find

$$\text{Common Use (CU)} = \frac{\sum_{j=1}^{\text{E}} \text{A}}{E} = \frac{1119}{71} = 15.8$$

What this means is that each element in the Fire Support Capability is a member, on average, of 15.8 simple functionalities. As common use increases, a greater percentage of the elements are needed to execute each simple functionality. This leads to competition for resources when multiple functions must occur concurrently. From Common Use alone, it is difficult to determine whether 15.8 is high or low, good or bad. Proportion of Use, described next, addresses this issue.

Table 4: Common Use

Element Name	# Inf Flow Paths Associated with Element
P100	24
P200	24
P201	24
P400	16
P401	12
P402	12
P403	4
P404	8
P406	2
P407	12
P408	26
P409	12
P410	26
P500	6
P501	6
P502	2
P503	4
P504	1
P505	1
P506	12
P507	14
P508	14
P512	2
P600	39
P601	12
P602	12
P604	15
P605	3
P606	9
P607	6
P608	33
P609	11
P610	22
P613	37
P700	12
P701	14
P702	12
P704	2
P705	24
P706	24
P707	24
P708	12
Provide Posn & Timing	24
Provide COP	24
(FO) Maneuver	16
(FO) Report Position and Status	24
ID Target	8
Generate Call for Fire	8
Observe Effects	26
Send EOM Report	26
Threat Arrives	6
Signal Target Available	6
Operate Threat System	2
Threat Departs	1
(Tgt) Signal Munition Delivered	13
React to Attack	14
Signal Threat Ineffective/Effective	14
Start	39
(FSOV) Maneuver	15
(FSOV) Report Position and Status	24
Receive Call For Fire	9
Clear the Mission	17
Assign Weapon	33
Release Weapon	37
(FB) Maneuver	14
(FB) Report Posn & Status	24
(FB) Receive Fire Mission	24
(FB) Generate Firing Data	24
(FB) Orient Weapon	24
(FB) Fire Requested Mission	24
(FB) Signal Munition Delivered	12
E = 71	∑ A = 1119

A final measure of flexibility is Proportion of Use (PoU). Recall that Proportion of Use reflects the relative proportion of the total elements used by any given simple functionality to deliver the overall capability. As proportion of use decreases, a disruption is more likely to have more isolated effects. Systems with low proportions of use are easier to reorganize and more resilient to a disruption, since each element is involved in comparatively fewer simple functionalities. In the targeting architecture, a disruption to a given element can be expected to affect, on average, about 40% of the simple functionalities that support the overall capability under study, as shown in Fig. 28.

Information Flow Path	# Elements Contained by ℓ_i	Information Flow Path	# Elements Contained by ℓ_i
ℓ = 1	15	ℓ = 21	32
ℓ = 2	15	ℓ = 22	32
ℓ = 3	15	ℓ = 23	34
ℓ = 4	17	ℓ = 24	32
ℓ = 5	17	ℓ = 25	34
ℓ = 6	19	ℓ = 26	34
ℓ = 7	13	ℓ = 27	36
ℓ = 8	19	ℓ = 28	37
ℓ = 9	21	ℓ = 29	39
ℓ = 10	21	ℓ = 30	33
ℓ = 11	23	ℓ = 31	27
ℓ = 12	21	ℓ = 32	38
ℓ = 13	23	ℓ = 33	40
ℓ = 14	23	ℓ = 34	40
ℓ = 15	25	ℓ = 35	42
ℓ = 16	29	ℓ = 36	40
ℓ = 17	31	ℓ = 37	42
ℓ = 18	25	ℓ = 38	42
ℓ = 19	19	ℓ = 39	44
ℓ = 20	30	r = 39	∑ Bi = 1119
			E = 71

$$\text{PoU} = \frac{\frac{\sum_{i=1}^{r} B_i}{E}}{r} = \frac{\sum_{i=1}^{r} B_i}{rE}$$

$$= \frac{1119}{71*39} = \frac{1119}{2769} = 40.4\%$$

Fig. 28: Proportion of Use in the Targeting Case Study

5.1.3.4. Targeting Case Study Overall Results

Having examined each of the attributes (capacity, tolerance, flexibility), we can now use the architecture to evaluate the expected resilience of the proposed design. Earlier in Section 5.1.3, the key measures for each attribute most applicable to the case study were identified and the architecture assessed against each measure. We can now map that performance against an overlaid requirements locus to evaluate resilience. Determining the resilience requirements locus requires value judgments by the key stakeholders responsible for the architecture's development. In this case, the requirements locus was specified as shown in Table 5 and drawn as a gray box in Fig. 29.

Table 5: Resilience Required Values and Achieved Performance Values

Attribute	Selected Metric	Measure	Required Value (L_r)	Architecture Performance Value (L_p)
Capacity	Buffering Capacity	Available capability margin between current operating levels and a defined minimum threshold operating level at the time preceding a disruption.	> 50% Buffering Capacity	0.33
Tolerance	Rate of Departure	Rate of change in system performance with respect to its requirements (ie rate of loss of effectiveness) after a disruption.	< 50% Rate of Departure (Tolerance)	0.02
Flexibility	Proportion of Use	The ratio of the total elements used by any given simple functionality to delivery the overall capability	< 50% Proportion of Use (Flexibility)	0.40

As shown in Fig. 29, the requirements locus is depicted as the box, where acceptable performance with respect to the attributes of resilience is defined as the interior of that box. The proposed architecture design is sufficient with regard to tolerance and flexibility, but lacks required capacity. Increased buffering capacity is required to ensure that the system can perform effectively given a disruption to the GPS network. Improving capacity would move the design into the required space with respect to resilience.

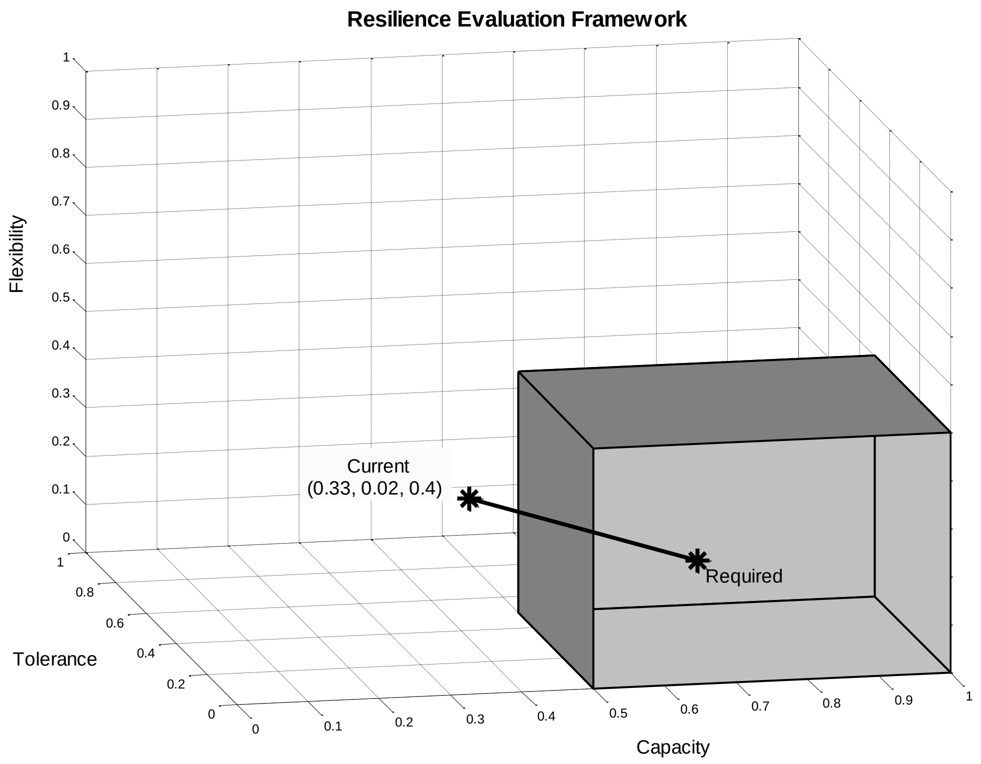


Fig. 29: Resilience Evaluation

This particular case study examined a single architecture. In this case, the resilience evaluation methodology identifies areas where design improvements can be made to move performance into a required sector. Alternatively, given two or more candidate architectures, this same approach can be used to support the selection between alternative architectures. In that case, the performance of both architectures could be evaluated to determine which meets the required resilience performance.

This case study provided a simple, but realistic demonstration of how to quantifiably evaluate resilience using the architecture. A targeting architecture case study was described in BPMN format per current DoD development practices. The BPMN architecture was translated to Petri Net form and analyzed to evaluate the effects of a disruption to GPS availability in the context of the resilience metrics developed in this approach

5.1.4. Targeting Case Study: Aerostat Excursion

We can see from the overall results, that the targeting case study architecture is deficient in terms of capacity. The fire support community would like to engage targets in the first half of the window of opportunity, allowing plenty of time to re-engage as needed. This leads to a desire for a resilience measure of 50% capacity, as we've seen in Fig. 29. However, the targeting case study did not meet this resilience requirement, achieving only 33% buffering capacity.

Given the already fast performance of the GPS enabled system, it is unlikely that the currently organized and equipped fire support team will meet a 50% capacity threshold. However, the lack of an alternative to GPS is worth investigating in terms of sub-standard capacity performance. This alternative to GPS is essentially reactive capacity and is addressed in this excursion.

The targeting case study shows that the US military is highly dependent on GPS for a wide variety of tasks. Forward observers use it to geo-locate themselves and potential targets. Fire Support Officers use GPS to assist in clearance of fires. Friendly forces use GPS to geo-locate themselves and report their position to higher headquarters and surrounding units as part of the Common Operational Picture. The Point of Failure

Tolerance analysis showed that disruption to the GPS network affected ~50% of the simple functionalities supporting the capability of ‘fire support coordination and synchronization’ within the targeting architecture.

Well aware of this vulnerability, the US military has been seeking alternatives to GPS [Hopson, 2010]. LORAN (Long-Range Aids to Navigation), is the only functioning backup to GPS, composed of a series of ground based transmission towers run by the US Coast Guard [Pappalardo, 2009]. However, LORAN is being shut down by the US government for obsolescence reasons, in favor of GPS. While the LORAN program has been terminated, the GPS program is also experiencing trouble. GPS relies on a constellation of satellites, some of which need replacement in the near future by the US Air Force. Unless the satellites are upgrades and replaced soon, GPS users face potential lack of GPS reliability to ground users. [GAO, 2009].

Some limited alternatives to GPS do exist. The Naval Research Advisory Council, identified a use for lighter than air systems as a potential navigation beacon [NRAC, 2005]. One such air system is an aerostat. Aerostats are lighter-than-air, large volume airships which are raised to high altitudes (below 10,000ft), but remain tethered to the ground. Aerostats have been used in the US wars in Iraq and Afghanistan for various surveillance purposes (see Fig. 30). At 6,000ft, the aerostat can cover approximately 300 nautical miles, providing a good range of coverage for maneuver forces [NRAC, 2005]. The US Border Patrol also uses aerostats as a part of the US Tethered Aerostat Radar

System for surveillance purposes near the US-Mexico border to combat drug smuggling via aircraft crossing into the United States [Tucson Sentinel, 2011].

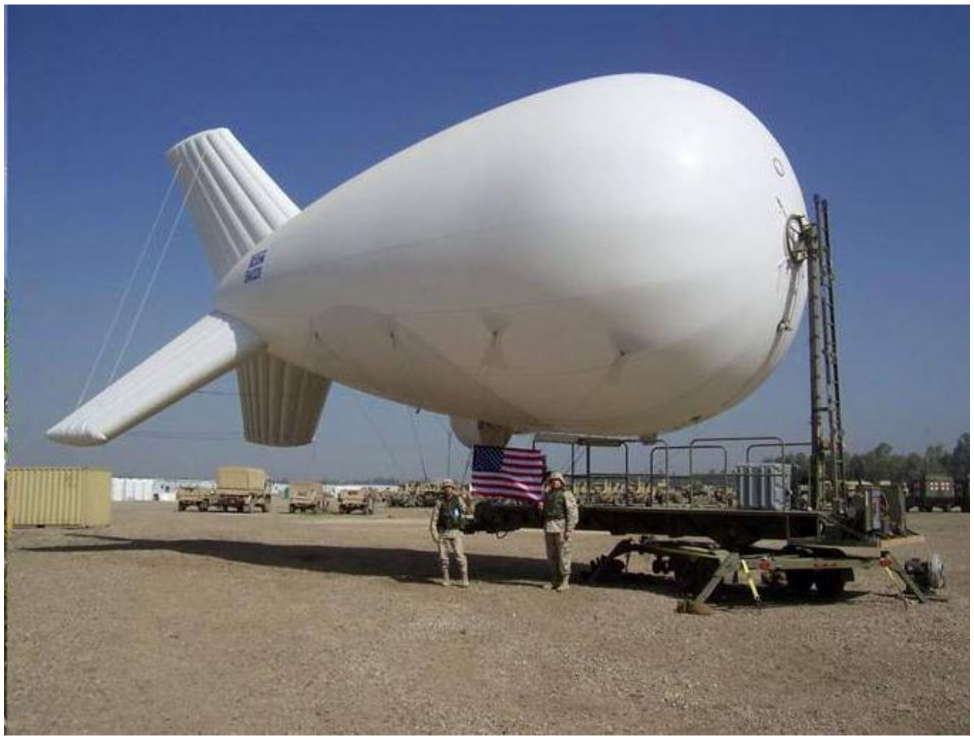


Fig. 30: US Military Aerostat Deployed in Iraq [NRAC, 2005]

This excursion will examine architectural changes and their effects on resilience by adding an aerostat backup to the GPS system in the targeting case study. In this excursion, the maneuver force is able to react to a disruption and loss of the GPS system by raising an aerostat with a navigational beacon. Once a disruption occurs, it takes 2 hours to get the aerostat into position on the ground, positioned aloft at 6,000ft. in altitude, and begin transmitting the navigation signal correctly. Therefore, t_{rc} = 2 hrs. However, an aerostat navigation system will have less accuracy and reliability than the GPS system. Additionally, there will be pockets of dead-space for signal coverage due to its oblique angles over intervening terrain, as compared to the nearly direct over-head coverage of GPS. Therefore, users will likely need to double check the aerostat based navigation signal results against their local terrain to ensure accuracy. This means that the capability will be improved, but not restored to GPS-level performance. Additionally, the aerostat is not a permanent capability. Its station keeping ability is limited from several days to approximately 2 weeks.

The modified BPMN architectural description of targeting case is shown in Fig.31. The aerostat is represented in Fig.31as a 'Backup Navigation Aid" and highlighted in yellow. The Aerostat provides navigation signals to the same users as a GPS signal.

As demonstrated earlier, the BPMN architecture may be directly mapped into Petri net form via the approaches identified in Raedts et al., [2007], Dijkman, Dumas, and Ouyang, [2008], Stahl, [2005], and Weske, [2010]. The addition of the aerostat modifies the Petri net architecture as shown in Fig. 32. The primary difference is the addition of the backup navigation signal shown at the very top of Fig. 32. To execute the Petri net in simulation, instrumentation places are added, which do not change the structural characteristics of the Petri net. Additional inscriptions are added to the arcs and guard functions to the transitions to account for the delay in availability of the aerostat as well as the improvement (but not complete return to GPS-level) in performance when the aerostat beacon is functioning.

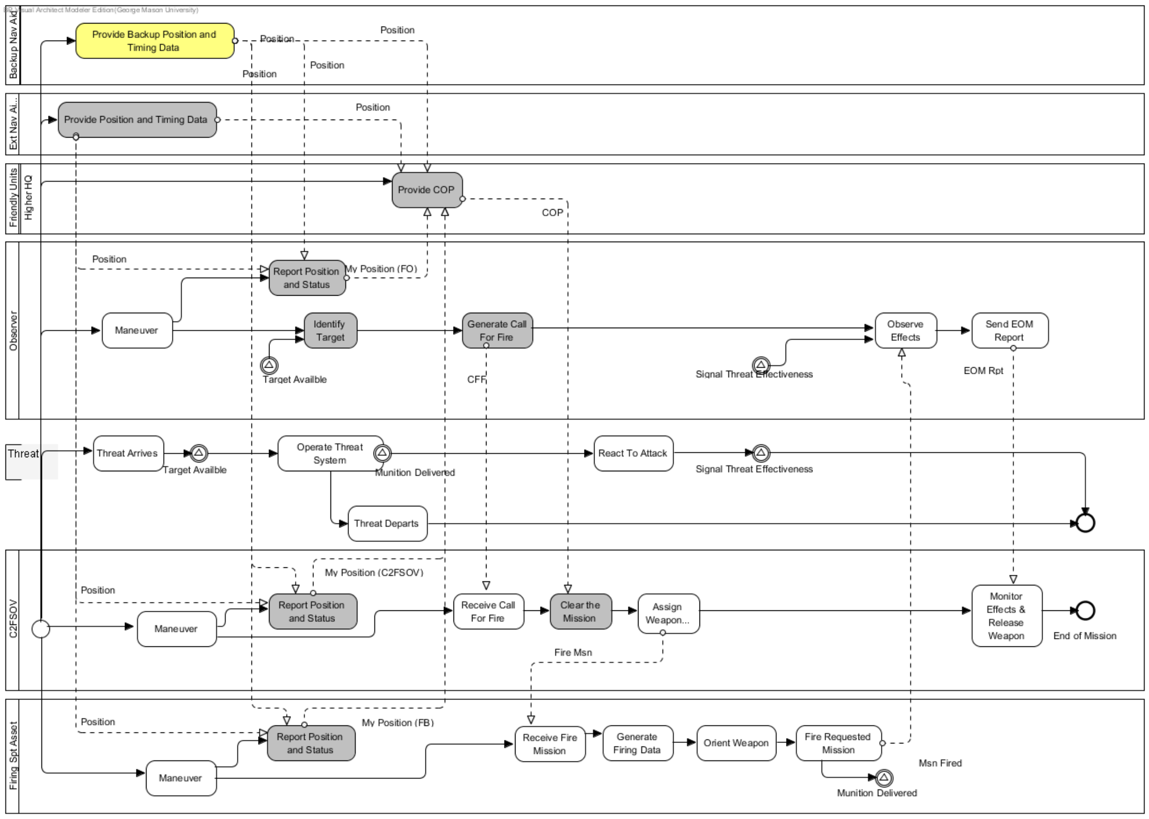


Fig. 31: Modified BPMN Architectural Description of Targeting Case with Aerostat

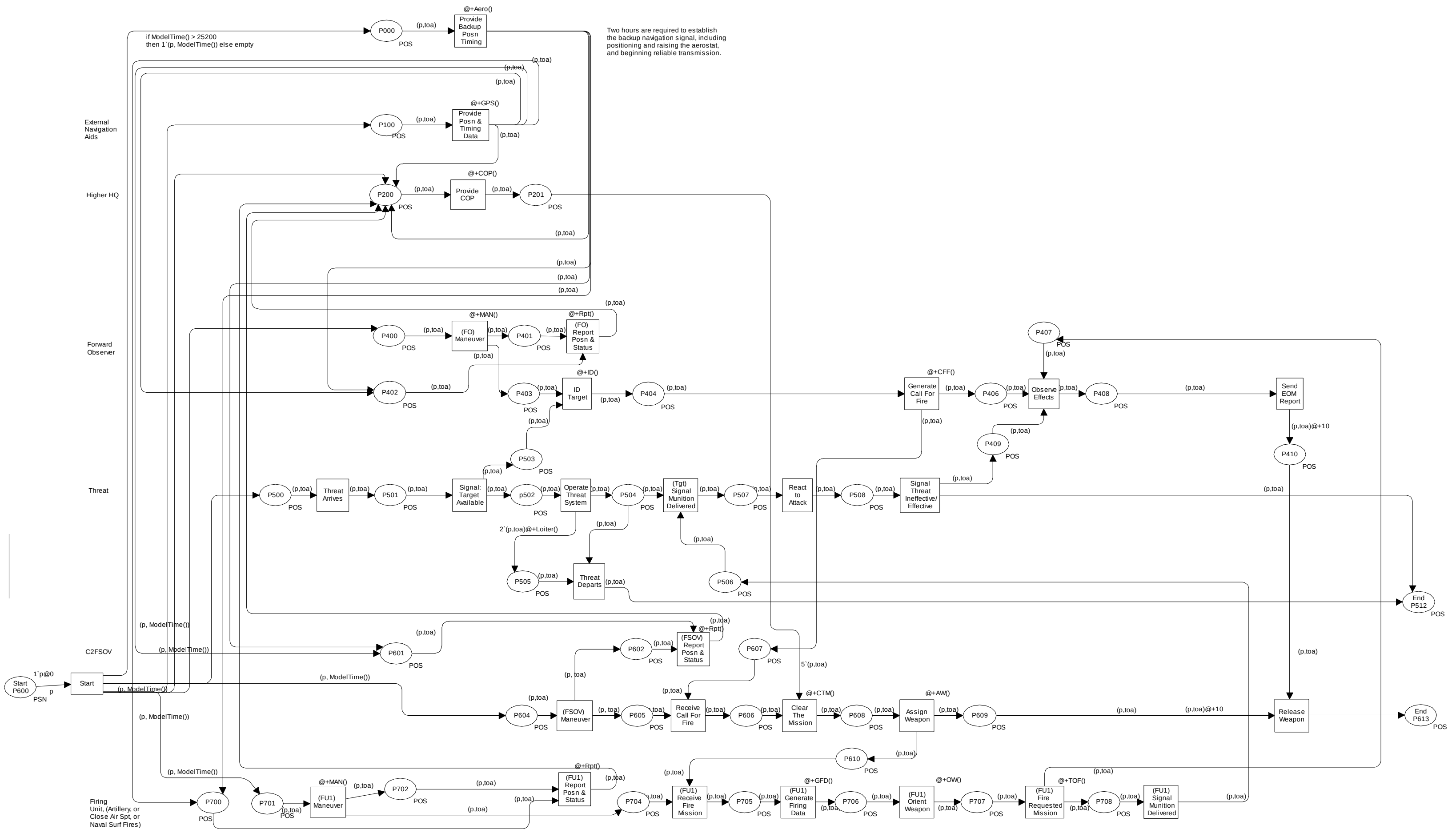


Fig. 32: Modified Petri Net Architecture with Aerostat Backup Navigation Signal

5.1.4.1. The Aerostat Excursion and Capacity

As stated, the primary advantage of the aerostat is as a limited backup capacity (i.e., reactive capacity) for a disruption to the GPS network, which is so important to the fire support capability. Figure 33 illustrates the effect on capacity when the aerostat is present. Prior to the disruption, at t_d = 5 hours, the system functions normally with an average remaining window of 559 seconds. As in the original targeting case, once the GPS network is disrupted and unavailable, performance drops off dramatically to an average remaining window of 56 seconds, as elements in the fire support system revert to manual means with much longer processing times. In certain cases, the system essentially fails, as targets depart prior to being engaged. These departed targets, i.e., missed opportunities, are shown as circles on Fig. 33.

After a reaction time of 2 hours, the aerostat is available and begins to operate as a backup to the GPS. The performance of the aerostat enabled fire support team, as described in the architecture, does not return to pre-disruption levels, but does improve to just above the threshold level of performance. With the aerostat in operation, the average remaining window is 401 seconds. The threshold level remains as previously stated as 390 seconds, based on subject matter expert judgment during the data collection efforts of this research.

However, here we have a special case of reactive capacity. In certain cases, reactive capacity is additive, as we will see in the next case study. In this case, however, the aerostat and GPS navigation signals are not truly additive. Rather, the aerostat provides surrogate capability. For example, a user really only needs either signal. While GPS is more accurate and reliable, the addition of an aerostat signal does not provide any added value. In reality, the aerostat is a surrogate for, but not additive to, the GPS-based capacity.

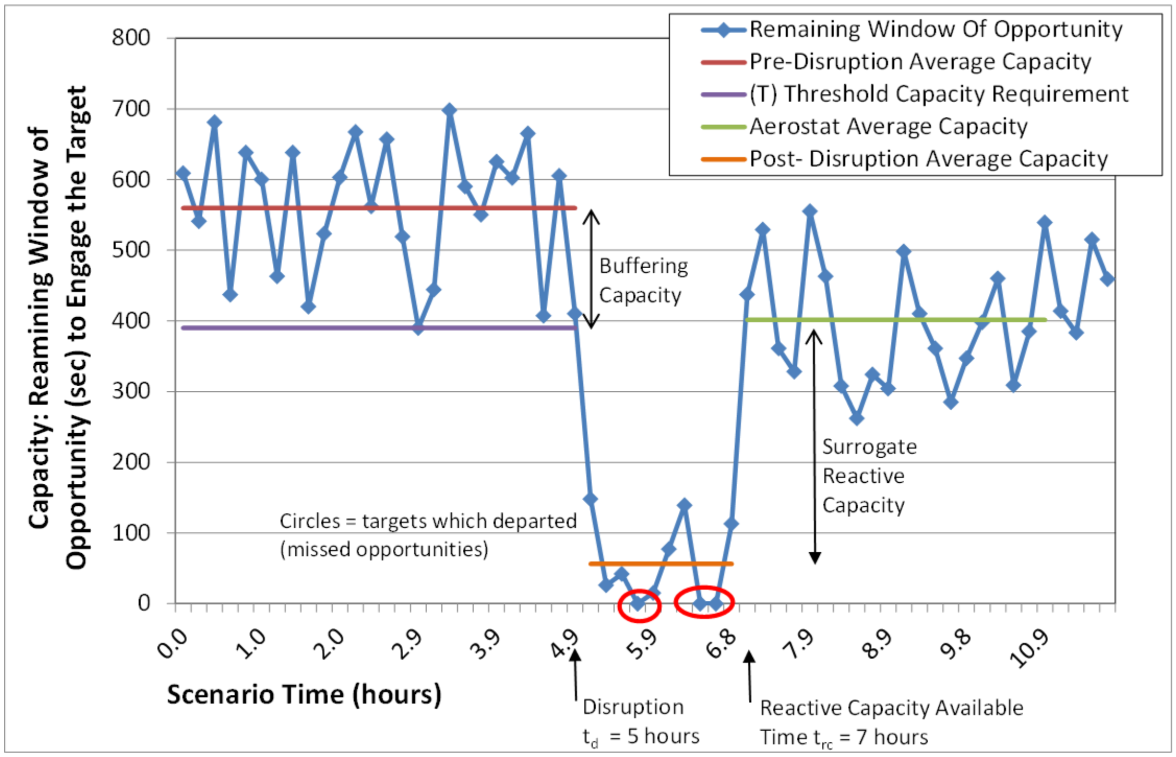


Fig. 33: Reactive Capacity of the Targeting Architecture with Aerostat

In Chapter 3, Fig. 2, the following equation was introduced for surrogate reactive capacity. This equation is solved using the results of the aerostat excursion.

$$SurrogateReactiveCapacity = \frac{V_{Trc} - V_{Tmin}}{V_{Trc}} = \frac{401 - 56}{401} = 86\%$$

The capacity results found in the base targeting architecture otherwise still apply. The aerostat excursion only changes the reactive capacity of the targeting architecture.

5.1.4.2. The Aerostat Excursion: Tolerance and Flexibility

The aerostat excursion's results for tolerance do not significantly vary from those found in the base case. In terms of graceful degradation, the additional surrogate reactive capacity is not available before the disruption incurs its full effect. Therefore, the results for graceful degradation are the same for both the base and aerostat cases. The revised architecture retains the same number of simple information flow paths(39), and does not alter the number or location of cut vertices. Therefore, the results for fault tolerance remain unchanged. Additionally, the association of elements to information flow paths also remains essentially unchanged, so the results for point of failure tolerance are also not significantly different.

5.1.4.3. The Aerostat Excursion: Flexibility

Proportion of Use was selected in the base targeting architecture as the best measure of flexibility. Figure 34 shows the results for Proportion of Use. As expected, the addition of the reactive capacity for the aerostat excursion does lower proportion of use (i.e. increased flexibility), however, this is only by a negligible amount.

Information Flow Path	# Elements Contained by ℓ_i	Information Flow Path	# Elements Contained by ℓ_i
ℓ = 1	15	ℓ = 21	33
ℓ = 2	15	ℓ = 22	33
ℓ = 3	15	ℓ = 23	35
ℓ = 4	17	ℓ = 24	33
ℓ = 5	17	ℓ = 25	35
ℓ = 6	19	ℓ = 26	35
ℓ = 7	13	ℓ = 27	37
ℓ = 8	20	ℓ = 28	37
ℓ = 9	22	ℓ = 29	39
ℓ = 10	22	ℓ = 30	33
ℓ = 11	24	ℓ = 31	27
ℓ = 12	22	ℓ = 32	39
ℓ = 13	24	ℓ = 33	41
ℓ = 14	24	ℓ = 34	41
ℓ = 15	26	ℓ = 35	43
ℓ = 16	29	ℓ = 36	41
ℓ = 17	31	ℓ = 37	43
ℓ = 18	25	ℓ = 38	43
ℓ = 19	19	ℓ = 39	45
ℓ = 20	31	r = 39	∑ Bi = 1143 E = 73

$$\mathrm{PoU} = \frac{\frac{\sum_{i=1}^{r} B_i}{E}}{r} = \frac{\sum_{i=1}^{r} B_i}{rE}$$

Base Targeting Case:

$$= \frac{1119}{71*39} = \frac{1119}{2769} = 40.4\%$$

Aerostat Excursion of the Targeting Case:

$$= \frac{1143}{73*39} = \frac{1143}{2847} = 40.1\%$$

Fig. 34: Flexibility (Proportion of Use) in the Aerostat Excursion

5.1.4.4. The Aerostat Excursion: Overall Results

As discussed, the primary difference for the aerostat excursion is the addition of reactive capacity. This excursion demonstrated one type of reactive capacity: surrogate reactive capacity. In the surrogate case, the reactive capacity cannot be combined with existing capacity. Other than a slight improvement in flexibility, as measured by Proportion of Use, the aerostat excursion's tolerance and flexibility do not significantly change from the base case.

We can now map performance against an overlaid requirements locus to evaluate resilience. Determining the resilience requirements locus requires value judgments by the key stakeholders responsible for the architecture's development. In aerostat case, the requirements locus remains the same as specified earlier in Table 5, with the exception of measuring reactive capacity, versus buffering capacity. Figure 35 shows the resulting evaluation. In the base case, the targeting architecture is deficient with regard to reactive capacity, but meets the requirements for flexibility and tolerance. The aerostat excursion resolves this deficiency in terms of reactive capacity, bringing the performance of the architecture into the required performance levels with respect to resilience.

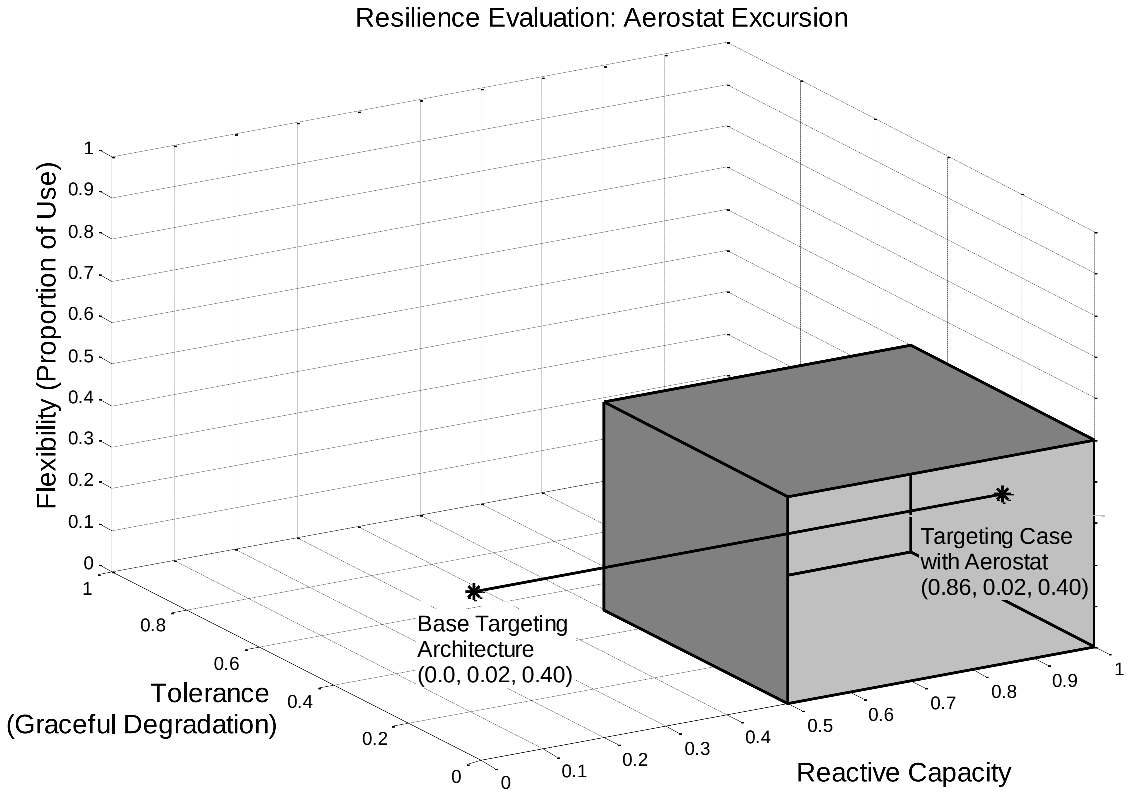


Fig. 35: Resilience Evaluation of the Aerostat Excursion

5.2. DECISION MAKING ORGANIZATION CASE STUDY

5.2.1. Background

This case study involves a new organization called the Maritime Operations Center (MOC). As the United States' presence and engagement continues on a global scale, the US Navy is transitioning portions of its command and control organizations to a MOC structure. A MOC is a large, distributed organization at the fleet level, with command and control responsibilities to "manage [routine] operations and be able to smoothly transition from peacetime operations to disaster relief operations and major combat operations, while still handling fleet management functions" [US Navy Public Affairs, 2009]. The MOC is organized beneath a Joint Force Maritime Component Command (JFMCC). The MOC receives orders from the JFMCC, conducts planning operations, and generates Operations Orders (OPORD) for execution by the units assigned to the MOC. Figure 36 shows a picture of ships from the US Navy 4th Fleet undergoing training exercises during MOC certification accreditation [US Navy Public Affairs, 2009]. Information regarding the MOC used in this case study is based on separate GMU Systems Architecture Laboratory (SAL) work for the Office of Naval Research (ONR) under contract number (N00014-08-1-0319). This case study used a baseline and augmented MOC model constructed by SAL staff as a foundation, made several modifications, and then applied the approach described in this research.

Fig. 36: US 4th Fleet MOC, International Exercise PANAMAX 2008

The MOC in this case study involves six major Decision Making (DM) organizations: Assessment, Operational Intelligence, Future Plans, Command, Current Plans, and Current Operations. These organizations work in concert to conduct command and control of Naval and Joint forces on the surface, below the surface, in the airspace and ashore.

Like many human organizations, augmentation is a typical strategy for dealing with crises and uncertainty in work load. This case study will compare two different candidate architectures for the MOC: a baseline MOC and an Augmented MOC, where the Augmented MOC adds additional nodes for Operational Intelligence and Future Plans, such that cross talk exists between nodes. These additional nodes, once called, require time to establish and are available after a given reaction time.

A primary capability of the MOC is to generate mission orders for subordinate unit execution, based on incoming JFMCC orders (higher HQ). The appropriate Measures of Performance (MoP) in this case is the mission orders generation rate, stated as number of mission orders generated per 24 hours, and the Average System Time from when an order from higher headquarters is received, to the time at which it is disseminated to subordinate units as an OPORD as a rate per 24 hours. Put another way, if an order takes 4 hours to process, the mission order generation rate is 6 orders per 24 hours.

Orders arrive at the MOC from the JFMCC approximately every 3.5 to 4 hours, with an execution time of 24 hours later. If the MOC spends more than 8 hours to generate mission orders for their subordinate units, then the subordinate units do not have sufficient time to conduct their own planning, move into position, and execute the mission. This is essentially an extension of the traditional 1/3:2/3 planning rule, where higher units do not take more than 1/3 of available time to ensure lower units can successfully execute the mission. Therefore, if the MOC takes longer than approximately 8 hours to generate mission orders (i.e., falls below a mission order generate rate of 3 per 24 hours), the mission is put in jeopardy because subordinate units may not be able to execute in time.

Like most operations centers, the MOC is dependent upon software to automate and improve its functioning. In this case study, we are examining the resilience of the

MOC's capability to 'Generate Mission Orders' to the disruption 'loss of situational awareness software.' When a new release was received, the update caused both versions to crash, and attempts to restart were unsuccessful.

Loss of this software affects the Information Fusion stage of each decision making organization, extending the process time associated with that step. Each DM organization can still complete the Generate Mission Order process, but the process transitions to a manual backup, and requires a longer time to complete. In this case, the manual process takes 3 to 5 times as long as the software supported information fusion process. The software failure occurs at t_d = 48 hours. 24 hours are required to bring additional (augmented) capacity on-line; therefore, t_{rc} = 24 hours.

5.2.2. Architecture

An organizational architecture, a potential design for the MOC, is depicted in Fig. 37 (Base MOC) and Fig. 38 (Augmented MOC). Note that in the Augmented MOC an additional Operational Intelligence and additional Future Plans cells are added, with cross talk to the original cells. These figures are generated in CAESAR III. Each Decision Making (DM) organization is shown as a modified rectangle; the arcs represent fixed and variable connections (interactions) between decision making organizations by which information (or signals) is passed. Fixed connections between decision nodes indicate interactions which do not vary, whereas variable connections may change between situations.

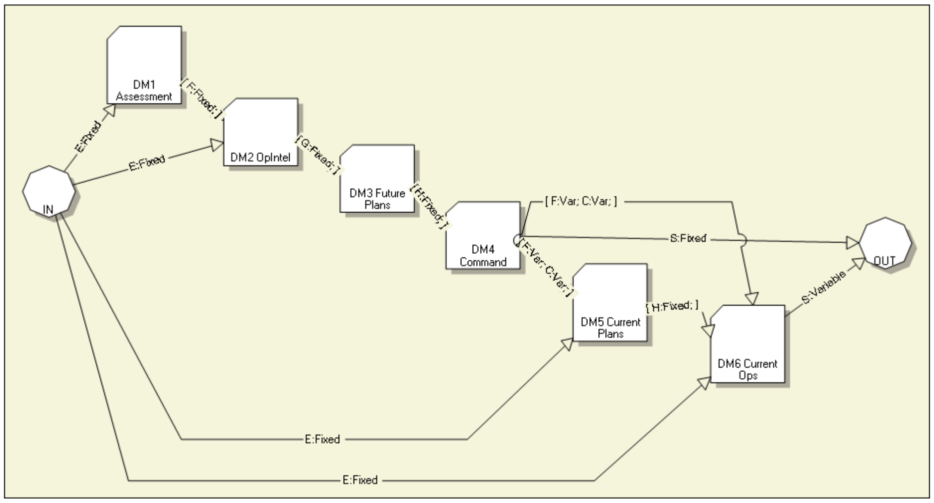


Fig. 37: The Base MOC Organizational Design

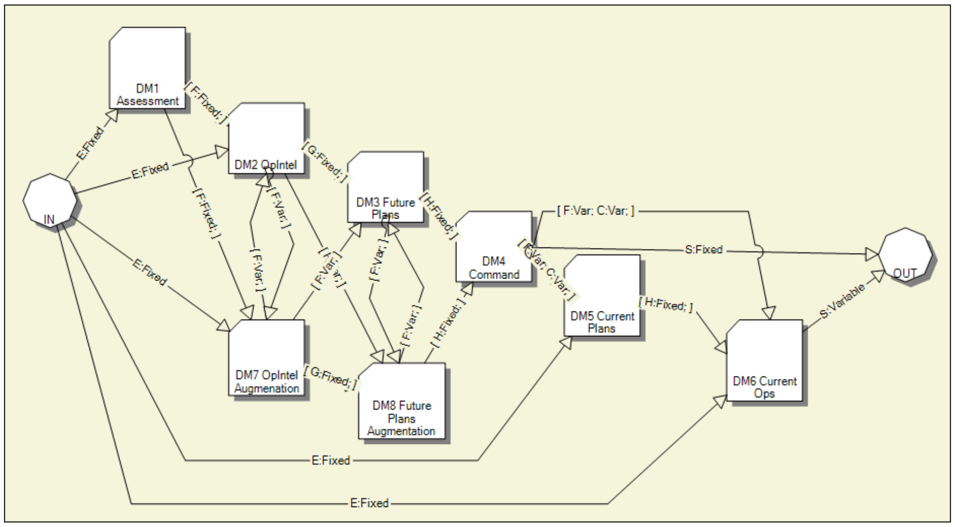


Fig. 38: The Augmented MOC Organizational Design

Remy and Levis [1986] introduced a four stage (later expanded to a five stage) interacting decision maker model based upon Petri Net Theory and the lattice algorithm.

Each DM organization is modeled using the five stage decision maker model, therefore each DM organization shown as a rectangle in Figs. 37 and 38 can be mathematically described using a Petri net with interactions defined in Remy and Levis (1986). See Fig.39 from Kansal et al. [2007].

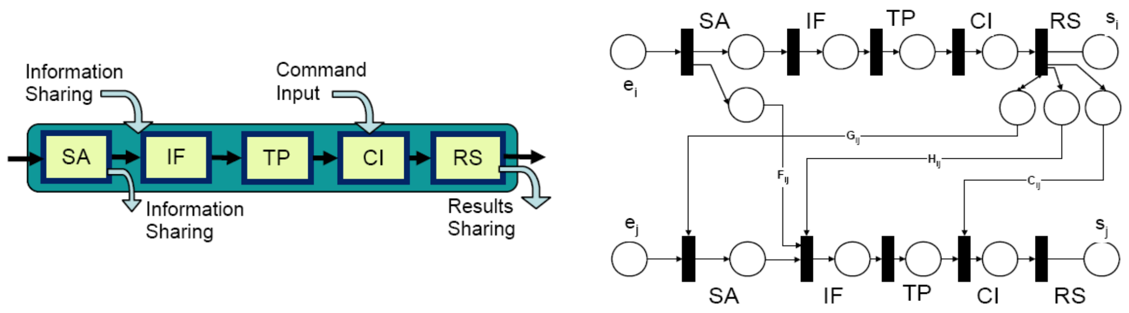


Fig. 39: Five Stage Model of Each DM Node

The individual DM nodes receive either a signal from the external environment or from another DM node. "The Situation Assessment (SA) stage represents the processing of the incoming signal to obtain the assessed situation that may be shared with other DMs. The decision maker can also receive situation assessment signals from other decision makers within the organization; these signals are then fused together in the Information Fusion (IF) stage to produce the fused situation assessment. The fused information is then processed at the Task Processing (TP) stage to produce a signal that contains the task information necessary to select a response. Command input from superiors is also received. The Command Interpretation (CI) stage then combines internal and external guidance to produce the input to the Response Selection (RS) stage. The RS stage then produces the output to the environment or to other organization members." [Kansal et al., 2007].Using the theory outlined in Remy and Levis [1986], CAESAR III uses the Lattice algorithm to generate feasible solutions that represent all possible architectures that meet a set of defined constraints. These solutions are represented as Ordinary Petri Nets. Figure 40 is a Petri Net representation of the DM organization shown in Fig. 38.

In this case, the primary output of the Generate Mission Orders capability is a mission order. The places P53 and P55 shown in Fig. 40 are the primary components of that mission order, where T5 is the transmission of that mission order to subordinate units. For example, the primary output of the future ops cell is an OPORD, corresponding to P53. The primary output of the current ops cell is a Fragmentary Order (FRAGORD) situation report and a synchronization matrix, corresponding to P55. A subordinate unit will execute the mission when either or both components are present, however, it will not execute if neither is present.

Using the CAESAR III generated Petri net, we can next add further necessary logic to the net and instrument it to support simulation. Care is taken to ensure that changes do not affect the overall structural properties of the original net (for example to change the nature of the information flow paths). Time was added to the original Petri Net and appropriate stochastic delays estimated for each step to represent the amount of time required for each task. The arcs were inscribed to ensure a single incoming mission order from a higher headquarters is matched up correctly as different organizations within the

MOC perform their roles (i.e. when the OPORD is approved in the Command cell, that it matches the Synch Matrix and FRAGORD from the Current Operations cell.) The resulting Petri net is shown in Fig. 41.

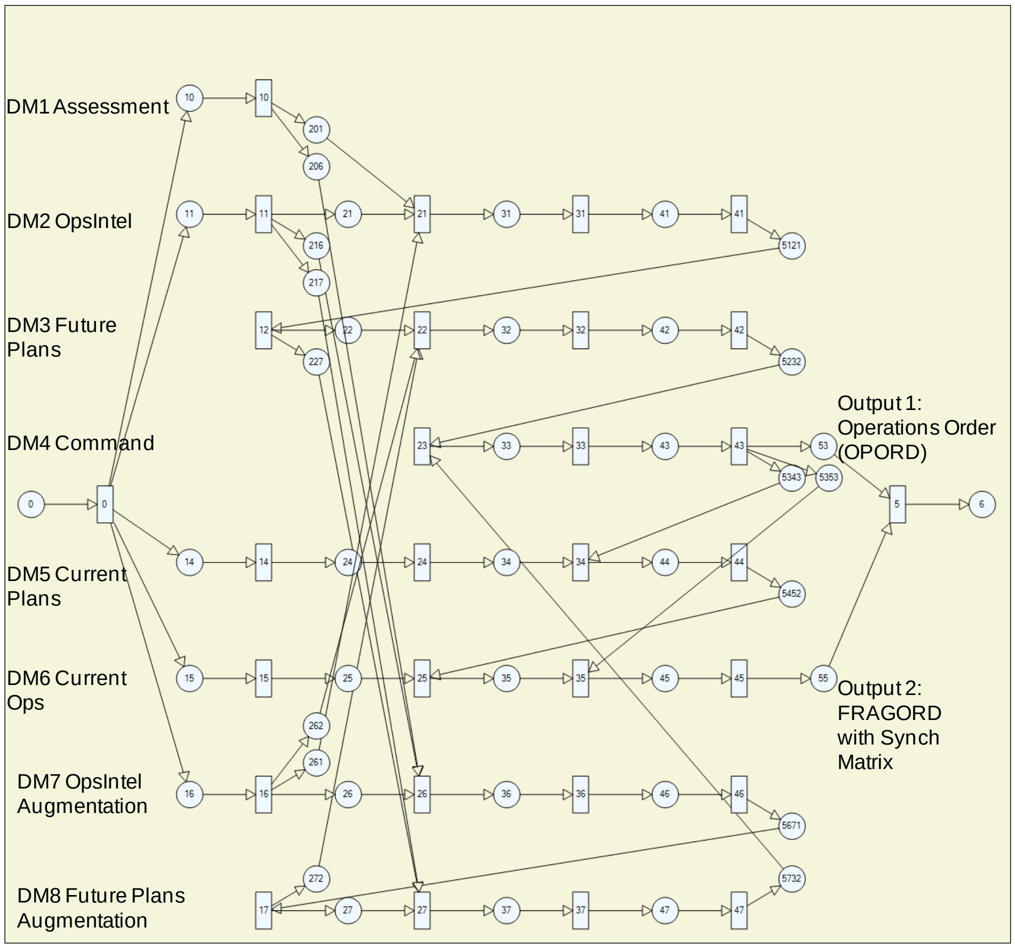

DM1 Assessment
DM2 OpsIntel
DM3 Future Plans
DM4 Command
DM5 Current Plans
DM6 Current Ops
DM7 OpsIntel Augmentation
DM8 Future Plans Augmentation
Output 1: Operations Order (OPORD)
Output 2: FRAGORD with Synch Matrix

Fig. 40: The Augmented MOC Universal Net in Petri Net Form

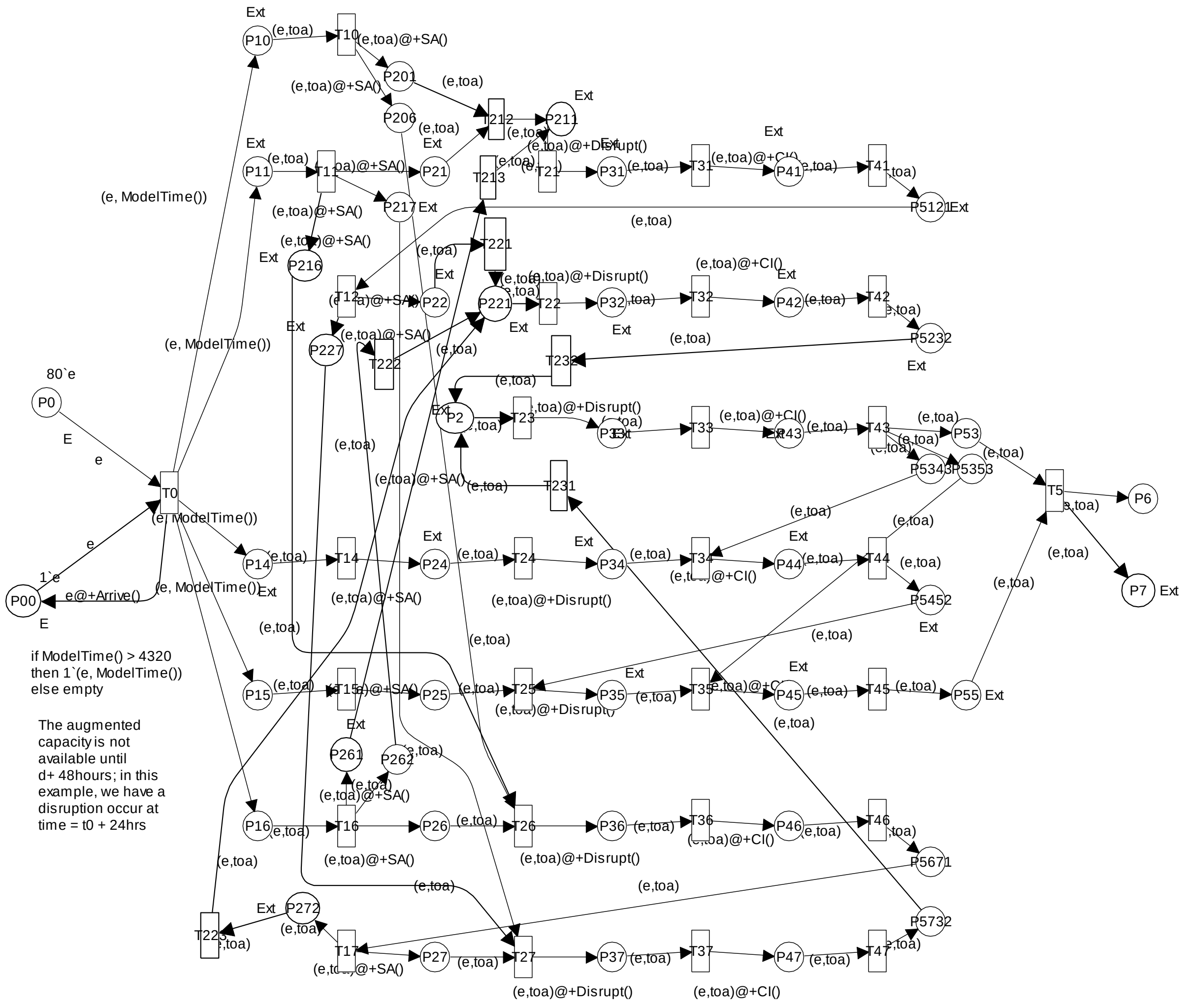

if ModelTime() > 4320
then 1`(e, ModelTime())
else empty
The augmented capacity is not available until d+ 48hours; in this example, we have a disruption occur at time = t0 + 24hrs

Fig. 41: Petri net for the Augmented MOC Used in Simulation

5.2.3. Decision Making Organization Case Study Results

As in the targeting case study, once the architecture is developed, sufficiently verified, and any errors / revisions are addressed, it may be used to support the analyses described in Chapter 3. In demonstrating the approach developed in this research, all measures of capacity, tolerance, and flexibility are calculated. However, an architect with the overall development team could in principle investigate only those measures of special interest, as guided by Figure 14 and Table 1 in Chapter 3. More generally, it is not necessary to calculate all measures if the architect and development team know apriori which measures are of greatest interest.

5.2.3.1. Decision Making Organization Case Study Results: Capacity

Returning to our method for calculating capacity, we can use the simulation results to calculate measures for buffering, reactive, and residual capacity. The MOC operates under normal conditions between times t_0 and t_{48}. At t_{48} ($t_{48} = t_d$), a disruption occurs, in this case the failure of the information fusion software. The time to execute the information fusion step in the MOC Mission Order Generation capability increases as the MOC staff switches from the automated software-based approach to a manual approach. At time t_d, augmented capacity is requested, however, it takes 24 hours to stand up this augmented capacity and integrate it into the existing MOC command and control structure. Therefore, $t_{rc} = 24$. The augmented capacity comes in the form of an additional Intelligence Cell, and an additional Future Plans cell. Essentially, the MOC is augmenting with additional manpower to retain its capability to generate mission orders.

Figure 42 reflects the architecture's modeled simulation results during the course of the scenario. The MoP for the Generate Mission Orders MoE is shown on the vertical axis as Mission Order Generate Rate (Orders/24hrs). The time is shown on the horizontal axis. Starting at time t_0, the MOC performs under normal, pre-disruption performance levels with respect to the capability Generate Mission Orders. At this point, the MOC is capable of generating mission orders in approximately just over 4 hours. From the model results, this translates into an average of 5.67 mission orders every 24 hours. The situational awareness (information fusion) software fails at time t_{48}, and the mission order generation rate falls off dramatically as the MOC switches to manual backup procedures. At the minimum point of performance ($t_{min} = t_{53}$) , the MOC is barely at the threshold level of performance of approximately 8 hours to generate a mission order, or 3 mission orders per 24 hours. By time t_{72}, additional capacity (manpower) has been integrated to stand up an augmented future plans cell and augmented operational intelligence cell. These additional cells are able to restore a part, but not all of the original capability in terms of the mission order generation rate MOP.

In Fig. 42, the red line denotes the threshold capacity, set in this scenario as 3 mission orders generated every 24 hours, as described earlier. The green line indicates the maximum performance the MOC could achieve with respect to this capability if the augmented capacity were in place, but no disruption had occurred. This was calculated by simulating the architecture with the augmented capacity in place, but without the effects of the disruption. Comparing the Mission Order Generation Rate to the threshold value establishes the MOE for this measure.

The primary difference between the two alternative architectures under examination in this case study is that the Augmented MOC is able to generate reactive (spare) capacity, and the Base MOC is not. Maximum capacity when augmentation is available was determined by running the simulation model without the effects of the disruption, and with the spare capacity in place. This was completed by slight modifications to the inscriptions on the arcs in the Petri net shown in Fig. 41.

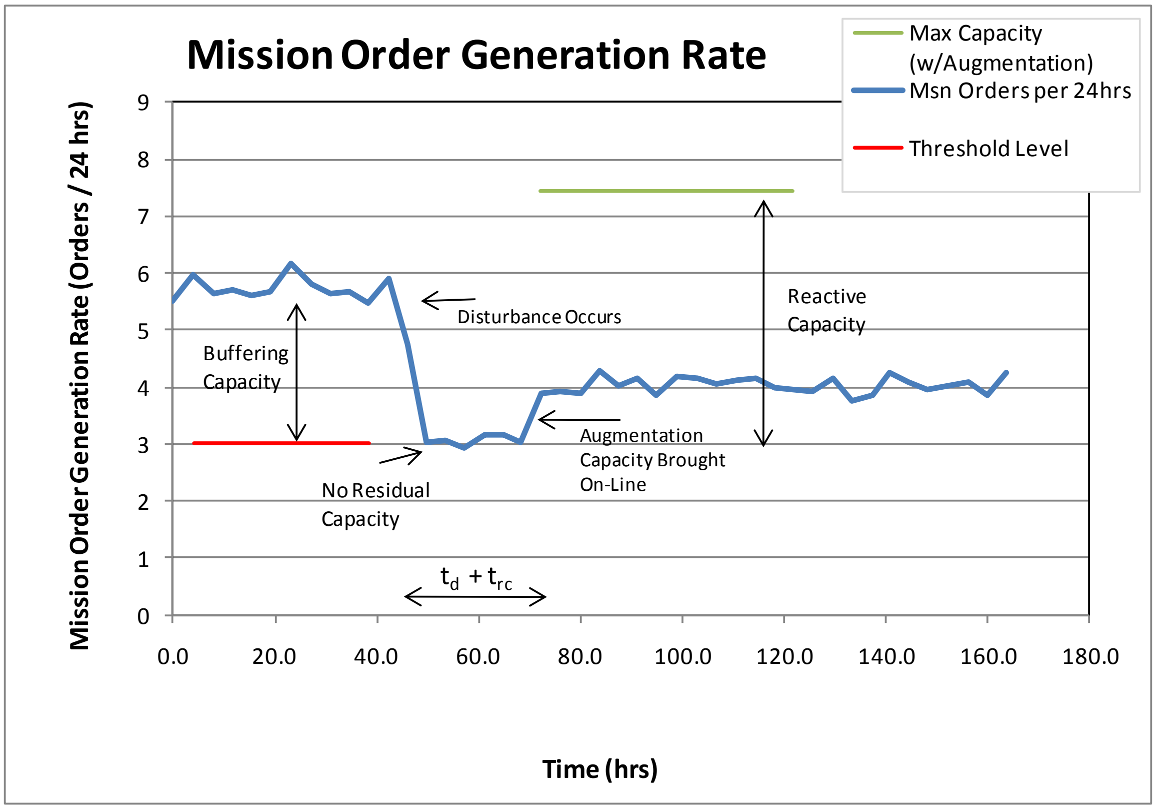


Fig. 42: Measuring Capacity in the Augmented MOC

Using the equations for buffering, reactive and residual capacities (see Fig. 2) we find the results shown in Table 6. When operating under pre-disrupted conditions, approximately half (47%) of the MOC’s capability was above the required threshold of 3 mission orders per 24 hours. During the survival phase (post disruption), the MOC was operating at the threshold of 3 orders per 24 hours. However, as the calculations indicate, no residual capacity exists, meaning that any further disruption could have resulted in catastrophic failure in terms of mission completion. The MOC was operating close to an edge in performance. Additional manpower assisted in raising MOC performance above the threshold, but did not return it to pre-disruption levels. The simulation results indicate that only restoration of the failed situational awareness software would return the MOC to pre-disruption performance levels. If the reactive capacity (the augmentation cells) were in place with no disruption, then 60% of the MOCs capacity would be above threshold.

Table 6: Determining Capacity in the MOC

Max Capacity (w/Augmentation)	7.44	V_{max}
Threshold Level	3.00	V_T
Normal Opn Level	5.67	V_2
Buffering Capacity	**47%**	
Reactive Capacity	**60%**	
Residual Capacity	**0%**	V_1

5.2.3.2. Decision Making Organization Case Study Results: Tolerance

As with the capacity related metrics, the rate of departure metric is determined by employing an executable model of the architecture to assess performance achieved against performance required. Recall from earlier discussion that we defined Rate of

Departure (Tol_{RD})as the rate of change over time in system effectiveness in meeting its requirements after a disruption occurs. This encapsulates both the temporal aspects of resilience (t_d and t_{min}), as well as the effectiveness aspects of how the system performs with respect to its requirements and how effectiveness changes during the survival phase (post disruption).

A parameter locus is generated to account for how key parameters affecting performance may vary during the scenario. The mission order inter-arrival time is an important parameter because it represents how quickly mission orders arrive from the JFMCC. Inter-Arrival time of orders from higher HQ (JFMCC) is varied to examine the effect of queuing as the MOC executes the Mission Order process based on those JFMCC orders. The disruption involved loss of the situational awareness software supporting the information fusion stage of the MOC. Since this drives the nodes within the MOC to use manual means, the time required for the Information Fusion tasks performed is varied to reflect various manual task durations. These two variables are included in the parameter locus, shown in Fig. 43.

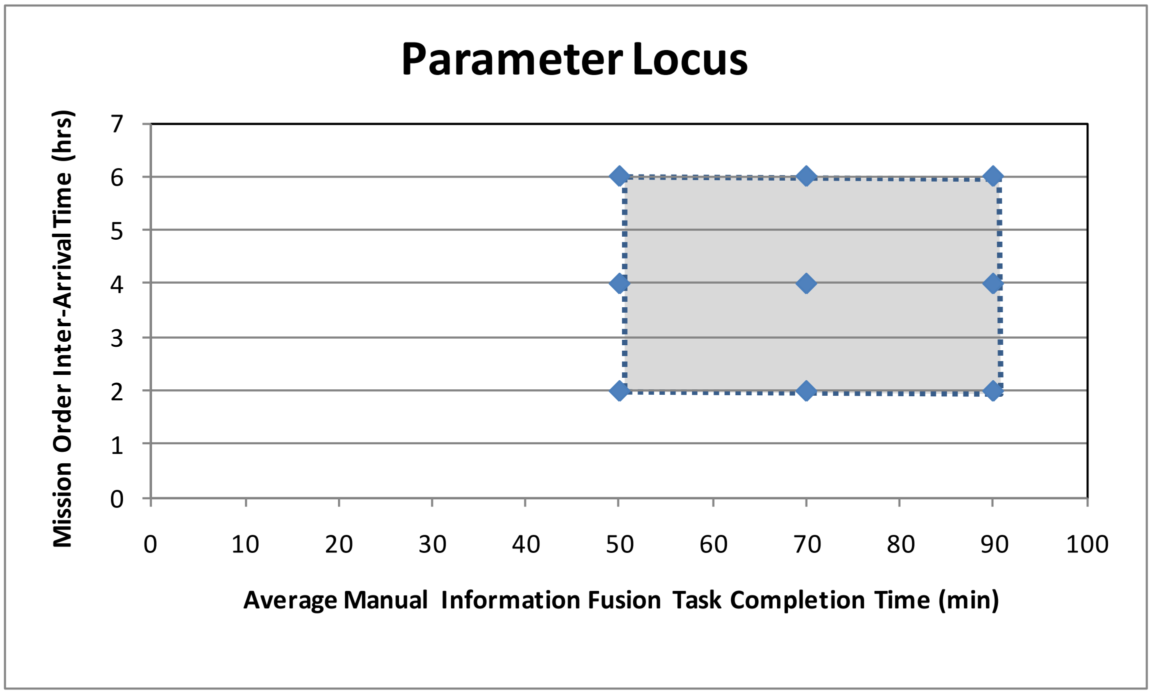


Fig. 43: Parameter Locus for the MOC

As in the targeting case study, the requirements locus is determined based on the specific variables of interest to the system under study. In the case of the MOC, the average mission order generation rate and the percent of orders delivered late to subordinate units are important. Figure 44 shows the requirements locus.

- Average Mission Order Generation Rate: number of mission orders generated per 24 hours. Per the 1/3 : 2/3 planning rule, higher units do not take more than 1/3 of available time to ensure lower units can successfully execute the mission. If the MOC takes longer than approximately 8 hours to generate mission orders (3 per 24 hours), subordinate units may not be able to execute in time.

- % Orders Delivered Late: Percentage of Mission Orders delivered to subordinates more than 8 hours after receipt at MOC, out of the total in the first 48 hours following the disruption. This addresses the effect on subordinate units. A threshold of 1 in 4 (25%) is established for this requirement.

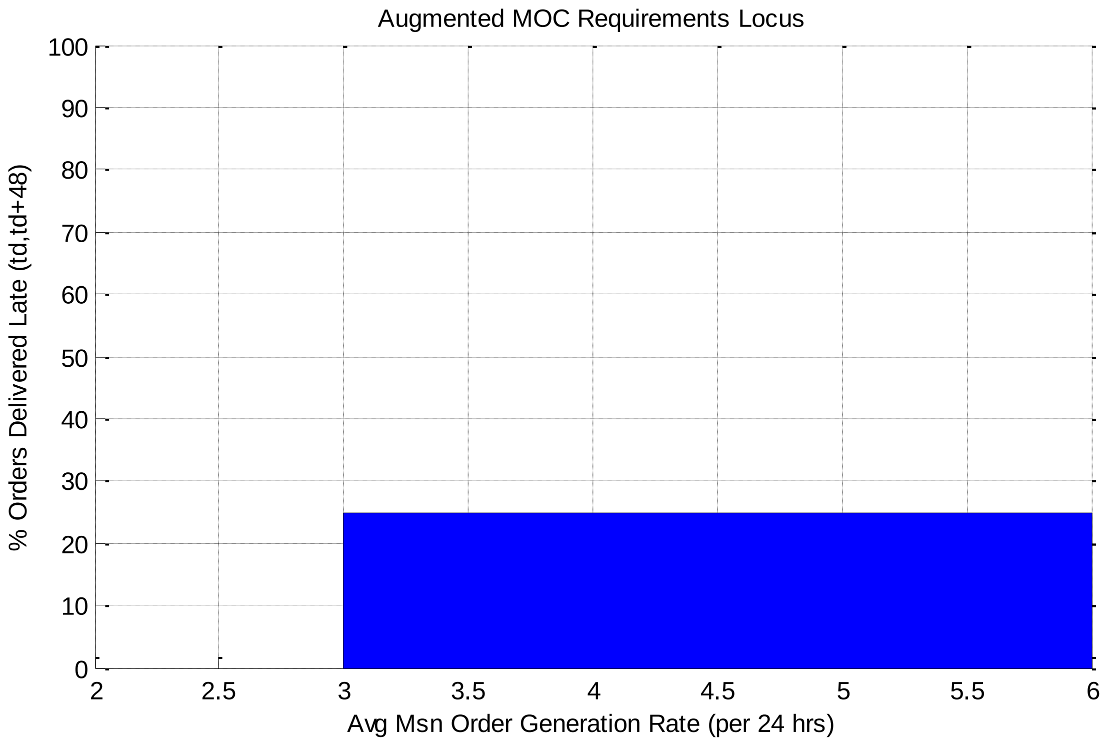


Fig. 44: Augmented MOC Requirements Locus

Executing the architecture at each point in the parameter locus (Fig. 43) yields a locus of performance. Figure 45 displays pre-disruption performance where data is collected before the disruption occurs. In the Augmented MOC, Mission Order Generation times are well within requirements, and zero orders are delivered late to subordinate units in any portion of the parameter space.

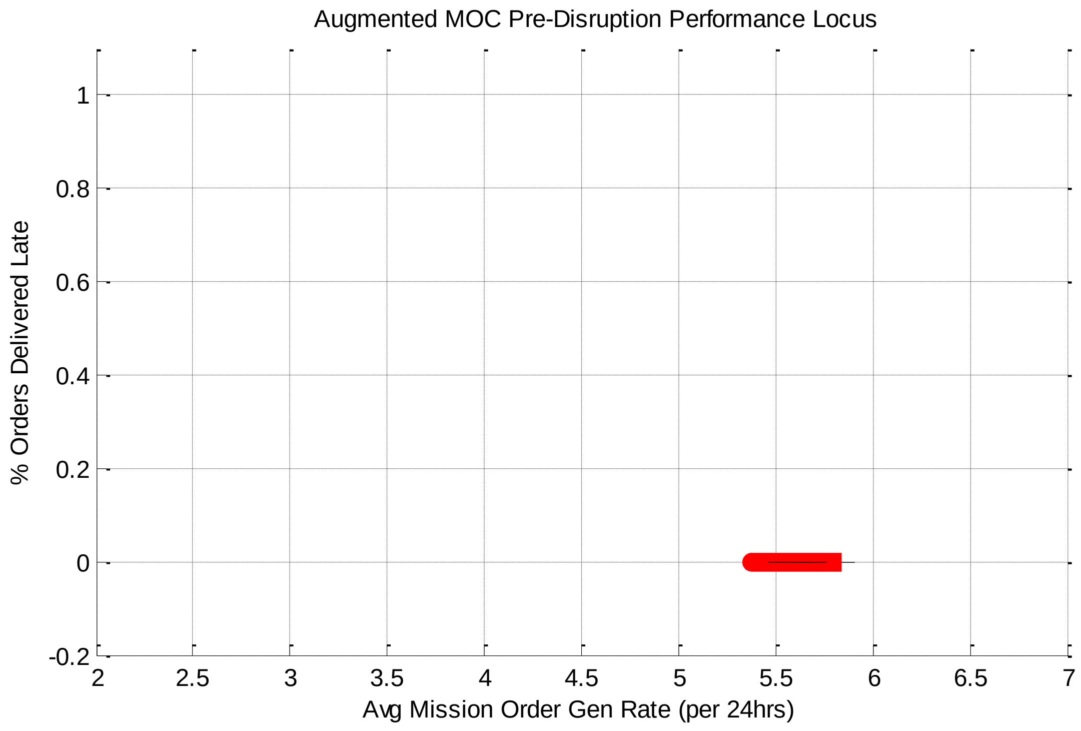


Fig. 45: Augmented MOC Pre-Disruption Performance Locus

Executing the architecture again at each point in the parameter locus, but *after* a disruption, yields a second locus of performance. Figure 46 displays post disruption performance where data is collected during the survival phase after the disruption occurs. After the disruption, the mission order generation rate slowed as the Information Fusion process required more and more time. For certain cases, up to half of the orders in the 48 hours following the disruption were delivered late. While augmented capacity is available in the Augmented MOC, it is not available until after the augmentation cells are established, approximately 48 hours after being called for. Mission order generation is highly dependent on software to enable tasks. Loss of situational awareness software causes a reversion to manual Information Fusion methods with much longer processing times. These problems are reflected in the degraded performance seen post disruption.

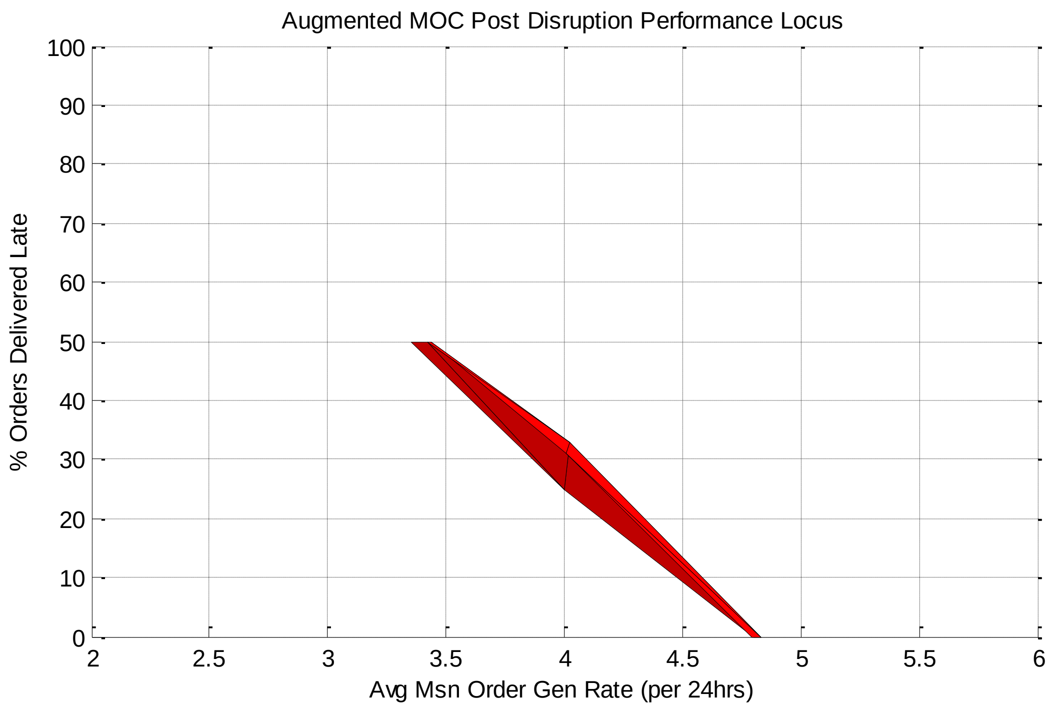


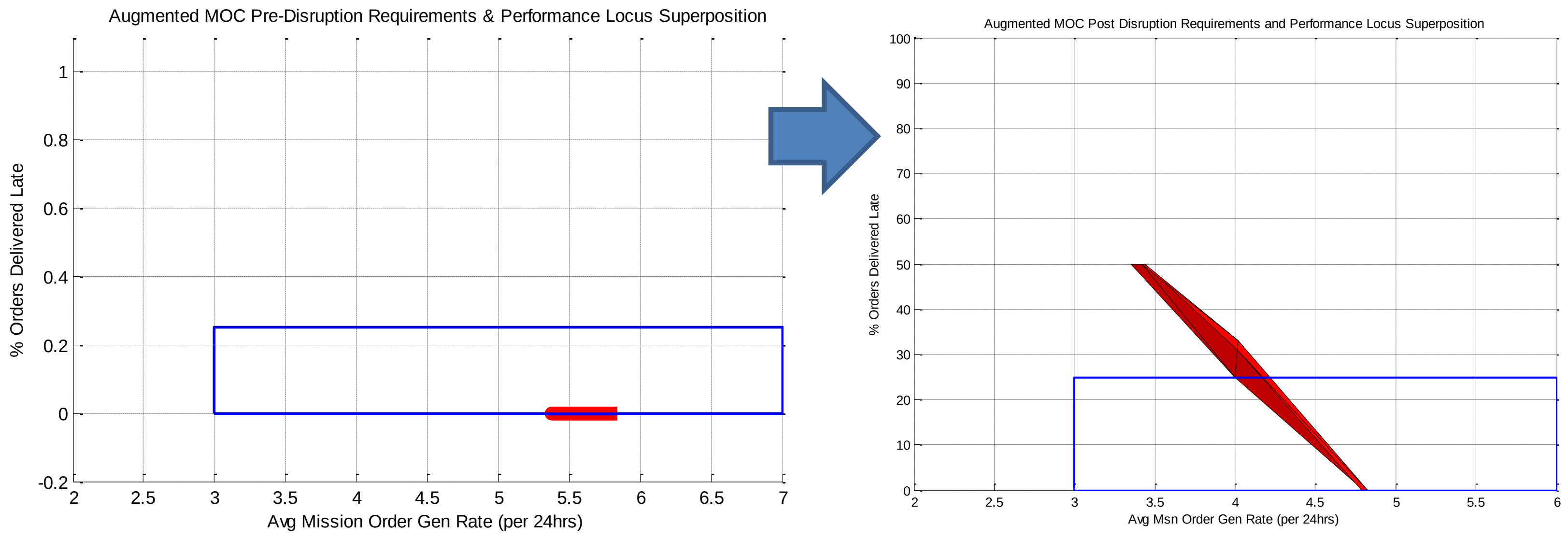
Fig. 46: Augmented MOC Post-Disruption Performance Locus

Prior to the disruption, we can see that the Augmented MOC is very effective at the Mission Order Generation Process. Zero orders are delivered late to subordinate units within the parameter space, and the Order Generation Rate is well within the required level of effectiveness. Prior to time t_d, the performance of the system meets the requirements over 100% of the parameter space (see Fig. 47A).After the disruption occurs, the system performance meets the requirements in only 70% of the parameter space, showing a significant loss of capability after the disruption (see Fig. 47B). The MOC Decision Making architecture degraded from 100% to 70% effectiveness over a course of ~ 1 hour on average (while the event was instantaneous, the effects take time to occur fully). The rate of departure is ~33% per hour loss of effectiveness. See Fig. 47.

Note here that the because the augmented capacity is not available in time prior to the disruption reaching its full effect, the Rate of Departure for the Base MOC and Augmented MOC are essentially equivalent. Additionally, care should be taken in determining the time at which the minimum performance is assessed using Eq. 2 as shown in Fig. 47. Since we are typically considering stochastic systems, the absolute point of minimum performance could skew the calculation of Eq. 2. It is recommended to use the point at which the system enters this new state of degraded performance, rather than the numerically absolute minimum performance which could significantly change the denominator of Eq. 2. In the MOC case study, we used the time at which the system enters the area of minimum (i.e., disrupted) performance, versus the absolute time of minimum performance. See Fig. 48.

$$TOL_{RD} = \frac{\left[\frac{L_p \cap L_r}{L_p}, t_d\right] - \left[\frac{L_p \cap L_r}{L_p}, t_{\min}\right]}{t_{\min} - t_d}$$

$$TOL_{RD} = \frac{1.0 - 0.70}{48.9 - 48} = 0.33$$



(A) (B)

Figures shown with the requirements locus as a transparent box for clarity. (The Requirements Locus is the inside of the box)

Fig. 47: Computing Rate of Departure in the MOC Case Study

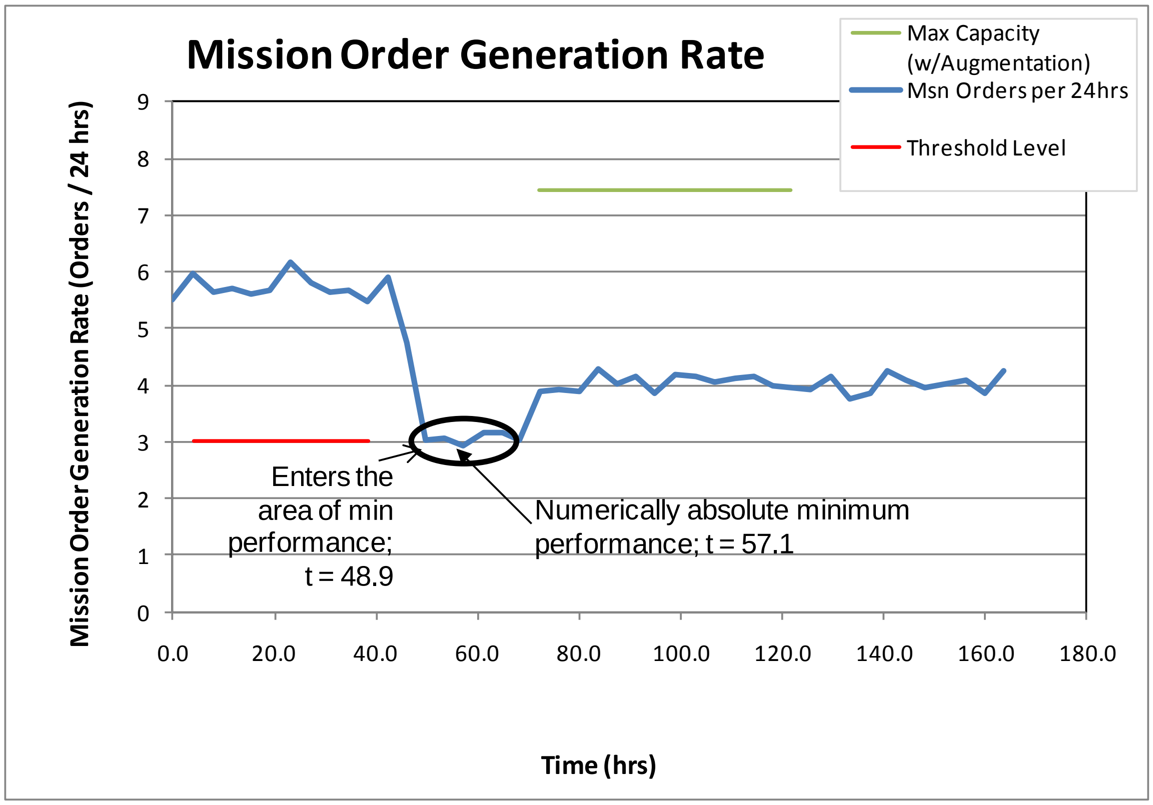


Fig. 48: Area of Minimum Performance versus Numerically Absolute Time

In addition to being executable (supporting simulation), Petri Nets have a graph theoretic interpretation that enables the analysis of properties. The identical model used in the simulations above (see Fig. 41- 48) was also analyzed in static form to assess other aspects of Tolerance and as well as Flexibility. As described in Chapter 3, examining these other aspects of Tolerance and Flexibility require an ability to determine the information flow paths which form the simple functionalities describing the overall capability under study. The information flow paths are derived from the place invariants in the architecture. CAESAR III generates the simple information flow paths associated with this net. Figure 49 shows an example simple information flow path of the Augmented MOC.

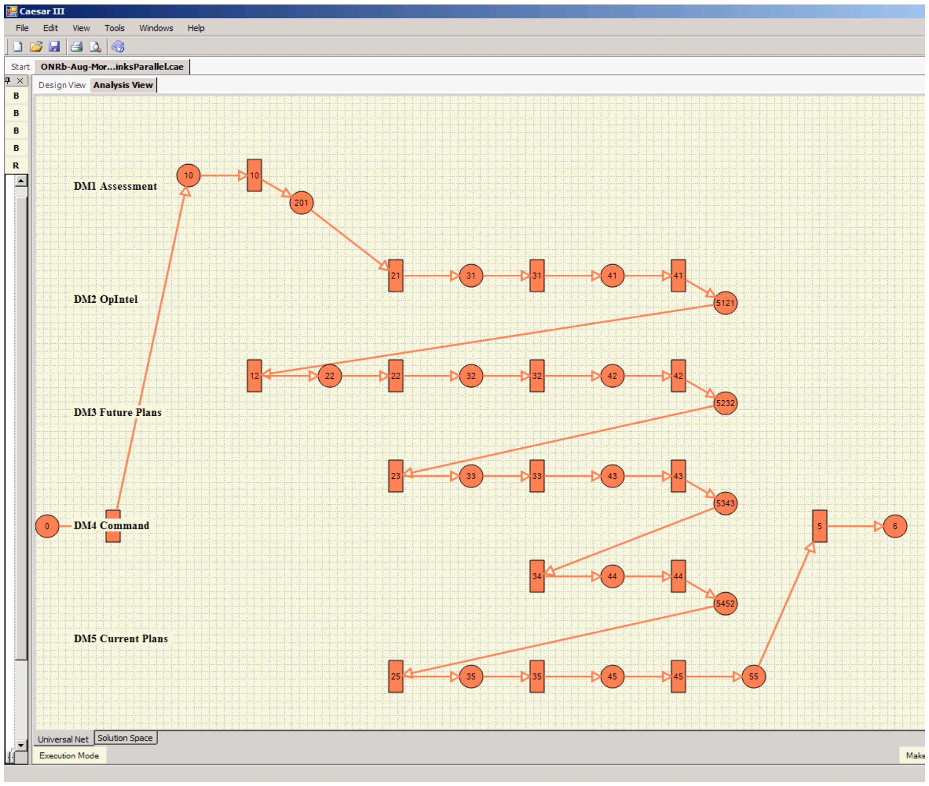


Fig. 49: Example Simple Information Flow Path of the Augmented MOC

Fault Tolerance (Tol_{FT}) is a measure which uses the graph-theoretic properties of Petri nets. Recall that Fault Tolerance (Tol_{FT}) is the ratio of simple information flow paths which may be disrupted prior to the loss of the capability to the total number of simple information flow paths. Those elements of the sub-graph (vertices) which can be removed without disconnecting the sub-graph or eliminating the complete functionality (capability) are those that may be disrupted.

From the universal net shown in Fig. 40, there are 44 simple information flow paths, containing as many as 37 elements, or as few as 13 elements out of a total of 74 elements contained in the universal net of the Augmented MOC. This large number of information flow paths results from the high level of interconnectivity and redundancy within the augmented MOC organizational design. The Base MOC contains only eight (8) information flow paths. Table 7 shows the elements contained within each simple information flow path for both the base MOC (8 paths) and the Augmented MOC (44 paths). Note that the Augmented MOC contains all 8 of the information flow paths contained by the Base MOC.

Table 7: Information Flow Paths in the Base and Augmented MOC

Base Path	Aug Path	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
0	0	p0	t0	p10	t10	p201	t21	p31	t31	p41	t41	p5121	t12	p22	t22	p32	t32	p42	t42	p5232	t23	p33	t33	p43	t43	p53	t5	p6										
1	1	p0	t0	p10	t10	p201	t21	p31	t31	p41	t41	p5121	t12	p22	t22	p32	t32	p42	t42	p5232	t23	p33	t33	p43	t43	p5343	t34	p44	t44	p5452	t25	p35	t35	p45	t45	p55	t5	p6
2	2	p0	t0	p10	t10	p201	t21	p31	t31	p41	t41	p5121	t12	p22	t22	p32	t32	p42	t42	p5232	t23	p33	t33	p43	t43	p5353	t35	p45	t45	p55	t5	p6						
	3	p0	t0	p10	t10	p201	t21	p31	t31	p41	t41	p5121	t12	p227	t27	p37	t37	p47	t47	p5732	t23	p33	t33	p43	t43	p53	t5	p6										
	4	p0	t0	p10	t10	p201	t21	p31	t31	p41	t41	p5121	t12	p227	t27	p37	t37	p47	t47	p5732	t23	p33	t33	p43	t43	p5343	t34	p44	t44	p5452	t25	p35	t35	p45	t45	p55	t5	p6
	5	p0	t0	p10	t10	p201	t21	p31	t31	p41	t41	p5121	t12	p227	t27	p37	t37	p47	t47	p5732	t23	p33	t33	p43	t43	p5353	t35	p45	t45	p55	t5	p6						
	6	p0	t0	p10	t10	p206	t26	p36	t36	p46	t46	p5671	t17	p27	t26	p37	t37	p47	t47	p5732	t23	p33	t33	p43	t43	p53	t5	p6										
	7	p0	t0	p10	t10	p206	t26	p36	t36	p46	t46	p5671	t17	p27	t26	p37	t37	p47	t47	p5732	t23	p33	t33	p43	t43	p5343	t34	p44	t44	p5452	t25	p35	t35	p45	t45	p55	t5	p6
	8	p0	t0	p10	t10	p206	t26	p36	t36	p46	t46	p5671	t17	p27	t26	p37	t37	p47	t47	p5732	t23	p33	t33	p43	t43	p5353	t35	p45	t45	p55	t5	p6						
	9	p0	t0	p10	t10	p206	t26	p36	t36	p46	t46	p5671	t17	p272	t22	p32	t32	p42	t42	p5232	t23	p33	t33	p43	t43	p53	t5	p6										
	10	p0	t0	p10	t10	p206	t26	p36	t36	p46	t46	p5671	t17	p272	t22	p32	t32	p42	t42	p5232	t23	p33	t33	p43	t43	p5343	t34	p44	t44	p5452	t25	p35	t35	p45	t45	p55	t5	p6
	11	p0	t0	p10	t10	p206	t26	p36	t36	p46	t46	p5671	t17	p272	t22	p32	t32	p42	t42	p5232	t23	p33	t33	p43	t43	p5353	t35	p45	t45	p55	t5	p6						
3	12	p0	t0	p11	t11	p21	t21	p31	t31	p41	t41	p5121	t12	p22	t22	p32	t32	p42	t42	p5232	t23	p33	t33	p43	t43	p53	t5	p6										
4	13	p0	t0	p11	t11	p21	t21	p31	t31	p41	t41	p5121	t12	p22	t22	p32	t32	p42	t42	p5232	t23	p33	t33	p43	t43	p5343	t34	p44	t44	p5452	t25	p35	t35	p45	t45	p55	t5	p6
5	14	p0	t0	p11	t11	p21	t21	p31	t31	p41	t41	p5121	t12	p22	t22	p32	t32	p42	t42	p5232	t23	p33	t33	p43	t43	p5353	t35	p45	t45	p55	t5	p6						
	15	p0	t0	p11	t11	p21	t21	p31	t31	p41	t41	p5121	t12	p227	t27	p37	t37	p47	t47	p5732	t23	p33	t33	p43	t43	p53	t5	p6										
	16	p0	t0	p11	t11	p21	t21	p31	t31	p41	t41	p5121	t12	p227	t27	p37	t37	p47	t47	p5732	t23	p33	t33	p43	t43	p5343	t34	p44	t44	p5452	t25	p35	t35	p45	t45	p55	t5	p6
	17	p0	t0	p11	t11	p21	t21	p31	t31	p41	t41	p5121	t12	p227	t27	p37	t37	p47	t47	p5732	t23	p33	t33	p43	t43	p5353	t35	p45	t45	p55	t5	p6						
	18	p0	t0	p11	t11	p216	t26	p36	t36	p46	t46	p5671	t17	p27	t26	p37	t37	p47	t47	p5732	t23	p33	t33	p43	t43	p53	t5	p6										
	19	p0	t0	p11	t11	p216	t26	p36	t36	p46	t46	p5671	t17	p27	t26	p37	t37	p47	t47	p5732	t23	p33	t33	p43	t43	p5343	t34	p44	t44	p5452	t25	p35	t35	p45	t45	p55	t5	p6
	20	p0	t0	p11	t11	p216	t26	p36	t36	p46	t46	p5671	t17	p27	t26	p37	t37	p47	t47	p5732	t23	p33	t33	p43	t43	p5353	t35	p45	t45	p55	t5	p6						
	21	p0	t0	p11	t11	p216	t26	p36	t36	p46	t46	p5671	t17	p272	t22	p32	t32	p42	t42	p5232	t23	p33	t33	p43	t43	p53	t5	p6										
	22	p0	t0	p11	t11	p216	t26	p36	t36	p46	t46	p5671	t17	p272	t22	p32	t32	p42	t42	p5232	t23	p33	t33	p43	t43	p5343	t34	p44	t44	p5452	t25	p35	t35	p45	t45	p55	t5	p6
	23	p0	t0	p11	t11	p216	t26	p36	t36	p46	t46	p5671	t17	p272	t22	p32	t32	p42	t42	p5232	t23	p33	t33	p43	t43	p5353	t35	p45	t45	p55	t5	p6						
	24	p0	t0	p11	t11	p217	t27	p37	t37	p47	t47	p5732	t23	p33	t33	p43	t43	p53	t5	p6																		
	25	p0	t0	p11	t11	p217	t27	p37	t37	p47	t47	p5732	t23	p33	t33	p43	t43	p5343	t34	p44	t44	p5452	t25	p35	t35	p45	t45	p55	t5	p6								
	26	p0	t0	p11	t11	p217	t27	p37	t37	p47	t47	p5732	t23	p33	t33	p43	t43	p5353	t35	p45	t45	p55	t5	p6														
6	27	p0	t0	p14	t14	p24	t24	p34	t34	p44	t44	p5454	t25	p35	t35	p45	t45	p55	t5	p6																		
7	28	p0	t0	p15	t15	p25	t25	p35	t35	p45	t45	p55	t5	p6																								
	29	p0	t0	p16	t16	p26	t26	p36	t36	p46	t46	p5671	t17	p27	t27	p37	t37	p47	t47	p5732	t23	p33	t33	p43	t43	p53	t5	p6										
	30	p0	t0	p16	t16	p26	t26	p36	t36	p46	t46	p5671	t17	p27	t27	p37	t37	p47	t47	p5732	t23	p33	t33	p43	t43	p5343	t34	p44	t44	p5452	t25	p35	t35	p45	t45	p55	t5	p6
	31	p0	t0	p16	t16	p26	t26	p36	t36	p46	t46	p5671	t17	p27	t27	p37	t37	p47	t47	p5732	t23	p33	t33	p43	t43	p5353	t35	p45	t45	p55	t5	p6						
	32	p0	t0	p16	t16	p26	t26	p36	t36	p46	t46	p5671	t17	p272	t22	p32	t32	p42	t42	p5232	t23	p33	t33	p43	t43	p53	t5	p6										
	33	p0	t0	p16	t16	p26	t26	p36	t36	p46	t46	p5671	t17	p272	t22	p32	t32	p42	t42	p5232	t23	p33	t33	p43	t43	p5343	t34	p44	t44	p5452	t25	p35	t35	p45	t45	p55	t5	p6
	34	p0	t0	p16	t16	p26	t26	p36	t36	p46	t46	p5671	t17	p272	t22	p32	t32	p42	t42	p5232	t23	p33	t33	p43	t43	p5353	t35	p45	t45	p55	t5	p6						
	35	p0	t0	p16	t16	p261	t21	p31	t31	p41	t41	p5121	t12	p22	t22	p32	t32	p42	t42	p5232	t23	p33	t33	p43	t43	p53	t5	p6										
	36	p0	t0	p16	t16	p261	t21	p31	t31	p41	t41	p5121	t12	p22	t22	p32	t32	p42	t42	p5232	t23	p33	t33	p43	t43	p5343	t34	p44	t44	p5452	t25	p35	t35	p45	t45	p55	t5	p6
	37	p0	t0	p16	t16	p261	t21	p31	t31	p41	t41	p5121	t12	p22	t22	p32	t32	p42	t42	p5232	t23	p33	t33	p43	t43	p5353	t35	p45	t45	p55	t5	p6						
	38	p0	t0	p16	t16	p261	t21	p31	t31	p41	t41	p5121	t12	p227	t27	p37	t37	p47	t47	p5732	t23	p33	t33	p43	t43	p53	t5	p6										
	39	p0	t0	p16	t16	p261	t21	p31	t31	p41	t41	p5121	t12	p227	t27	p37	t37	p47	t47	p5732	t23	p33	t33	p43	t43	p5343	t34	p44	t44	p5452	t25	p35	t35	p45	t45	p55	t5	p6
	40	p0	t0	p16	t16	p261	t21	p31	t31	p41	t41	p5121	t12	p227	t27	p37	t37	p47	t47	p5732	t23	p33	t33	p43	t43	p5353	t35	p45	t45	p55	t5	p6						
	41	p0	t0	p16	t16	p262	t22	p32	t32	p42	t42	p5232	t23	p33	t33	p43	t43	p53	t5	p6																		
	42	p0	t0	p16	t16	p262	t22	p32	t32	p42	t42	p5232	t23	p33	t33	p43	t43	p5343	t34	p44	t44	p5452	t25	p35	t35	p45	t45	p55	t5	p6								
	43	p0	t0	p16	t16	p262	t22	p32	t32	p42	t42	p5232	t23	p33	t33	p43	t43	p5353	t35	p45	t45	p55	t5	p6														

Represented in this fashion, the methodology based on Lemma 2 described in Section 3.1.2.2 is used to determine the cut vertices needed to compute Fault Tolerance. Recall from Lemma 2 any vertex that is on every path from sources S_i to sink U_j is a cut vertex. The Augmented MOC includes one source (*p0*) and two sinks (*p53, p55*). (With sinks defined as *p53* and *p55*, this eliminates the need to consider elements *t5* and *p6*, which are elements after the sinks as designated above.) We can partition the above set of information flow paths into the 14 which have *p53* as a sink, and the 30, which have *p55* as a sink. Solving for the elements common to every path from source *p0* to sink *p53*, and from source *p0* to sink *p55*, we find the following cut vertices:

sink p53 cut vertices: {p33,p43,t0,t23,t33,t43}

sink p55 cut vertices: {p45,t0,t35,t45}

There are no cut vertices common to both sinks except for *t0*. The set of non-cut vertices, (V_{nc}) includes every other element. In this example, every information flow path contains many non-cut vertices, meaning multiple flow paths exist to connect each source to its corresponding sink. In the Augmented MOC, there are 44 information flow paths so that r = 44. Since every flow path contains multiple non-cut vertices, meaning multiple flow paths exist to connect each source to its corresponding sink:

$\ell_{1\ldots}\ \ell_{44}$each contain members of V_{nc} therefore $x_{1..44} = 1$

$$\mathrm{Tol}_{\mathrm{FT}} = \frac{x}{r} = \frac{\sum_{i=1}^{r} x_i}{r} = \frac{44}{44}$$

The MOC with Augmented Capacity displays maximum fault tolerance. Every information flow path can be disrupted in some way without a loss of capability. This result is not surprising, given the extensive redundancy and interconnectivity built into the Augmented MOC organizational design. Additionally, the parallel dissemination of information from *t0* fosters an embedded redundancy in the transmission of information. This parallel transmission structure is typical in military organizations, where a "warning order" is often broadly disseminated to initiate early planning activities. What this means is that multiple paths exist connecting the source to each sink, such that the elimination of a single element does not result in elimination of the overall capability.

Point of Failure Tolerance, ($\mathrm{Tol}_{\mathrm{PF}}$) examines a situation different from Fault Tolerance. $\mathrm{Tol}_{\mathrm{PF}}$ captures the relatedness of a local failure of any given element to the broader loss of functionality or loss of capability. More generally, $\mathrm{Tol}_{\mathrm{PF}}$ addresses whether an element-level failure induces a system-level failure. This is accomplished by examining the localization of failures during a disruption. Elements which are a member of only one simple information flow path are said to have localized failure effects. Table 8 associates the elements of the Petri net based architecture of the Augmented MOC with the information flow paths while Table 9 addresses the Base MOC. Executing Eq. 4 from Section 3.2.1.3 yields:

For the Augmented MOC:

$$\text{Tol}_{\text{PF}} = \frac{\sum_{j=1}^{E} q_j}{E} = \frac{q}{E} = \frac{8}{71} = 0.11$$

For the Base MOC:

$$\text{Tol}_{\text{PF}} = \frac{\sum_{j=1}^{E} q_j}{E} = \frac{q}{E} = \frac{8}{50} = 0.16$$

These results imply highly non-localized failures in both the Augmented and Base MOC. Only about 11% of the elements are associated with one information flow path in the Augmented MOC, and 16% in the Base MOC. In the case of Point of Failure Tolerance, higher is better because this indicates a higher proportion of elements with localized failure effects.

Table 8: Augmented MOC: Associating Elements with Information Flow Paths

Element	# Flow Paths Associated w/Element	q_i =
p0	44	0
p10	12	0
p11	15	0
p14	1	1
p15	1	1
p16	15	0
p21	6	0
p22	9	0
p24	1	1
p25	1	1
p26	6	0
p27	9	0
p31	18	0
p32	21	0
p33	42	0
p34	1	1
p35	16	0
p36	18	0
p37	21	0
p41	18	0
p42	21	0
p43	42	0
p44	15	0
p45	30	0
p46	18	0
p47	21	0
p53	14	0
p55	30	0
p6	44	0
p201	6	0
p206	6	0
p216	6	0
p217	3	0
p227	9	0
p261	6	0
p262	3	0
p272	9	0
p5121	18	0
p5232	21	0
p5343	14	0
p5353	14	0
p5452	15	0
p5671	18	0
p5732	21	0
t0	44	0
t10	12	0
t11	15	0
t12	18	0
t14	1	1
t15	1	1
t16	15	0
t17	18	0
t21	18	0
t22	21	0
t23	42	0
t24	1	1
t25	16	0
t26	18	0
t27	15	0
t31	18	0
t32	21	0
t33	42	0
t34	15	0
t35	30	0
t36	18	0
t37	21	0
t41	18	0
t42	21	0
t43	42	0
t44	15	0
t45	30	0
t46	18	0
t47	21	0
t5	44	0
E= 74	1308	q = 8

Table 9: Base MOC: Associating Elements with Information Flow Paths

Element	# Flow Paths Associated w/Element	q =		Element	# Flow Paths Associated w/Element	q =
p0	8	0		p5121	6	0
p10	3	0		p5232	6	0
p11	3	0		p5343	2	0
p14	1	1		p5353	2	0
p15	1	1		p5452	3	0
p16	0	0		p5671	0	0
p21	3	0		p5732	0	0
p22	6	0		t0	8	0
p24	1	1		t10	3	0
p25	1	1		t11	3	0
p26	0	0		t12	6	0
p27	0	0		t14	1	1
p31	6	0		t15	1	1
p32	6	0		t16	0	0
p33	6	0		t17	0	0
p34	1	1		t21	6	0
p35	4	0		t22	6	0
p36	0	0		t23	6	0
p37	0	0		t24	1	1
p41	6	0		t25	4	0
p42	6	0		t26	0	0
p43	6	0		t27	0	0
p44	3	0		t31	6	0
p45	6	0		t32	6	0
p46	0	0		t33	6	0
p47	0	0		t34	3	0
p53	2	0		t35	6	0
p55	6	0		t36	0	0
p6	8	0		t37	0	0
p201	3	0		t41	6	0
p206	0	0		t42	6	0
p216	0	0		t43	6	0
p217	0	0		t44	3	0
p227	0	0		t45	6	0
p261	0	0		t46	0	0
p262	0	0		t47	0	0
p272	0	0		t5	8	0
				E = 50	222	q = 8

Point of Failure Tolerance is also intended to draw an architect's attention to areas in the design where greater attention may be required. In the Augmented MOC, of particular interest is that 42 of the 44, or about 95%, of the information flow paths use elements: *t23, p33, t33, p43,* and *t43*; and 30 of the 44, or almost 70%, of the information flow paths use elements: *p45* and *t45*. From Fig. 40, we can see this represents the command and current operations cells respectively. While it is natural for a military organization to rely heavily on the commander to make decisions, a disruption affecting this portion of the organizational design would have broad ranging consequences. The architect should direct attention at these portions of the architecture to determine if changes are required.

5.2.3.3. Decision Making Organization Case Study Results: Flexibility

The final set of metrics to examine in the decision making organization case study deal with flexibility, where flexibility refers to the ability of a system to reorganize and adapt itself to changing conditions. Recall from Chapter 3, one measure of flexibility is Cohesion, as defined by Liles [2008]. Cohesion looks at the average cohesion of the individual nodes. A set of nodes with higher cohesion implies that the individual nodes are less flexible and less resilient.

We will examine flexibility where each decision making organization within the MOC identified in Figs. 37 and 38 is treated as a node (i.e. Assessment, Operational Intelligence, Future Plans, Command, Current Plans, and Current Operations). Executing Eq. 5 and Eq. 6 from Chapter 3 yields the results shown in Fig. 50. These results show that the Augmented MOC is less cohesive than the Base MOC and therefore more flexible.

Cohesion (Mult Nodes) Augmented MOC

Node	Inputs	Outputs	Paths	Coh (nki)
DM1	1	2	2	1.00
DM2	3	3	5	0.56
DM3	3	2	4	0.67
DM4	2	3	6	1.00
DM5	2	1	2	1.00
DM6	3	1	3	1.00
DM7	3	3	5	0.56
DM8	3	2	4	0.67

m = 8 Coh(f_k) = 0.81

Eq (5) $$Coh(n_{ki}) = \frac{z_{ki}}{x_{ki}}$$

Eq (6) $$Coh(f_k) = \frac{\sum_{i=1}^{m} Coh(n_{ki})}{m}$$

Cohesion (Mult Nodes) Base MOC (non- Augmented)

Node	Inputs	Outputs	Paths	Coh (nki)
DM1	1	1	1	1.00
DM2	2	1	2	1.00
DM3	1	1	1	1.00
DM4	1	3	3	1.00
DM5	2	1	2	1.00
DM6	3	1	3	1.00

m = 6 Coh(f_k) = 1.00

Fig. 50: Calculating Cohesion in the MOC

These results are somewhat intuitive. In this case study, we are essentially adding capacity for the intelligence and future planning functionality through augmentation cells which provide a redundant capability in those areas. This should naturally increase the flexibility of the MOC as an organization. This approach quantifies that increase.

Liles [2008] also introduces a second measure of flexibility, which we have renamed as Common Use. Recall that CU measures the extent of reuse of the elements to support multiple simple functionalities that comprise the overall capability. Tables 8 and 9 associate the number of simple functionalities that each element is a member. Executing Eq. 7 yields the following:

Augmented MOC:

$$\text{Common Use (CU)} = \frac{\sum_{j=1}^{E} A}{E} = \frac{1308}{74} = 17.7$$

Base MOC:

$$\text{Common Use (CU)} = \frac{\sum_{j=1}^{E} A}{E} = \frac{222}{50} = 4.4$$

From Common Use alone, it is difficult to determine whether 4.4 vs. 17.7 is an improvement or not. This is because there are 44 information flow paths in the Augmented MOC, but only 8 in the Base MOC. Therefore, the numbers for Common Use will be inherently different. The next section helps explain these metrics in a more comparable fashion to support evaluation.

Recall from Section 3.1.3.3 that we defined Proportion of Use as the relative proportion of elements used by any given simple functionality to deliver the overall capability. We note two principal advantages to this metric. First, it describes what proportion of the elements is contained within the average simple functionality of a capability. For example, does the average functionality use a relatively small (say 10%) or a relatively large (say 80%) of the elements? As proportion of use increases, a disruption to a given element within a capability is more likely to have broad ranging effects. Systems with high proportion of use are more difficult to reorganize (less flexible), because each element is more related to each functionality. Second, proportion of use normalizes the Common Use such that one can compare different architectures from a common perspective. This allows us to determine whether a particular value for Common Use is comparatively high or low. Figures 51 and 52show the results of computing Proportion of Use for the Augmented and Base MOC alternatives.

For the MOC with Augmentation, Proportion of Use is 0.4, meaning that each simple functionality involves about 40% of the elements required to deliver the capability. In the base MOC without augmentation, each simple functionality involves approximately 56% of the total elements. From this perspective, we can say that the augmented MOC is more flexible. In the augmented MOC, a disruption to a given element can be expected to affect about 40% of the overall functionality of the system under evaluation. In the base MOC, a disruption to a given element can be expected to affect about 56%.

Inf Flow Path	# Elements Contained by ℓ$_i$
ℓ = 1	27
ℓ = 2	27
ℓ = 3	37
ℓ = 4	31
ℓ = 5	37
ℓ = 6	31
ℓ = 7	19
ℓ = 8	13
ℓ = 9	27
ℓ = 10	26
ℓ = 11	27
ℓ = 12	27
ℓ = 13	26
ℓ = 14	27
ℓ = 15	19
ℓ = 16	27
ℓ = 17	27
ℓ = 18	27
ℓ = 19	27
ℓ = 20	19
ℓ = 21	37
ℓ = 22	31
ℓ = 23	36
ℓ = 24	30
ℓ = 25	37
ℓ = 26	31
ℓ = 27	37
ℓ = 28	31
ℓ = 29	36
ℓ = 30	30
ℓ = 31	37
ℓ = 32	31
ℓ = 33	29
ℓ = 34	23
ℓ = 35	37
ℓ = 36	31
ℓ = 37	37
ℓ = 38	31
ℓ = 39	37
ℓ = 40	31
ℓ = 41	37
ℓ = 42	31
ℓ = 43	29
ℓ = 44	23

∑ B = 1308
E = 74
r = 44

$$\text{PoU} = \frac{\dfrac{\sum_{i=1}^{r} B_i}{E}}{r} = \frac{\sum_{i=1}^{r} B_i}{rE}$$

$$= \frac{1308}{74 * 44} = \frac{1308}{3256} = 40.4\%$$

Fig. 51: Proportion of Use in the Augmented MOC

Inf Flow Path	# Elements Contained by ℓ$_i$
ℓ = 1	27
ℓ = 2	27
ℓ = 3	37
ℓ = 4	31
ℓ = 5	37
ℓ = 6	31
ℓ = 7	19
ℓ = 8	13

∑ B = 222
E = 50
r = 8

$$\text{PoU} = \frac{\dfrac{\sum_{i=1}^{r} B_i}{E}}{r} = \frac{\sum_{i=1}^{r} B_i}{rE}$$

$$= \frac{222}{50 * 8} = \frac{222}{400} = 55.5\%$$

Fig. 52: Proportion of Use in the Base MOC

5.2.3.4. Decision Making Organization Case Study Overall Results

In this case study, we have applied the individual components of the resilience evaluation approach for both the Base MOC, and the Augmented MOC alternatives. The MOC is designed as a series of Decision Making Nodes, with each node as a five stage decision making structure with interactions between nodes. This architecture was transformed into an ordinary Petri Net using the theory outlined in Remy and Levis, 1986. Necessary logic and instrumentation were added to the Petri Net such that it became an executable form of the MOC architecture suitable for behavioral and performance analyses.

The capability under study was the capability to Generate Mission Orders, where the threshold performance level was to generate the order within 8 hours of receipt from the High Joint Command.

It is critical to consider the resilience 'of what' 'to what,' focusing on the resilience of a capability to a disruption in a particular environment (under what conditions). In this case study, we examined the resilience of the MOC's capability to 'Generate Mission Orders' to the disruption 'loss of situational awareness software.' When a new release was received, the update caused both versions to crash, and attempts to restart were unsuccessful. The MOC, transitioned from automated procedures based on the software, to manual procedures, and called upon augmented capabilities in the form of additional Operations Intelligence and Future Plans cells, which required additional time hours to establish. However, this augmented capability could not be established until well after the disruption had induced its full effect.

For both cases of the MOC, the disruption brought the MOC's capability to the brink of not meeting the threshold. If another disruption occurred before the augmented capacity could be brought online, the MOC would have been incapable of completing one of its key capabilities, the generation of mission orders. The Augmented capability did return a portion of the MOC's mission order generation capability, but not back to pre-disruption levels. Table 10 reports the results for each metric in the base MOC and the Augmented MOC.

Table 10: Resilience Metrics for the Base and Augmented MOC

Attribute	Metric	Measures	Question Answered	Augmented MOC	Base MOC
Capacity: *"the ability to operate at a certain level as defined by a given measure."*	Buffering Capacity	Available capability margin between current operating levels and a defined minimum threshold operating level at the time preceding a disruption.	Can a disruption be absorbed with immediately available (on-hand) resources?	47%	47%
	Reactive Capacity	Available capability margin between maximum operating levels (i.e. including any spare capacity) and a defined minimum threshold operating level.	Can a disruption be absorbed with the addition of spare capacity?	60%	0%
	Residual Capacity	Available capability margin between operating levels at the end of the survival phase and a defined minimum threshold operating level.	Given survival, how vulnerable is the system to a follow-on disruption that occurs before the system can recover?	~0%	~0%
Tolerance: *"the ability to degrade gracefully after a disruption"*	Rate of Departure	Rate of change in system performance with respect to its requirements (ie rate of loss of effectiveness) after a disruption.	What level of capability is lost per unit of time during the survival phase?	0.33	0.33
	Fault Tolerance	The ratio of simple functionalites which may be disrupted without a loss of capability to the total number of simple functionalities.	How many simple functionalities can be disrupted prior to losing the capability. Primarily a tool to draw architects attention to key areas in the design.	1.0	1.0
	Point of Failure Tolerance	Relatedness of failures at the element level to an overall loss of capability	Are element level failures relatively localized, or do failures incur broad system-level effects? Primarily a tool to draw the architect's attention to key design areas.	0.11	0.32
Flexibility: *"the ability of a system to reorganize its elements to maintain its capabilities"*	Cohesion	Relatedness of the elements within a node or module which support a given capability	How difficult is it to reorganize the system at the node / module level?	0.81	1
	Common Use	Extent of common use of the elements among the simple functionalities which support the overall capability.	Can a system execute multiple functionalites concurrently, or is it limited by competition for resources?	17.6	4.4
	Proportion of Use	The ratio of the total elements used by any given simple functionality to deliver the overall capability	Are most of the elements needed for a given functionality, making it more difficult to reorganize?	0.40	0.56

The architect along with the overall development team must consider which aspects of resilience are most important to the system under consideration. Figure 14 in Chapter 3 along with Table 10 assist the architect to determine which aspects of resilience are most applicable to the architecture definition and resilience issues at hand.   In the case of the MOC and capacity, the most appropriate metric is Buffering Capacity. As was shown, reactive capacity is not available in time to play a role in the survival phase, although it does distinguish the two candidate architectures and, once in place, the augmented capacity offers a number of benefits. Further, since this was a non-malicious type of disruption, the residual capacity is of less concern because a follow-on disruption is not necessarily likely. For Tolerance, the most appropriate metric is again Rate of Departure. In this case, Rate of Departure directly measured the time sensitive nature of the MOC's capability of generate mission orders by assessing the rate of generation and the number of "late" orders delivered to subordinate units. In terms of flexibility, Proportion of Use was also selected because it directly addresses the ability of the system to be reorganized based on the average use of the elements across the simply functionalities supporting that capability. The cohesion metric is not as useful, because the disruption is likely to take effect much more quickly than any reorganization could occur. This makes cohesion a metric potentially more useful in the 'recovery' phase of resiliency. Additionally, the overlap in the Common Use and Proportion of Use was discussed, leading to a selection of Proportion of Use for this assessment.

Having determined which metrics are most important, resilience can be considered from an intersecting requirements and performance locus perspective. The two alternative architectures can be compared in this manner. Determining the resilience requirements locus requires value judgment. The example is shown with a requirements locus:

- 33% Buffering Capacity
- < 50% Rate of Departure (Tolerance)
- < 50% Proportion of Use (Flexibility)

Figure 53 shows the results of the overall evaluation. The Augmented MOC meets the resilience attribute requirements of capacity, tolerance and flexibility making it the preferred candidate architecture from the point of view of resiliency. The Base MOC lacks required flexibility but meets the other requirements.

Figure 54 shows a comparison that better captures the difference of augmentation between the two candidate architectures. The Augmented MOC is able to bring reactive capacity on-line, whereas the Base MOC is not. Using the same perspective, the following graph shows the impact of considering capacity in terms of reactive capacity rather than buffering capacity. While these results do not change the overall resilience comparison of alternative architectures, it is shown here as another possible viewpoint since the two alternatives differ in terms of reactive capacity.

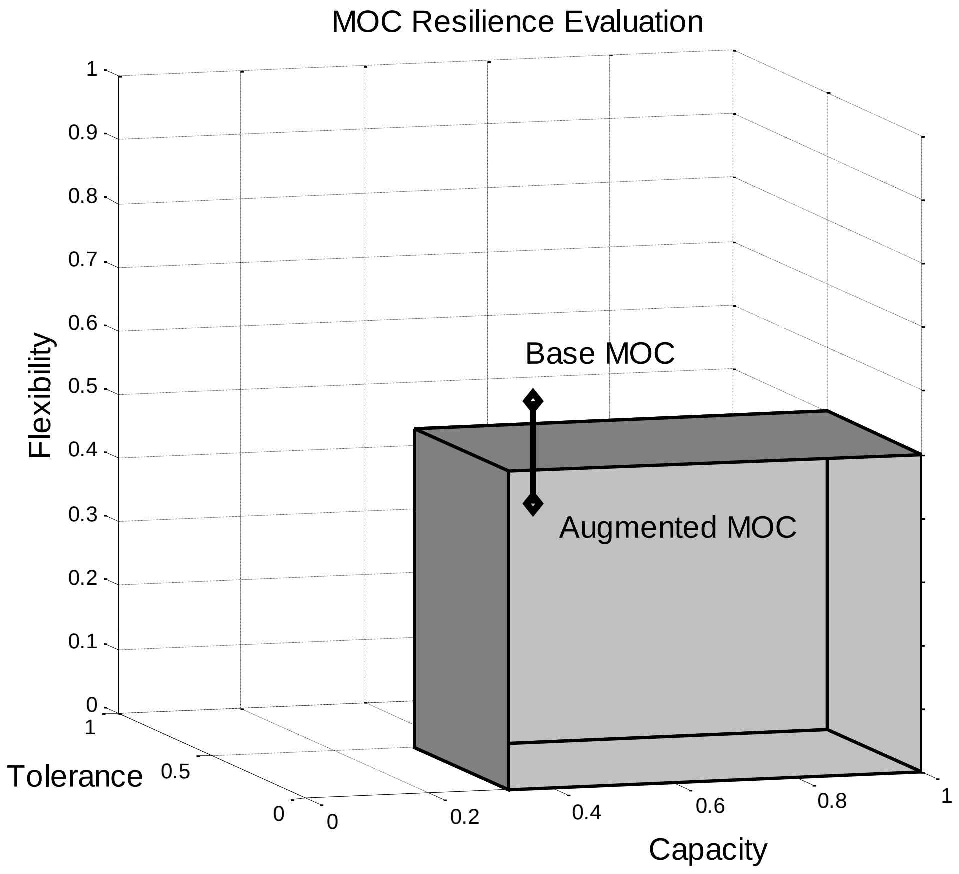


Fig. 53: Resilience Evaluation of the Base and Augmented MOC

As we consider the performance of the two designs, Base MOC, and Augmented MOC, the evaluation framework allows us to make useful, quantitative comparisons. In the case of capacity, the two designs have equivalent buffering capacity and residual capacity. While the augmented MOC has greater reactive capacity, the augmented MOC cannot bring that reactive capacity on line fast enough to make a difference in the survival phase. In the case of point of failure tolerance, the Base MOC actually performs better. This is because its failures are the most localized. More specifically, about 1/3 (0.32) of the base

MOC element failures are localized, as compared to ~ 1/10 (0.11) of the augmented MOC. The greater interconnectivity of the augmented MOC accounts for this difference. In the case of flexibility, the augmented MOC performs best in terms of Proportion of Use. A smaller proportion of its elements, on average, are needed for a given functionality, as compared to the base MOC (40% vs. 56%).

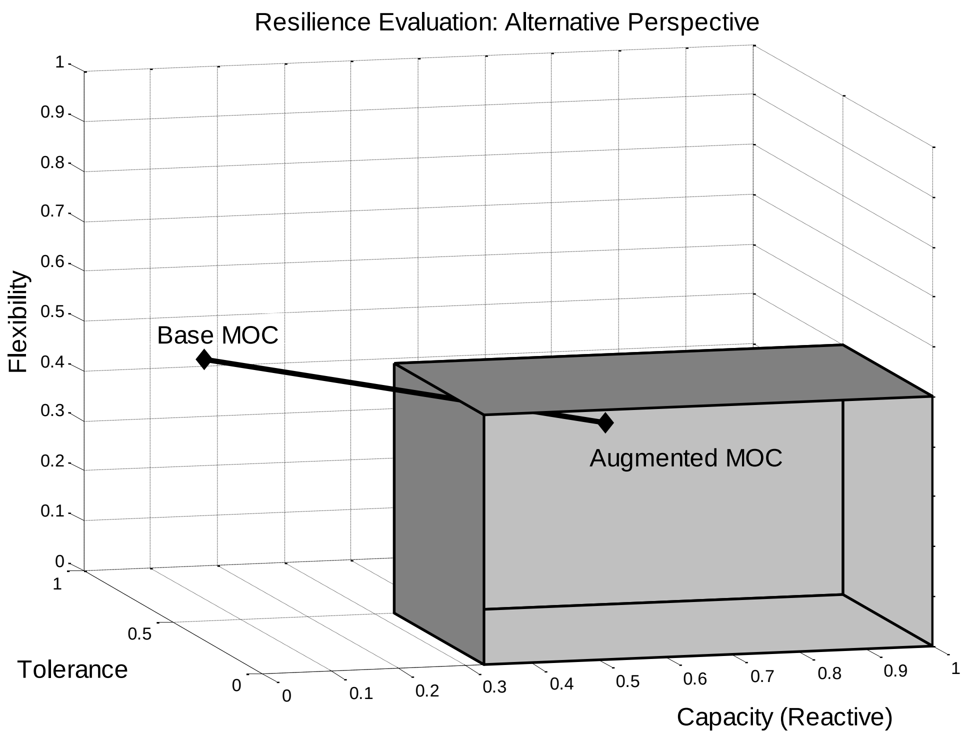


Fig. 54: An Alternative Resilience Evaluation for the Base and Augmented MOC

The primary advantage of the augmented MOC (additional reactive capacity) is not relevant during the survival phase; the augmented and base MOC perform equivalently in terms of buffering capacity. However, the reactive capacity does allow the augmented MOC to restore performance above the (T), making a future disruption less likely to have catastrophic effect when compared to the base MOC. In terms of tolerance, the augmented MOC performs worse because its failures are less localized and therefore more likely to have broader, system level effects. In terms of flexibility, the augmented MOC performs better; the fact that fewer elements are needed for the average functionality will make the augmented MOC more easily reorganized.

This case study provided a simple demonstration of how to compare alternative architectures in terms of resilience. It applied the same methodologies as in the targeting case study. A MOC case study was developed in a 5 stage architecture model for representing Decision Making organizations approach as supported by CAESAR III. CAESAR III supports the translation of the 5-stage DM model to Petri Net form. From this point forward the methodology is identical to that used in the targeting architecture case study. Resilience was evaluated using the architecture by: identifying the appropriate measure for each resilience attribute and mapping the architecture performance of each measure against an overlaid requirements locus. The resilience-related potential improvements to the design were highlighted and alternatives can now be quantified or compared. This case study demonstrated that the approach to evaluating resilience is robust to differing architecture methodologies.

CHAPTER 6

CONCLUSIONS

6.1. SUMMARY

This research focused on the challenge of resilience. Chapter One identified the well accepted need for national defense and civilian structure infrastructure systems resilient to disruption. Today's society is embarking on engineering challenges of ever greater complexity, and failures of these new systems often bring significant consequence. Command and control systems, in particular, tend to be mission critical, where failures in mission critical systems often mean the failure of the entire mission. Early engineering activities for mission critical systems should include an evaluation of resilience. A need exists for quantitative methodologies to measure resilience and implement changes to systems. The approach developed in this research offers a potential solution to address this need for command and control systems.

Our hypothesis was that the resilience of command and control systems could be measured by its attributes using the architecture. In measuring the attributes, the architecture can be improved to include desirable features, or to eliminate undesirable ones. By representing the architecture of a system in a rigorous way, one can analyze the design for key properties, and simulate the design to examine for desired performance and behavior aspects. In this manner, one can make decisions and improvements far earlier in the process, saving time, money and ultimately delivering better results.

Petri Net based architecture models are used in the approach developed in this research for a number of reasons. They are rigorous (meaning that defined mathematical models underlie all aspects of Petri Net theory), visualize-able because of its graph theoretic underpinnings, and executable. These properties of Petri Nets support analyzing structural, behavioral, and performance characteristics of the architecture via simulation as well as static analyses. Finally, established and traceable means exist for translating other architectural approaches (for example Business Process Model and Notation or BPMN) into Petri Net format.

The approach developed in this research began with the attributes of resilience already identified in basic form in the existing literature. Significant work already exists in the literature, defining key ideas in the definition and characteristics of resilience. However, existing research relied primarily on heuristics for measuring resilience or as guidelines for developing resilient systems. The existing work in resilience tends not be quantitative does not include the architecture. Quantitative results and approaches are required in engineering systems. The approach demonstrated in this research uses the architecture, meaning that existing engineering products can be leveraged in novel ways, and the results of these analyses can be explicitly implemented in the design of the system.

Chapter Two summarized the existing body of research in the field of resilience. It described the various definitions of resilience in the existing literature, and noted the common thread of disruption avoidance survival and recovery present in the body of knowledge. Therefore, we used Jackson's [2010] definition of resilience: the ability to avoid, survive and recover from disruption. Disruptions were defined as sudden or sustained events leading to a loss of performance and jeopardizing the system's ability to perform its mission [INCOSE Resilient Systems Working Group], [Madni and Jackson, 2008], and [Jackson, 2010]. We additionally used the attributes of resilience summarized in Jackson [2010]: capacity, tolerance, flexibility, and inter-element collaboration. Inter-element collaboration is later scoped out of this research due to the difficulties in assessing emergent properties. This research noted that resilience must consider time, and acknowledged importance of understanding the resilience "of what to what," from Carpenter et al. [2001]. This research focused on the resilience of a command and control system to implement a capability, where the disruptions were defined to two separate case studies.

Chapter Three described the approach developed in this research for evaluating resilience using the architecture. Each of the attributes of resilience: capacity, tolerance, and flexibility are further defined and extended well beyond the existing literature. Quantifiable measures are proposed for each, and a holistic framework for combining the measures is shown. Capacity is divided into buffering, residual, and reactive capacity, where reactive capacity is further explained with additive and surrogate reactive capacity. Tolerance is separated into graceful degradation, fault tolerance, and point of failure tolerance. Flexibility leverages existing measures from Liles [2008]: cohesion and common use (renamed from degree of reuse), and adds the measure proportion of use.

The calculation of each measure is demonstrated using a series of simple examples. These measures leverage the executable characteristics of Petri Nets (buffering capacity, reactive capacity residual capacity, and graceful degradation) which assess performance and behavior, as well as the graph-theoretic aspects of Petri nets (fault tolerance, point of failure tolerance, cohesion, common use, and proportion of use) which assess behavioral and structural aspects. For many of these measures, the methodology shows how an architect's attention is directed to key aspects highlighted by the metric that support improvement in the design. Chapter Three also provides a framework to assist the architect and overall development team in selecting which of these metrics best apply to their system. Finally, Chapter Three provides a holistic approach to assessing overall resilience of the system. The system's architecture's performance is overlaid upon a resilience requirements locus to determine where shortfalls in performance and behavior exist. This final step supports comparison of alternatives, or measuring improvements to an architecture to move its performance into a desired sector.

Chapter Four described the research approach. It identified two case studies intended to demonstrate the approach, and to show that it is robust against various architectural development styles, as well as robust against the type of command and control system under study. Each case study is developed using the Wagenhals and Levis[2008] approach, where a Mission and CONOPS are identified, the architecture is designed in static form, an executable model is constructed and the architecture is evaluated using the methods described in Chapter Three (see Fig. 16). The first case study examined a time sensitive targeting example, with a US Marine ground force conducting a fire support coordination capability. The second case study involved a naval decision making system, call the Maritime Operations Center (MOC), conducting the capability of generating mission orders. Chapter Four also summarized the key models used in this research, including CPN Tools, CAESAR III, Business Process Visual Architect, MATLAB, SEAT, and MS Excel.

Chapter Five developed each of the case studies and fully applied the resilience evaluation methodology. Each case is introduced with background information, a description of the architecture, a translation of the architecture into Petri net format, and a description of the potential disruption. The complete approach to evaluating resilience is then demonstrated for both case studies.

The targeting case study is presented first, where the disruption is a loss of GPS geo-location functionality to the capability of conduct fire support coordination. The targeting case study was shown to be sufficient with regards to tolerance and flexibility, but lacking required capacity. An excursion to the targeting case study, examines an aerostat navigation beacon as a reactive capacity to the loss of GPS. The aerostat excursion shows how a given architecture can be adapted to bring its performance with respect to resilience into a required sector.

The naval Maritime Operations Center (MOC) is presented next, where the disruption is the loss of a key software program supporting situational awareness to the capability of 'generate mission orders.' This case study presents an example of the comparison of two alternative architectures for the purposes of down-selection among alternative candidates.

A ‘Base MOC’ and an ‘Augmented MOC’ are examined, where the Augmented MOC contains additional parallel resources for intelligence and future operations planning. The Base MOC was shown to have sufficient capacity and tolerance, but lacking flexibility. The Augmented MOC met the requirements for resilience in for all three attributes: capacity, tolerance and flexibility. The two candidate architectures are also compared in terms of reactive capacity in a short excursion.

For both of these case studies, the purpose was to demonstrate the approach, detailing how resilience can be measured through its attributes using the architecture. Subject matter experts were used to assist in data collection and verification of the architectures. However, the point was not recommend these architectures or alternatives as definitive solutions, or to, for example, recommend investments in aerostat navigation beacons or augmented operations centers. Rather to demonstrate how credible architecture initiatives can be measured in terms of resilience.

6.2. CONTRIBUTIONS

This research accomplished several important goals and contributes significantly in the following ways:

1) Developed an Approach to Evaluating Resilience Through its Attributes. No formal, quantifiable means exist to evaluate the resilience of a proposed architecture to assist in design and selection alternative architectures. The approach uses the architecture, and develops measures along with an integrative framework from which to quantitatively evaluate resilience. The approach may be used in alternative architecture

down-selection, or the architect may use this information to make improvements to an existing design. The approach is important because it shows both how to measure, and what actions might be taken for improvement. It shows how to improve the resiliency of a given architecture to move it into the required sector and how to compare alternative architectures with respect to resilience.

2) Proposed New, and Significantly Extended Existing Measures of Resilience Attributes. The existing body of knowledge provides a qualitative foundation, but is not quantifiable. Starting with the existing body of knowledge in resilience, each of the attributes of resilience were better defined, and extended by provided metrics measuring key portions.

3) The approach developed in this research is important because it uses the architecture. The architecture is an existing engineering product. This approach demonstrates how, if generated in a rigorous way, the architect can use something that must be generated anyways (i.e. the architecture) to support an evaluation of resilience. This allows the architect to explicitly measure the architecture and point explicitly to areas in the design where improvements may be required. Petri nets were selected for their rigorous ability to support executable analysis as well as the analysis of properties, connecting structure to behavior.

4) Robust to Architectural Styles and the Type of Command and Control System. The proposed approach is intentionally demonstrated on two divergent case studies. One is an organizational architecture for a decision making organization. The

other is a time sensitive targeting architecture, focused more on hardware systems. One was developed using a CAESAR III using a five stage organization model with variable and static interactions. The other is developed in BPMN. The use of these two case studies is intended to show this robustness to architectural style and type of system.

5) Organized the conceptual foundations of resilience. Organizations often view resilience from multiple perspectives. Some focus on redundancy, others on fault tolerance, others on modularity and repair-ability, etc. By organizing the topics of resilience along defined measurable terms, this research helps establish a common understanding of resilience from which further research can be conducted.

6.3. FUTURE WORK RECOMMENDATIONS

The conduct of this research identified several areas where improvements and advancements remain.

First, this work focused on the survival phase. The avoidance and recovery phases are important and an end-to-end approach to evaluating resilience will assist system developers. This work could be extended in the recovery phase by examining the how systems return to normal performance, either via change of state, or via a re-starting process. One approach to examine recovery is using state-space analysis. State space analysis is already supported by Petri nets, and would be an excellent extension. Since one method used by many systems to account for failures and disruptions is the ‘reboot’, one could look at potential disruption states, and then determine if a path existed back to a home state, as defined by a Petri Net model of the system. State space analysis could also be used to identified degraded states from which recovery is not possible. These are termed “absorbing states” and architects could evaluate the Petri net based architecture to eliminate such potential states. Additionally, where graceful degradation examined rate of departure, an analogous metric for recovery could be ‘rate of return’, measured in a similar way to rate of departure.

Second, this research examined resilience within the boundary conditions of the field of Command and Control architectures. Further research is needed to determine which portions of this research could be applied to other fields, specifically to hardware related fields.

Third, the structural analysis techniques used in this research make the assumption that each information flow path (simple functionality) is equally important and do not discriminate between the length or composition of each information flow path. Further research could be done to assess how relative importance between information flow paths affects these results. Additionally, further research could examine whether the length of a simple information flow path affects each appropriate measure used in this research.

Finally, further work could be done to support automatic generation of Petri net architecture based off of BPMN and other formats. Liles [2008] demonstrated automatic generation of Petri net models from UML architecture descriptions. However, no automatic tools exist to generate Petri net forms of BPMN. Given the prevalence of BPMN in DoD architectures, such a tool would be of great value over the existing BPMN simulation tools, which are not underpinned by a formal mathematical model.

REFERENCES

REFERENCES

Arsenault, D. and Sood, A., 2007. *Resilience: A Systems Design Imperative*, CIPP Working Paper 02-07. Arlington, VA: George Mason University.

Ashby, M., Shercliff, H., & Cebon D, 2007. *Materials: engineering, science, processing and design.* Oxford, UK: Butterworth-Heinemann Elsevier.

Axelrod, C.W., Investing in Software Resiliency, *CROSSTALK: The Journal of Defense Software Engineering,* September/October 2009, pp.20-25.

Booch, G., Rumbaugh, J., & Jacobson, I., 1998. *The Unified Modeling Language User Guide,* Reading, MA: Addison Wesley, First Ed.

Bouthonnier, V., & Levis, A., 1984. Effectiveness of C^3 Systems, *IEEE Transactions on Systems, Man, and Cybernetics* Vol SMC-16, No 14.

Bruneau, M., and Reinhorn, A., *Overview of the Resilience Concept* [Online] Available at: mceer.buffalo.edu/research/resilience/resilience_10-24-06.pdf, [Accessed 30 January 2011].

Carpenter, S., Walker, B., Anderies, J., & Abel, N., 2001. From Metaphor to Measurement: Resilience of What to What?, *Ecosystems*, vol 4, pp. 765–781.

CBS News, *Iran Confirms Stuxnet Worm Halted Centrifuges* [Online] Available at: http://www.cbsnews.com/stories/2010/11/29/world/main7100197.shtml [Accessed 25 January 2011].

Cameron, A., *GPS World: Perspectives - June 2008* [Online] Available at: http://www.gpsworld.com/gnss-system/perspectives-june-2008-7254 [Accessed 19 August 2011]

Chartrand, G., 1997. *Graphs as Mathematical Models*, Belmont, CA: Wadsworth.

Chartrand, G., & Lesniak, L., 1986. *Graphs and Digraphs* 2nd Ed, Monterey, CA: Wadsworth.

Commodity Futures Trading Commission and the Securities and Exchange Commission, 2010. *Findings Regarding the Market Events of May 6th, 2010, Report of the*

Staffs of the CFTC and the SEC to the Joint Advisory Committee on Emerging Market Issues.

Cormen, T., Leiserson, C., & Rivest, R., 1990. *Introduction to Algorithms*, New York, NY: McGraw-Hill.

Cothier, P., & Levis, A., 1986. Timeliness and Measures of Effectiveness in Command and Control, *IEEE Transactions on Systems, Man, and Cybernetics* Vol SMC-16, No 6.

Department of Commerce, National Telecommunications and Information Administration, Institute for Telecommunications Services, *Federal Standard 1037C*, [Online] Available at: http://www.its.bldrdoc.gov/fs-1037/ [Accessed June 2011].

Department of Defense (DoD), *DoD Architecture Framework (DoDAF) version 2.0*, May 28, 2009. [Online] Available at: http://cio-nii.defense.gov/sites/dodaf20/ [Accessed January 2010].

Department of Homeland Security, *2009 National Infrastructure Protection Plan (NIPP)*, [Online] Available at: http://www.dhs.gov/xlibrary/assets/NIPP_Plan.pdf [Accessed March 2010].

Department of Homeland Security, *Homeland Security Presidential Directive (HSPD) 7*, [Online] Available at: http://www.dhs.gov/xabout/laws/gc_1214597989952.shtm [Accessed March 2010]

Damaël, J. J., & Levis, A. H., 1994. On Generating Variable Structure Architectures for Decision Making Systems, *Information and Decision Technologies*, vol. 19, pp. 233-255.

Dijkman, R., Dumas, M., & Ouyang, C., 2008. Formal Semantics and Analysis of BPMN Process Models using Petri Nets, *Information and Software Technology*, November, 2008, Volume 50, Issue 12, pp. 1281-1294.

Farley, T.R., & Colbourn, C. J., 2007. Multiterminal Resilience for Series-Parallel Networks, *Networks*, pp. 164-172.

Farley, T.R., & Colbourn, C. J., 2009. *Multi-Terminal Measures for Network Reliability and Resilience*," In: *7th International Workshop on the Design of Reliable Communication Networks*, 2009, pp. 107-114, Washington, DC.

Farkas, J., 1902. Theorie der einfachen Ungleichungen, *Journal Fur reine und angew, Mathematik* (124).

Federal Emergency Management Agency, 2000. *FEMA-351, Recommended Seismic Evaluation and Upgrade Criteria for Existing Welded Steel Moment-Frame Buildings*.

Friedenthal, S., Moore, A., & Steiner, R., 2008. *A Practical Guide to SysML The Systems Modeling Language*, Burlington, MA: OMG Press.

Girault, C., & Valk, R., 2003. *Petri Nets for Systems Engineering*, Berlin, Germany: Springer.

Grassman K., & Tremblay, J., 1996. *Logic and Discrete Mathematics: A Computer Science Perspective*, Upper Saddle River, NJ: Prentice Hall.

Hanzálek, Z., Implementation of the Martinez and Silva Invariant Generation Algorithm in a MATLAB m file (silva.m) [Online] Available at: http://www.mathworks.com/matlabcentral/fileexchange/6501 [Accessed May 2010]

Hillion, H., 1986. *Performance Evaluation of Decisionmaking Systems Using Timed Petri Nets*, MS Thesis, Dept of Mechanical Engineering, Massachusetts Institute of Technology, Boston, MA, August 1986.

Hollnagel, E., Woods, D., Leveson, N., 2006. *Resilience Engineering: Concepts and Precepts*. Hampshire, England: Ashgate.

Hopson, D.J., *Addicted to Satellites? Air Force Searches For Alternatives to GPS*, [Online] Popular Mechanics, March 4th, 2010, Available at: http://www.popularmechanics.com/technology/military/satellites/4343983 [Accessed 20 August 2011]

IEEE Standard Glossary of Software Engineering Terminology, IEEE Standard 610.12, 1990

INCOSE, *Resilient Systems Working Group* [Online] Available at: http://www.incose.org/practice/techactivities/wg/rswg/ [Accessed September 2009]

ISO/IEC Systems and software engineering 2007. *Recommended practice for architectural description of software-intensive systems*, ISO/IEC 42010.

Jackson, S., 2010. *Architecting Resilient Systems: accident avoidance and survival and recovery from disruptions*. Hoboken, NJ: Wiley Series in Systems Engineering and Management.

Jensen, K., & Kristensen, L.M., 2007. *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*, Berlin, Germany: Springer.

Jet Propulsion Laboratory, *Report on the Loss of the Mars Polar Lander and Deep Space 2 Missions*, JPL Special Review Board, JPL D-18709, 22 Mar 2000, [Online] Available at: spaceflight.nasa.gov/spacenews/releases/2000/mpl/mpl_report_1.pdf [Accessed April 2011]

Kansal, S.K., AbuSharekh, A.M., & Levis, 2007. A.H., Computationally Derived Models of Adversary Organizations, In: *Proc. IEEE Symp. on Computational Intelligence for Security and Defense Applications,* April 2007, Honolulu, HI.

Karoly, P., Ruehlman, L., 2006. Psychological "Resilience" And Its Correlates In Chronic Pain: Findings From A National Community Sample, *Pain*, 123(1-2), pp. 90-97.

Kemeny, J., et al, 1979. *Report of the President's Commission on the Accident at TMI* [Online] Available at: http://www.threemileisland.org/downloads/188.pdf [Accessed November 2010]

Knight, J.C., & Sullivan, K.J., *On the Definition of Survivability* [Online] Available at: www.cse.msu.edu/~cse870/Materials/FaultTolerant/john.dsn.pdf Department of Computer Science, University of Virginia [Accessed May, 2011].

Leveson, N., 2003. *The Role of Software in Spacecraft Accidents*, MIT, [Online] Available at: mit.dspace.org/bitstream/handle/1721.1/35848/16.../0/jsr.pdf [Accessed January 2011].

Levin, A., Copeland, L., 2009. Delays ripple across country, *USA Today*, 20 Nov, 2009

Levis, A. H., 2008. Systems 621: System Architecture Design Course Notes. George Mason University.

Liles, S., 2008. *On The Characterization and Analysis of System of Systems Architectures* PhD Dissertation, Dept of Information Technology and Engineering, George Mason University, Fairfax, VA, August 2008.

Madni, A., Jackson, S., 2009. Towards a Conceptual Framework for Resilience Engineering, *IEEE Systems Journal*, Special issue on Resilience Engineering, No 2, pp. 181-191.

Martinez J., & Silva, M., 1982. A simple and fast algorithm to obtain all invariants of generalized Petri nets, *Informatik-Fachbrichte* 52, Springer-Verlag, pp. 301 – 303.

Maier, M. W., & Rechtin, E., 2009. *The Art of Systems Architecting*. Boca Raton, FL: CRC Press / Taylor and Francis Group, 3rd Ed.

Mitchell, R., Auld, M., Le Duc, M., & Marrs, R., 2000. Ecosystem stability and resilience: a review of their relevance for the conservation management of lowland heaths, *Perspectives in Plant Ecology, Evolution and Systematics*, vol 3/2, pp. 142-160.

National Transportation Safety Board, *Accident Report NTSB/AAR-10/03*, May 4, 2010, [Online] Available at: https://ntsb.gov/Publictn/2010/AAR1003.pdf [Accessed June 2011]

Naval Research Advisory Council, 2005. *Lighter-Than-Air Systems for Future Naval Missions*, [Online] October 4th, Available at: http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA444479&Location=U2&doc=GetTRDoc.pdf [Accessed 24 August 2011]

Markoff, J., & Sanger, D.E., 2010. *In a Computer Worm, a Possible Biblical Clue*. *New York Times*, 29 September 2010.

Nicol, D. M., 2011 Hacking the Lights Out, *Scientific American*, vol 305, no 1, July 2011, pp70-75.

Nilchiani, R., 2005 *Measuring Space Systems Flexibility: A Comprehensive Six-element Framework*, PhD Dissertation, Department of Aeronautics and Astronautics, Massachussettes Institute of Technology, Boston, MA, September 2005.

Object Management Group, *Business Process Model and Notation (BPMN) Specification Version 2.0*, June 2010, [Online] Available at : http://www.omg.org/spec/BPMN/2.0 (Accessed: January 2011).

O'Rourke, T., 2007. Critical Infrastructure, Interdependencies, and Resilience, *The Bridge*, National Academy of Engineering, Vol 37, No 1.

Papazoglouand, M. P., & van den Heuvel, W.J., 2007. Service oriented architectures: approaches, technologies and research issues, *VLDB Journal*, 16, pp. 389-415.

Pappalardo, J., 2009. Will Obama Kill Navigation Backup System as GPS Threatens to Fail? *Popular Mechanics*, December 18, [Online] Available at: http://www.popularmechanics.com/technology/military/satellites/4318471 [Accessed 19 August 2011].

Plaut, R., 2008. Snap loads and torsional oscillations of the original Tacoma Narrows Bridge, *Journal of Sound and Vibration*, Vol 309, pp. 613-636.

Raedts I., et al, 2007. Transformation of BPMN models for Behaviour Analysis, Modelling, Simulation, Verification and Validation of Enterprise Information Systems, In: *Proceedings of the 5th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems, MSVVEIS-2007*, In conjunction with INSTICC Press, 2007.

Rash, G., *GPS Jamming in A Laboratory Environment* Naval Air Warfare Center Weapons Division (NAWCWPNS)/China Lake. [Online] Available at: www.fas.org/spp/military/program/nav/labjam.pdf [Accessed 20 August 2011]

Ray-Chaudhuri, S., & Shinozuka, M., 2010. Enhancement of Seismic Sustainability of Critical Facilities Through System Analysis, *Probabilistic Engineering Mechanics*, Vol. 25, Issue 2, April, pp. 235-244.

Rausand, M., & Hoyland, A., 2004. *System Reliability Theory: Models, Statistical Methods, and Applications*, Hoboken, NJ, Wiley Series in Probability and Statistics.

Reason, J., 1988. Errors and Evaluations: the Lessons of Chernobyl, In: *Conference Record for 1988 IEEE Fourth Conference on Human Factors and Power Plants*, 5-9 June 1988, pp. 537-540.

Reed, D. A., Kapur, K. C., & Christie, R. D., 2009. Methodology for Assessing the Resilience of Networked Infrastructure, *IEEE Systems Journal*, 3(2) pp. 174-180.

Remy, P. A., Jin, V. Y., & Levis, A. H., 1988. On the Design of Distributed Organization Structures, *Automatica*, 24(1).

Resilience Alliance, *Key Concepts*, [Online] Available at: http://www.resalliance.org/index.php/key_concepts [Accessed November 2010].

Resilient Engineering Network, *What is resilience?* [Online] Available at: http://www.resilience-engineering.org/faq.htm [Accessed November 2010].

Stahl, C., 2005. *A Petri net semantics for BPEL*, Berlin Professoren des Inst. fürInformatikInformatik-Berichte, Nr. 188, Berlin.

Sterman, J., 2000. *Business Dynamics: Systems Thinking for a Complex World*, Boston, MA: Irwin/McGraw-Hill.

Svádová, M., & Hanzálek, Z. *MATLAB ToolboxForPetri Nets*, Center for Applied Cybernetics, DCE FEE, Czech Technical University in Prague. [Online] Available at: citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.94.658&rep [Accessed 19 March 2011]

Tang, A., Han, J., & Chen, P., 2004. *A Comparative Analysis of Architecture Frameworks*, Swinburne University of Technology and Australian DSTO, Department of Defence Technical Report: SUTIT-TR2004.01, CeCSES Centre Report: SUT.CeCSES-TR001, 25 August 2004.

Tavana, M., Busch, T.E., & Davis, E.L., 2011. Modeling Operational Robustness and Resiliency with High-Level Petri Nets, *International Journal of Knowledge-Based Organizations*, vol 17 April-June 1(2), pp. 17-38.

Tuscon Sentinel, D. Smith, Aerostat balloon crashes in Sierra Vista, May 9, 2011, Available at: http://www.tucsonsentinel.com/local/report/050911_aerostat/aerostat-balloon-crashes-sierra-vista/

US Navy Public Affairs, 2009. US 4th Fleet, Specialist 2nd Class Alan Gragg, *U.S. 4th Fleet Stands Up Maritime Operations Center*, 26 March 2009, [Online] Available at: http://www.navy.mil/search/display.asp?story_id=43775 [Accessed August 2011].

US Navy, Office of Naval Research, *Scalable Adaptive Architectures for Maritime Operations Center Command and Control*, ONR Award No: N00014-08-1-0319 to George Mason University Systems Architectures Laboratory.

United States Government Accountability Office, 2009. *Global Positioning System: Significant Challenges in Sustaining and Upgrading Widely Used Capabilities*, May 7th, 2009 [Online] Available at: http://www.gao.gov/new.items/d09670t.pdf [Accessed 20 August 2011]

F. Valraud and A. Levis, *On the Quantitative Evaluation of Functionality in C3 Systems*, AFCEA International Press, pp132-139, August 1989.

Wagenhals, L. W., & Levis, A. H., 2009. Service Oriented Architectures, the DoD Architecture Framework 1.5, and Executable Architectures, *Systems Engineering*, 12(4) pp. 312-343.

Wagenhals, L. W., Haider, S., & Levis, A. H., 2003. Synthesizing Executable Models of Object Oriented Architectures, *Systems Engineering*, 6(4).

Waugh, C., Fredrickson, B., Taylor, S., 2008. Adapting To Life's Slings And Arrows: Individual Differences In Resilience When Recovering From An Anticipated Threat, *Journal of Research in Personality*, 42(4), pp. 1031-1046.

Weske, M., 2010. *Business Process Management: Concepts, Languages, Architectures*, Berlin, Germany: Springer-Verlag.

Westrum, R., 2006. *A Typology of Resilience Situations*, appearing in: E. Hollnagel, D Woods, N Leveson, Resilience Engineering: Concepts and Precepts. Hampshire, England: Ashgate.

White, S., & Miers, D., 2008. *BPMN: Modeling and Reference Guide*, Lighthouse Point, FL: Future Strategies Inc.

Woods, D., 2006. *Essential Characteristics of Resilience*, appearing in: E. Hollnagel, D Woods, N Leveson, Resilience Engineering: Concepts and Precepts. Hampshire, England: Ashgate.

Yourdon, E., & Contantine, L., 1979. *Structured Design*, Englewood Cliffs, N.J: Prentice-Hall, Inc.

Zaidi, A., 1994. *Validation and Verification of Decision Making Rules*, PhD Dissertation, Dept of Information Technology and Engineering, George Mason University, Fairfax, VA.

Zio, E. 2009. Reliability Engineering: Old problems and new challenges, *Reliability Engineering and System Safety*, vol 94, pp. 125-141.

CURRICULUM VITAE

Mark Pflanz graduated from Valley High School, West Des Moines, Iowa, in 1992. He received his Bachelor of Science from United States Military Academy at West Point in 1996. He served as an infantry officer in the United States Army for five years. Following service in the Army, he worked briefly for International Paper. He has worked as a consultant at Booz Allen Hamilton since 2002. In 2005, he received his Master of Science in Systems Engineering at Virginia Polytechnic Institute and State University.