MANAGING THE IMPACT OF CHANGE THROUGH SURVIVABILITY AND PLIABILITY TO ACHIEVE VIABLE SYSTEMS OF SYSTEMS

by
Brian Mekdeci

Bachelor of Systems Design Engineering, University of Waterloo (2002)
Master of Systems Design Engineering, University of Waterloo (2005)

Submitted to the Engineering Systems Division in Partial Fulfillment of the Requirements for the Degrees of

Doctor of Philosophy in Engineering Systems
at the
Massachusetts Institute of Technology

February 1, 2013

Signature of Author_____
Engineering Systems
February 1, 2013

Certified by_____
Daniel E. Hastings
Dean for Undergraduate Education,
Cecil and Ida Green Education Professor of Aeronautics and Astronautics and Engineering Systems,
Thesis Committee Chair

Certified by_____
Donna H. Rhodes
Principal Research Scientist and Senior Lecturer, Engineering Systems, Director, SEAri
Thesis Committee Member

Certified by_____
Adam M. Ross
Research Scientist, Engineering Systems, Lead Research Scientist, SEAri
Thesis Committee Member

Certified by_____
Daniel Frey
Professor of Mechanical Engineering and Engineering Systems,
Co-Director, Singapore-MIT International Design Center
Thesis Committee Member

Accepted by_____
Oliver L. de Weck
Associate Professor of Aeronautics and Astronautics and Engineering Systems
Chair, Engineering Systems Division Education Committee

PAGE LEFT INTENTIONALLY BLANK

MANAGING THE IMPACT OF CHANGE THROUGH SURVIVABILITY AND PLIABILITY TO ACHIEVE VIABLE SYSTEMS OF SYSTEMS

by

Brian Mekdeci

Submitted to Engineering Systems Division on February 1, 2013 in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Engineering Systems

ABSTRACT

As technology improves, traditional systems are being interconnected into larger systems of systems (SoS) that operate in diverse contexts, where numerous perturbations exist that threaten the ability of the SoS to deliver acceptable value to its diverse set of stakeholders. Furthermore, the systems of systems themselves can change form voluntarily or involuntarily in response to contextual variability or stakeholder whims. Various system properties, or "-ilities" have been defined that may help traditional systems provide value to stakeholders in spite of change, but they have not specifically addressed the issue of systems operating within larger systems of systems. This dissertation defines the concept of viability for engineered systems, as a likelihood that systems will satisfy their stakeholder needs over the system's expected lifetime, and identifies and develops strategies that system architects can use to create viable systems. The concept of viability helps system architects design systems that can survive contextual perturbations, whether they are from entities outside the traditional system boundary, or from other constituent systems within a SoS.

In addition to external perturbations, this dissertation addresses the need to ensure that endogenous changes made to improve value delivery, do not inadvertently cause unintended interactions that harm the system overall. This is particularly a concern with the proliferation of systems of systems, and the recent drive towards making systems more changeable as a mechanism for value sustainment in dynamic environments. A new "ility", pliability, is introduced that specifies the limits on how a system can change, without "breaking" or violating an architecture that was intended and validated. Like changeability, pliability increases robustness by allowing systems to voluntarily change in response to dynamic contexts, and increases survivability and robustness by increasing the likelihood that unintentional changes are still within the set of allowable instances. It also distinguishes allowable changes from those that would require validation, reducing the effort required to get those changes approved by a diverse set of stakeholders.

Thesis Chair: Daniel E. Hastings

Title: Dean for Undergraduate Education, Cecil and Ida Green Education Professor of Aeronautics and Astronautics and Engineering Systems

This dissertation is dedicated to my parents.

ACKNOWLEDGEMENTS

This dissertation is dedicated to my parents, Johanna Mekdeci and Dr. Anthony Mekdeci, who always loved and supported me. A special acknowledgement should be made to my mother, who risked her life so that I could be born. Words cannot express my gratitude or my love. Of course, I could not have done this without the support of other family members including my brother John Mekdeci, sister Zerlene Mekdeci, brother-in-law Neil Appalsamy, niece Carmen Appalsamy, nephew Tristen Appalsamy, uncle Etienne Mekdeci, cousin Dr. Philip Teixeira, and aunt Rita Teixeira. I would also like to acknowledge my extended family for their encouragement, including Michael Sheedy, Lorraine Sheedy, John Clarke, Caroline Clarke, Betty Clarke, John Byrne, David Byrne, Mary Byrne, Frank Byrne, and all my other cousins, aunts and uncles in the Republic of Ireland.

I would like to thank my advisor and committee chair, Dr. Daniel E. Hastings, for guiding me and helping me reach my potential. Prof. Hastings has both the personal and professional attributes that I strive for, and I will always consider it a tremendous honor to have studied under his tutelage. This dissertation would not have been possible without the financial support of the Systems Engineering Advancement Research Initiative (SEAri) at MIT, and in particular, committee members Dr. Donna Rhodes and Dr. Adam Ross. Dr. Donna Rhodes deserves special thanks for providing unwavering support, professionally and personally, through every trial and tribulation I incurred along the way. I would also like to give thanks and a special acknowledgement to Dr. Adam Ross, who I looked up to (even though he is younger than me). Dr. Ross is wise beyond his years, and I was truly blessed to have been able to spend at least an hour with him every week discussing research, current events, and anything else that was on my mind. I will truly miss those meetings. I would like to thank committee member Prof. Dan Frey, who taught me ESD.86, and provided critical feedback on the quantitative methods used in this dissertation. I would also like to thank Prof. Richard de Neufville, who arranged and sponsored my trip to Singapore in the summer of 2011. That trip was a tremendous personal and professional experience, and I am grateful to Prof. de Neufville, Prof. Magee and Prof. Frey for making that happen. I am also grateful to Prof. Ollie de Weck, who helped arrange the ESD Graduate Fellowship I received during my last term at MIT. Finally, I would like to thank Prof. Joe Sussman, who supported me at every important moment of my PhD experience at MIT, from my first class in ESD 83, through my General Exams, and finally at my doctoral dissertation defense.

Beth Milnes, the Academic Administrator for Engineering Systems Division (ESD), deserves a special mention. For nearly three years, I was able to take my dog Tuco into MIT with me every day only because Beth looked after him in her office. Tuco was very important to me and Beth made it possible for me to keep Tuco while I pursued my PhD. I would often engage in long

conversations with Beth, sometimes on a daily basis, and I consider her the best friend I had at MIT. I wish her nothing but the best.

A very special acknowledgement goes to my girlfriend Camille McNeill, who loved and supported me during the final, critical months of my dissertation, and made me feel like anything was possible. I love you, Camille. I would also like to acknowledge her parents Lori Muncy and Zane McNeill, as well as her sister Marlene McNeill, who treated me like family and never made me feel like a stranger even though I was thousands of miles from home. I would also like to thank Thomas Leung for his support and friendship over the years. Thomas will always be my best friend, and truly one of the best human beings walking this earth.

I would like to acknowledge the teachers, professors, and other professionals who either directly trained me, or inspired me in some way. Some of those that deserve special recognition include Dr. Ed Jernigan, Dr. Catherine Burns, Dr. Debbie Nightingale, Dr. John Carroll, Dr. Chris Magee, Dr. Michael De Robertis, Mr. Pesando, Mr. Allen, Miss Mann, Tom Haapanen, Dave Weiler and Albert Sulmistras.

A final thanks goes to all my friends, not previously mentioned, who supported me or influenced me along the way. Although it would be impossible to list every name, but some of those that come to mind include Dr. Daniel Molke, Aaron St. Bernard, Dave Scarth, Matt Kaciak, Dr. Michael Kadour, Geoff Carrigan, Christin Hart, Dr. Michel Cardin, Dr. Yves Boussemart, Dr. Carl Nehme, Yale Song, Farzan Sasangohar, Dr. Nirav Shah, Dr. Jeff Crandall, Dr. Luca Bertuccelli, Laura Campbell, Dr. Birsen Donmez, Jazelle Heng, Jane Ng, Micheka Caskanette, Bob Caskanette, Kate Caskanette, Rene Caskanettte, various individuals at St. Michaels Choir School, the sportsandwrestling message board. Annunciation, Senator O'Connor, and Mr. Grocers Underhill. Last but not least, I would like to thank Aixa Almonte not only for her friendship, but also for her assistance designing some of the graphics that appeared in the discrete event simulation used in this dissertation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AOI	Area of Interest
DHS	Department of Homeland Security
DDM	Design Decision Maker
DM	Decision Maker
EMS	Emergency Medical Services
ERS	Emergency Response Services
FEMA	Federal Emergency Management Agency
FoS	Family of Systems
ISO	International Standards Organization
ISP	Internet Service Provider
JF	Joint Force
JTF	Joint Task Force
MIT	Massachusetts Institute of Technology
MarSec	Maritime security System of Systems
ODM	Operational Decision Maker
OUAV	Observational Unmanned Aerial Vehicle
PS3	PlayStation 3
PSN	PlayStation Network
RQ	Research Question
SA	System Architecture
SE	Systems Engineering
SDM	Strategic Decision Maker
SLA	Service Level Agreements
SP	Service Provider
SoS	System of Systems
TUAV	Tactical Unmanned Aerial Vehicle
UAV	Unmanned Aerial Vehicle
VSM	Viable Systems Model

1 INTRODUCTION

"Change is the law of life, and those who look only to the past or the present are certain to miss the future."

- John F. Kennedy

1.1 MOTIVATION FOR THE RESEARCH

Although some of the basic concepts of Systems Engineering (SE) can be traced back to ancient times, formal research and instruction in the field began in the 1940s and 1950s at institutions such as Bell Laboratories and MIT (Brill, 1998; Hall, 1962). In its infancy, SE was typically practiced by engineers trained in other disciplines (such as mechanical or electrical engineering), who needed to combine individual components into holistic systems, so that higher-level functionality could be realized. As stakeholder needs increased, so too did the level of complexity of the systems they formed, and researchers began developing system engineering methods and techniques to overcome the numerous challenges that arose. Over the years, these methods and techniques proved to be successful in numerous major projects, such as missile defense and the Apollo missions, and subsequently SE became a distinct engineering discipline, with graduate and undergraduate degree programs worldwide, dedicated conferences and journals, and ardent practitioners spanning a diverse set of industries.

1.1.1 THE RISE OF SYSTEMS OF SYSTEMS

Most systems designed in the last century were *traditional* in the sense that they consisted of components that were not very useful on their own, but interacted in such a way that they produced a higher-level functionality that was useful. In the last decade, technology (particularly communication technology) has improved significantly, allowing traditional systems to be connected together to form larger, more complex systems of systems (SoS). For example, as recently as 2004, standalone video game consoles, such as the Sony PlayStation 2, were state-of-the-art in the consumer electronics market. Although these consoles had powerful graphics processors, multi-functionality (e.g. the ability to play DVD movies as well as games), and innovative user interface devices, they were essentially just incremental improvements on traditional video game systems that have been in homes since the late 1970s. However, starting with the seventh generation in 2005 (Long, 2012), video game consoles began to be inter-connected into massive systems of systems that allowed gamers and vendors from all over the world to play and interact with each other creating a virtual marketplace worth billions of dollars. In order for this to happen, however, system engineers had to create a

suitable network infrastructure including multitudes of servers that authenticate and match users for games, deliver premium streaming content such as movie rentals, and process credit card transactions between customers and vendors. Additionally, since these systems of systems cross international boundaries, there are non-trivial social, legal and contextual issues that need to be addressed in order for the overall system to be successful.

While video game consoles have always been at the cutting-edge of consumer electronics, even traditional home appliances are being connected together to form systems of systems. As seen in the Consumer Electronics show of 2012, some "smart" appliances can regulate their energy use via a connection to a "smart grid" energy infrastructure that consists of other power-consuming devices and power suppliers. This means that instead of an appliance acting as a stand-alone system with its user, it can regulate its energy usage based on the usage of other devices attached to the grid, and the cost of energy at any given time. Some new appliances also boast Internet connectivity where users can interact with third party vendors and their appliances to do online grocery shopping, for example (LG Electronics, 2012). Of course, systems of systems are not limited to home appliances and entertainment devices. The proliferation of systems of systems has also extended into industries such as health care information systems, banking and critical infrastructure. Of course, the US military has been combining mechanized arms into larger Joint Task Forces and other similar systems of systems for years, but recent efforts, such as the US Army modernization program, highlight that the drive is continuing, if not accelerating.

> *The Army's new information network remains our top modernization priority. With tighter budgets but an active threat environment, the Army will have to produce a force that is smaller yet still highly capable. The Network is the core of that smaller but highly capable force.*
>
> *Secretary of Army John McHugh (2012)*

The new information network that the US Army is developing is a massive SoS that extends right down to the soldier level. The US Army's official website describes the Army's new mobile network as follows:

> *The Army's mobile network will enable Soldiers to access key information anytime, anyplace; share information to facilitate fire and maneuver, and survive in close combat; provide collaboration capability to aid in seizing and controlling key terrain; employ lethal and non-lethal capabilities, coupled with sensors, to effectively engage targets at extended ranges; distinguish among friend, enemy, neutral and noncombatant; and integrate indirect fires.*
>
> *("Networking the Soldier," 2012)*

1.1.1.1 Factors Contributing to the Rise of Systems of Systems

Three factors are driving the rise of systems of systems, as shown in Figure 1-1. First, technology improvements, particularly in data storage, manipulation and transmission, make inter-connection of independent systems into larger systems of systems technically feasible. Reduction in costs, particularly manufacturing, computing and bandwidth, make these capabilities financially feasible as well. The combination of these two factors leads to these new capabilities being available as options for particular SoS designs. As these options become available, stakeholder expectations rise as a result. The increase in stakeholder expectations drives research and development to increase technology levels and decrease costs even more, enabling new capabilities to emerge and raising stakeholder expectations even further. While the proliferation of systems of systems are creating opportunities for users and stakeholders that were unheard of decades ago, it is also creating significant challenges for the system engineers and architects that are responsible for designing, building and testing them. Unfortunately, SoS techniques are not developing as quickly as the systems themselves.

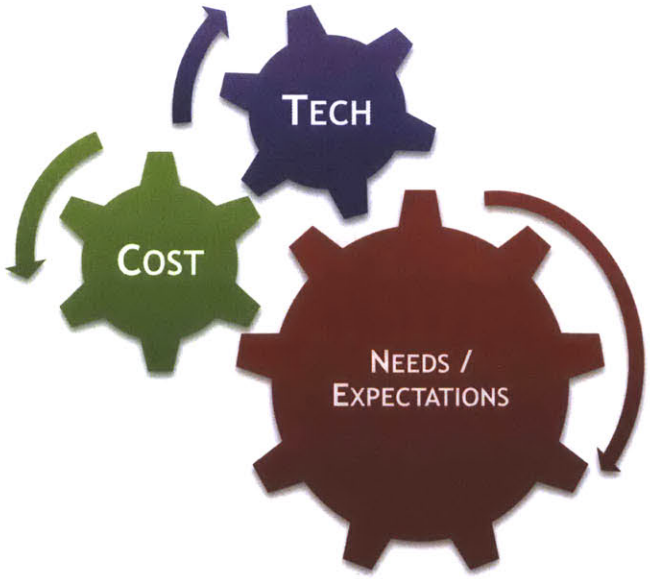

FIGURE 1-1: FACTORS INFLUENCING THE RISE OF SYSTEMS OF SYSTEMS

1.1.2 SoS Engineering Challenges

As illustrated in Table 1-1, when traditional systems are interconnected into larger systems of systems, the SoS itself is often quite different from its constituent systems in many ways. Some might argue that the number of components in a SoS is at least the sum of the number of components that make up in each of its constituents systems. Thus, a SoS always has more components that the systems they are made up of, if "sub-components" are counted. Even if

"sub-components" are not counted as components, many systems of systems scale to tremendous sizes, particularly if co-location of components is not required. Instead of being bound to physical locations and enclosures, the constituent systems of many systems of systems are geographically separated. This often makes synchronizing and coordinating tasks more difficult than a traditional, hardwired system, but allows for the SoS to influence (and be influenced by) many different environments. Different manufacturers may also have made the constituent systems at different periods of time and under different specifications, creating potential interoperability issues. The operational and managerial independence of the constituent systems also adds significantly to the complexity of the overall SoS. The system response, i.e. the outputs given a particular set of inputs, is in many cases non-deterministic and subject not only to stochastic natural processes, but also to real-time choices made by operational and strategic decision makers. The social aspect of many systems of systems, can add a considerable amount of complexity above the technological aspects necessary for the SoS to work. For example, Facebook consists of not only millions of diverse devices communicating and coordinating their actions, but also millions of human users interacting as well. The actions of the humans change the experience for other users on Facebook, for better or for worse, and introduce uncertainty into the overall benefit of the SoS to its stakeholders. In systems of systems, there are often many stakeholders with diverse interests that influence the design and operation of the SoS throughout its entire lifecycle. For example, a Joint Task force may have units from the Army, Navy and Air Force, each with their own command structures and needs. Furthermore, while in many traditional systems, the human operator is outside the system, they must often be considered within a constituent system while in a SoS.

TABLE 1-1: VIDEO GAME SYSTEM VS. VIDEO GAME SOS

	PlayStation Console (System)	PlayStation Network (SoS)
Lifespan	Several Years	Indefinite
# of Components	Hundreds	Millions
Location	Local	Global
Context	Single	Diverse
Human	Exogenous	Endogenous
Form	Static	Dynamic
Behavior	Easy to Predict	Difficult to Predict
Worth	$100s	$1,000,000,000s

Since system complexity tends to increase with size (Lankford, 2003) systems of systems can be expensive to design, build, test, launch, operate and maintain. With shrinking budgets, particularly in public projects, stakeholders expect critical systems of systems to have long lifetimes. Often the difference in lifespan between a system and a SoS can be due to the fact that systems of systems are often unique, whereas traditional systems tend not to be. A system like an Apple iPhone, for example, is only expected to be useful for a couple of years before it is replaced with something else. There are millions of iPhones in the world today, and like all gadgets, they come and go. There is only one Facebook, however, and even though it is expected to evolve and grow with new technology, partners and services, it is not expected to go away anytime soon. Such is the case with other systems of systems, such as major metropolitan transportation systems, as well. While their constituent systems, such as computers, trains, and even human operators, are expected to come and go, the systems of systems themselves is expected to last indefinitely.

The complexity and extended lifetime of systems of systems presents many challenges to system architects. In addition to having multiple stakeholders, many complex systems of systems operate in diverse contexts, which can become unfavorable at times. The longer a system exists, and the more interactive it is with its environment, the more likely it is that something will change that can threaten the overall value the system provides to its stakeholders. Systems of systems are particularly prone to contextual fluctuations, since they tend to have multiple stakeholders and geographically dispersed components. Changes in context can adversely affect the system and in some cases, render it useless. Sudden, major disruptions can be particularly disastrous, such as the 2011 tsunami that caused the Chernobyl-scale nuclear meltdown in Japan (Figure 1-2) (McNeill, 2011), and the $170 million dollar security breach that shut down the Sony PlayStation Network (Snuder, 2011). Slower contextual changes can have major impacts as well, such as how cell phone technology is gradually making landline phone systems obsolete (Passikoff, 2012). It is clear, then, that a complex system, which is expected to continue to provide value to stakeholders over its entire lifetime, will need to be able to do so even if the context changes inimically.

FIGURE 1-2: FUKUSHIMA NUCLEAR DISASTER, JAPAN, 2011

1.1.2.1 VIABLE SYSTEMS OF SYSTEMS

A system architect can develop a feasible system, if it meets functional requirements and satisfies constraints for a particular context. For example, a system architect can design a factory that is feasible, because it combines the efforts of humans and machines to turn raw materials into finished goods that the stakeholders can sell for a profit. However, as discussed in this section, it is not good enough for most complex systems to only "work" for a short period of time and under one particular context. Rather, stakeholders require that their systems continue to be useful over a long period of time and variety of contexts. If the system stops being useful to its stakeholders, we can assume that its very existence is threatened, unless there are some legacy inheritance or contractual obligations preventing it from being discontinued. Thus, the issue for modern system architects is to not only design feasible systems, but also ones that will exist through long periods of time and possibly varying contexts. In other words, stakeholders seem to want *viable* systems (or viable systems of systems, for that matter), instead of just feasible ones. A viable business model is a concept that can be turned into a profitable business that could exist for a period of time and under certain contexts, but what does it mean for an engineered system to be viable and what properties enable viability? Do viable systems have to be robust? Survivable? Autonomous? These issues form the basis of the first research question.

Research Question 1 (RQ1): What does it mean for an engineered system to be viable? How does viability relate to existing "-ilities" already defined?

1.1.2.2 Independence Within a SoS

To illustrate some of the human issues involved with merging traditional systems into system of systems, consider an individual laptop owner interacting with his/her laptop (Figure 1-3). Although the system boundary can be drawn in many different ways, we can say that the system is the laptop and the individual owner is both the stakeholder and the operator of the system. Most modern laptop are essentially von Neumann machines (Von Neumann, 2012) and will produce the same outputs for a given set of inputs. For example, taking any two, identically configured laptops from an assembly line and providing them with the same inputs, will always result in the exact same outputs. Therefore, if the stakeholder understands what his/her needs are, then it should not be difficult for the stakeholder to determine if the laptop will meet them.

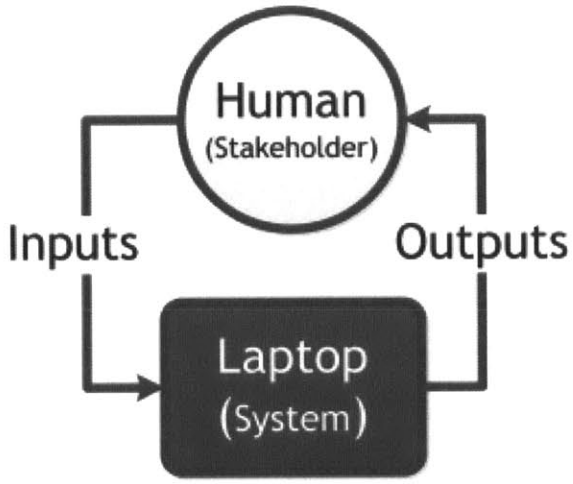

FIGURE 1-3: BLACK BOX REPRESENTATION OF A SIMPLE SYSTEM WITH HUMAN OPERATOR

Although laptops are standalone, traditional systems, they can easily be components in larger systems of systems via a connection to the Internet. Facebook is one such example of a system of systems that consists of millions of people, businesses and electronic devices all interconnected via the Internet. The users of Facebook are on diverse devices, differing in hardware (laptops, cell phones, etc.) and software (Windows, Linux, etc.) and can be considered clients. Although each device is a black box, a traditional system itself similar to that shown in Figure 1-3 from the perspective of the SoS, the clients that make up the largest portion of Facebook's constituent systems consist of both the device and the user together (Figure 1-4). This is because from the SoS perspective, the behavior, and in particular the decisions, that the human stakeholders controlling the devices make, significantly affect the overall experience for other clients, and thus the overall benefit Facebook provides to its stakeholders. In fact, while the devices are important for the Facebook experience, and their

differences are significant (e.g. Facebook on mobile devices can be a less satisfying experience than the desktop version), the human interaction on Facebook is really the whole point, and the devices, servers and other subsystems are really just tools and components necessary to enable and enhance that interaction.

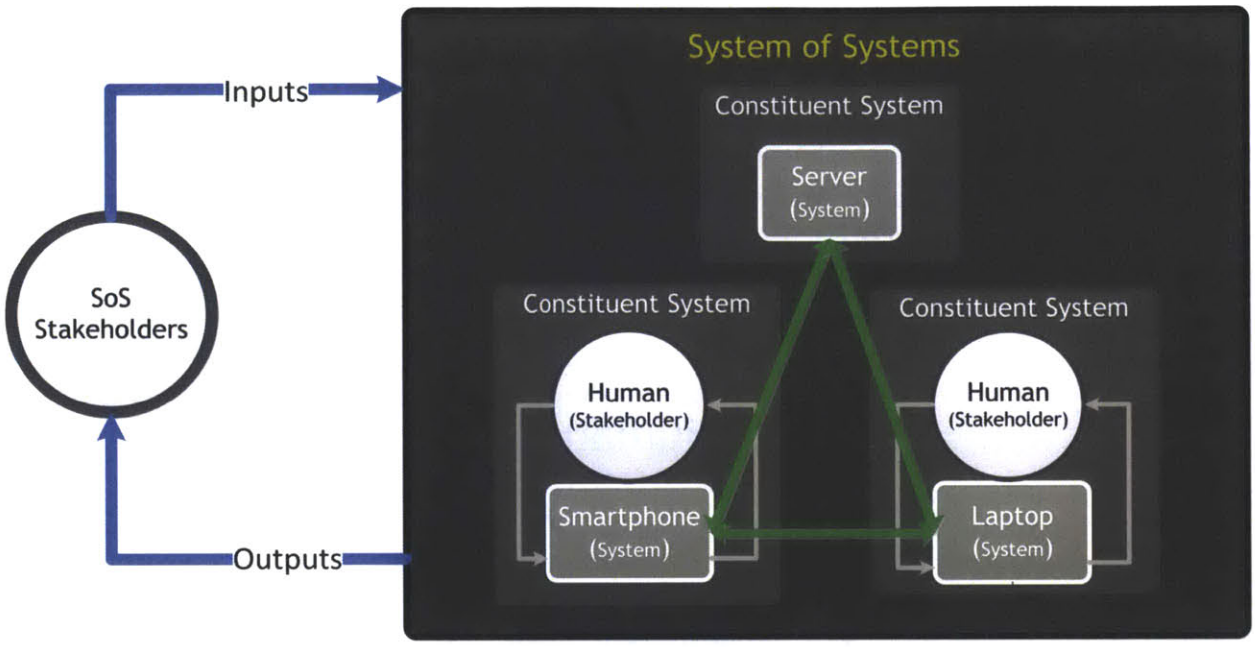

FIGURE 1-4: BLACK BOX EXAMPLE OF A SYSTEM OF SYSTEMS AND ITS STAKEHOLDERS

The human component is an obvious strength, but also a weakness for Facebook. If the user is considered part of the constituent system, then that system no longer acts like von Neumann machine. The outputs, and the behavior of a human interacting with a laptop, are generally not deterministic. Humans, like many constituent systems in other systems of systems, are independent and can choose how they wish to participate in the SoS, and/or whether or not they wish to participate at all. Also, unlike the devices they use to communicate with, no two humans are alike and while their actions may be similar, they are not exactly the same. In fact, the same human may act differently from moment to moment, depending on a variety of factors, such as the weather, his/her mood, what he/she had for breakfast that morning, etc. Unlike the system in Figure 1-3, the stochastic nature makes predicting the behavior of the constituent systems difficult, which creates problems for stakeholders trying to determine whether the SoS will meet their needs. If the owner of a laptop wants to do something, then she just gives the correct inputs and the laptop does it. The owners of the Facebook however, can only exert that kind of control on part of the SoS (such as the servers). As for the independent constituent systems, the SoS owners can only try to influence the behavior of the constituent systems and thus, the overall SoS itself, which is not a trivial matter.

Research into determining how stakeholders with diverse interests and influence can affect the value delivery of a complex SoS (Shah, 2013) is just one of the many research threads currently looking at effective management of systems of systems. However, it is important to understand how the independence of constituent systems and the diversity of the stakeholders affect the viability of the constituent systems as well as of the SoS itself. This forms the basis of the second research question:

Research Question 2 (RQ2): How does viability of constituent systems relate to the viability of systems of systems?

1.1.2.3 Strategies for Enabling Viability

Once system architects understand what properties enable a system that operates in dynamic environments to provide acceptable value to stakeholders, they need to implement these properties into their designs. As with functional requirements, enabling non-functional "-ilities" can be expensive, and typically involves tradeoffs. Decision makers are forced to select between alternatives that balance benefit, cost and risk according to their needs, but in systems of systems, the problem is often compounded due to diverse stakeholders and conflicting risk mitigation strategies (Ellison & Woody, 2007). Systems engineering design principles to aid designing systems in resisting the harmful effects of contextual changes have been identified in the literature (Jackson, 2010; Richards, 2009), but many of the case studies upon which they were developed and validated, involved traditional systems, such as a satellite constellation-based radar system. There has been some debate as to whether traditional systems engineering methods and practices are still valid at the SoS level (Dickerson, 2009). This is a particularly challenging question, since the literature is unclear as to the definition of a system of systems, and how it is distinct from a traditional system (Chattopadhyay, 2008). This is not surprising, since the definition of a "system" itself is also ambiguous (Backlund, 2000). Regardless, systems of systems have certain characteristic properties that are differentiated from traditional systems, and it is important to determine how these qualities might affect its ability to resist the harmful effects of contextual change. This leads to the following research questions:

Research Question 3 (RQ3): Do existing system-level strategies for enabling systems to resist the harmful effects of contextual change apply to systems of systems as well? Are there additional strategies that enable viability in systems, particularly systems of systems?

1.1.2.4 Endogenous Changes

In addition to exogenous contextual changes, systems of systems often experience endogenous change as well. Often, decision makers implement endogenous changes to mitigate the harmful effects of a contextual change, or take advantage of new opportunities. Systems of systems are continuously changing to the point where the activities of the SoS can be thought of as a "wave" (Dahmann, Rebovich, Lowry, Lane, & Baldwin, 2011). In this "wave" model, the SoS goes through various changes, some of which can be made immediately, some of which can be made only after an analysis is performed, while others require a significant re-engineering effort that results in an "evolution" in the system architecture (Figure 1-5).

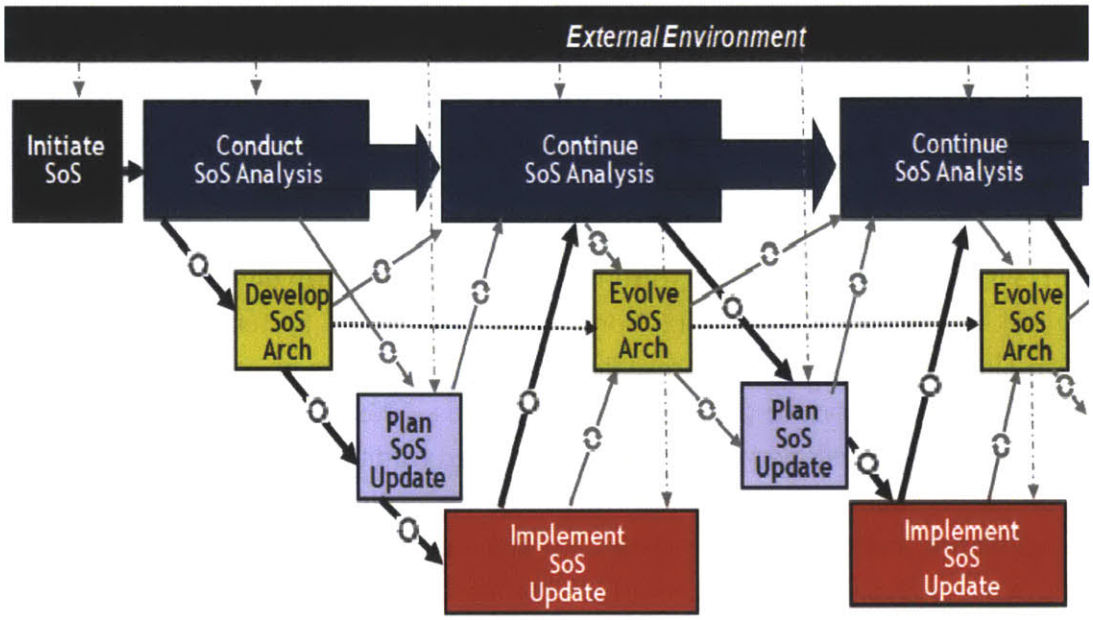

FIGURE 1-5: WAVE MODEL OF SOS ENGINEERING (DAHMANN, ET AL., 2011)

The likelihood that a system will change is higher for systems that have long lifetimes, operate in dynamic contexts, or have complex behavior. Perhaps there will be a change in components, in either the number and type of components, or their attributes and capabilities. This is particularly an issue with systems of systems, since they go through evolutionary development as components are added, removed and changed over time (Maier, 1998). Even if the components themselves do not change, what they do within the overall system might change. For example, certain F-16s, acting as components of a larger military SoS, may change from air-to-air combat roles to air-to-surface attack roles. An emergency response dispatcher may process calls in a priority queuing manner (based on location or some other factor), instead of a first-in, first-out (FIFO) sequence. Of course, changes often do not occur in isolation. A change in components, will often require operational changes as well, and vice-versa. If the wireless

LAN suddenly fails, then mobile system components may be forced to operate near an available Ethernet port. Similarly, a decision to operate certain components in a mobile fashion may require batteries or some other portable power source instead of an AC power supply.

There has been increasing research studying how, when and why systems need to change in response to shifts in context (McManus and Hastings, 2006). Some examples include:

- Designing latent capabilities that grant the ability to change at a later date (e.g., de Neufville and Scholtes, 2011)
- The ability to change during operational phases (e.g., Gupta and Goyal, 1989)
- The ability to change easily / rapidly (e.g., McGaughey, 1999)
- The ability to change in size only (e.g., Elkins et al., 2004)

The terms "flexible", "adaptable", "agile", and "modifiable" are some of the labels that researchers have used to describe the ability to change. Unfortunately, researchers have been inconsistent with their definitions and meaning when applied to systems engineering. This makes it very difficult for stakeholders to communicate the properties that they desire, and for architects to know that they have met those requirements. The ability to change, known as changeability (Ross et al., 2008), is an important quality of a system that allows stakeholders to reduce the impact of uncertainty of future contexts. However, not all changes are beneficial and care must be taken by architects to ensure that any modifications to the components, or the way they operate, will not reduce value delivery to its stakeholders.

While it may seem that the more a system can change, the better it will be able to respond to shifts in context, but there are limits. In their minds, and sometimes in documents, system architects have a concept of operations (CONOPs)[1] that describes how the components within the system function and interact with each other to produce value to the stakeholders within the context of an operational environment (Mekdeci et al, 2011). A change in one of the components, their capabilities, or in the way the components interact or function, may violate an aspect of the CONOPs, and cause a reduction in value delivery. It is important then to only change what should be changed, and not change what should not. For this reason, there may be a significant bureaucratic hurdle in getting changes "approved" before they are allowed in a complex, critical system of systems. While the benefits of having the ability to change has been demonstrated numerous times in the literature, what has been lacking is a property that architects can use to set boundaries around the change that a system is allowed to undergo.

[1] To some researchers and practitioners, the word "CONOPs" refers to *documentation* that describes how a system works, rather than the *concept* itself of how the system works.

This is important so that decision makers can make certain changes to the SoS without the need to re-architect the system and get approval from stakeholders. The fourth research question addresses this issue.

Research Question 4 (RQ4): Is changeability always a good thing? What can system architects do to enable positive change in complex systems, while restricting change that can be harmful?

1.2 Research Questions

The research questions posed in this chapter are repeated below for reference:

RQ1: What does it mean for an engineered system to be viable? How does viability relate to existing "-ilities" already defined?

RQ2: How does viability of constituent systems relate to the viability of systems of systems?

RQ3: Do existing system-level strategies for enabling systems to resist the harmful effects of contextual change apply to systems of systems as well? Are there additional strategies that enable viability in systems, particularly systems of systems?

RQ4: Is changeability always a good thing? What can system architects do to enable positive change in complex systems, while restricting change that can be harmful?

1.3 Organization of the Dissertation

Following this introductory chapter, the dissertation is organized as follows:

Chapter 2: Maritime Security Case Study. Although several case studies will be cited from the literature to derive perturbation and SoS properties, as well as survivability design principles and approaches, an empirical study based on a maritime security system of systems (MarSec) will be the primary case study that will be discussed throughout this dissertation. The reasons for selecting maritime security as the case study are given in Chapter 2, as well as an overview of the stakeholder needs, context, existing components under consideration and initial perturbations of interest. At the end of the chapter, an overview of modeling and simulation techniques is given, as well as an explanation of the discrete-event simulation (DES) that was used to model the maritime security SoS and to generate the quantitative performance data.

Chapter 3: Value Delivery in a Dynamic Context. This chapter provides a summary of what is known and what has previously been done in the technical literature, in regards to how value delivery is threatened, and sustained.

Chapter 4: Engineering for Viability. The case for a new "ility", viability, is made in this chapter by showing that the current definitions of survivability and other related "-ilities" are confused in the literature, and do not address the full scope of the value sustainment, particularly the case where the context remains static, and yet the system still produces value.

Chapter 5: Viability Strategies for Systems. In this chapter, existing design principles for viability in the technical literature is discussed and it is shown that the list of principles vary by author. To clarify the literature, existing viability design principles are divided into general heuristics and specific design patterns.

Chapter 6: Applying Viability Strategies to Systems of Systems. In this chapter, existing strategies for enabling viability in systems are applied to systems of systems at both the system and SoS levels. It is also shown, through the North American power grid case study, that the existing strategies are not complete, and that the strategy of intensity regulation (i.e. throttling) is a strategy that is used to avoid and prevent perturbations within a SoS from becoming a cascading failure.

Chapter 7: Additional Viability Strategies. Using the concept of viability (i.e. avoiding things the system cannot survive, and surviving what it cannot avoid), the concept of system context and a cause-effect mapping, and various perturbations of interest to the MarSec case study are developed in this chapter. Then, strategies are deduced for avoiding or surviving those perturbations, and validated with historical case studies, such as Operation Titanic in WWII and Amazon's EC2 cloud Service Level Agreement.

Chapter 8: Endogenous Change. This chapter answers research question 3 by showing examples from historical case studies, as well as the MarSec that show that changeability is not always a good thing.

Chapter 9: System Architecture and Pliability. In this chapter, a new "ility", pliability is introduced, which both restricts change, and facilitates it.

Chapter 10: Conclusions. A summary of the major contributions is presented, along with concise, direct answers to the research questions posed in Chapter 1. In addition, outstanding questions that may pose as interesting research opportunities for future studies are discussed.

1.4 SUMMARY

This introductory chapter highlighted the problem of system architects trying to design complex systems that are expected to provide value to stakeholders over long periods. In particular, systems are increasingly being interconnected to form large, systems of systems that could last

indefinitely, and have multiple stakeholders and diverse contexts. As seen with some of the recent, large-scale SoS failures such as the Fukushima Nuclear Disaster, the larger and more complex a system is, the more likely it will encounter something that will change, whether it is stakeholder needs, the context or even the system itself. An applicable concept, *viability*, has been defined for biological systems and organizations, but it has not been defined for engineered systems. This chapter developed research questions revolving around defining viability for engineered systems and developing strategies for enabling it, particularly in systems of systems, and highlighted a research approach on how those questions will be answered. The chapter ended with a description of the organization of this dissertation, highlighting what is to be expected in each chapter. The literature review begins in Chapter 3, but first, the next chapter discusses the selection of the case study that was simulated to provide empirical evidence for some of the concepts tested and developed in this research.

2 RESEARCH APPROACH

To answer the questions posed in Section 1.2, an approach based on (Ross & Rhodes, 2008a) was used (Figure 2-1). First, the literature was reviewed for existing "-ilities" that may help systems maintain value delivery in spite of perturbations, as well help or hinder exogenous change. These normative definitions were carefully studied, and problems and gaps were identified in the literature. To address some of these issues, a new "ility" (viability) was proposed. This research then identified existing design principles and strategies, either explicitly stated or implied in the literature, and examined historical case studies to form a descriptive research basis of what principles and/or strategies are currently being used.

The existing value sustainment strategies were then examined from a systems of systems perspective, and shown to be incomplete by providing an example of a missing value sustainment strategy that is particularly common for systems of systems. Using the concept of viability and applying a new approach to examining complex systems for perturbations, additional value sustainment strategies were induced from historical case studies and validated with the MarSec SoS simulation (Section 2.4). In other cases, viability strategies were deduced from deficiencies identified during cause-effect mappings of the MarSec SoS (Section 7.3), and validated with examples from historical cases. Finally, to address the issue of endogenous change, the incorporation of CONOPs into system architectures and "pliability" is introduced. The entire research approach is illustrated in Figure 2-2.

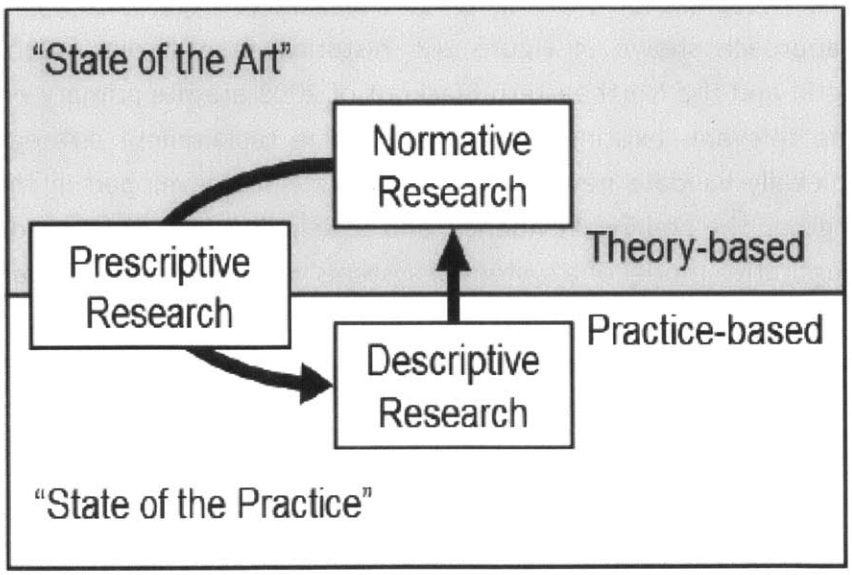

FIGURE 2-1: UNDERLYING STRUCTURE OF THE SEARI RESEARCH PROGRAM (ROSS & RHODES, 2008A)

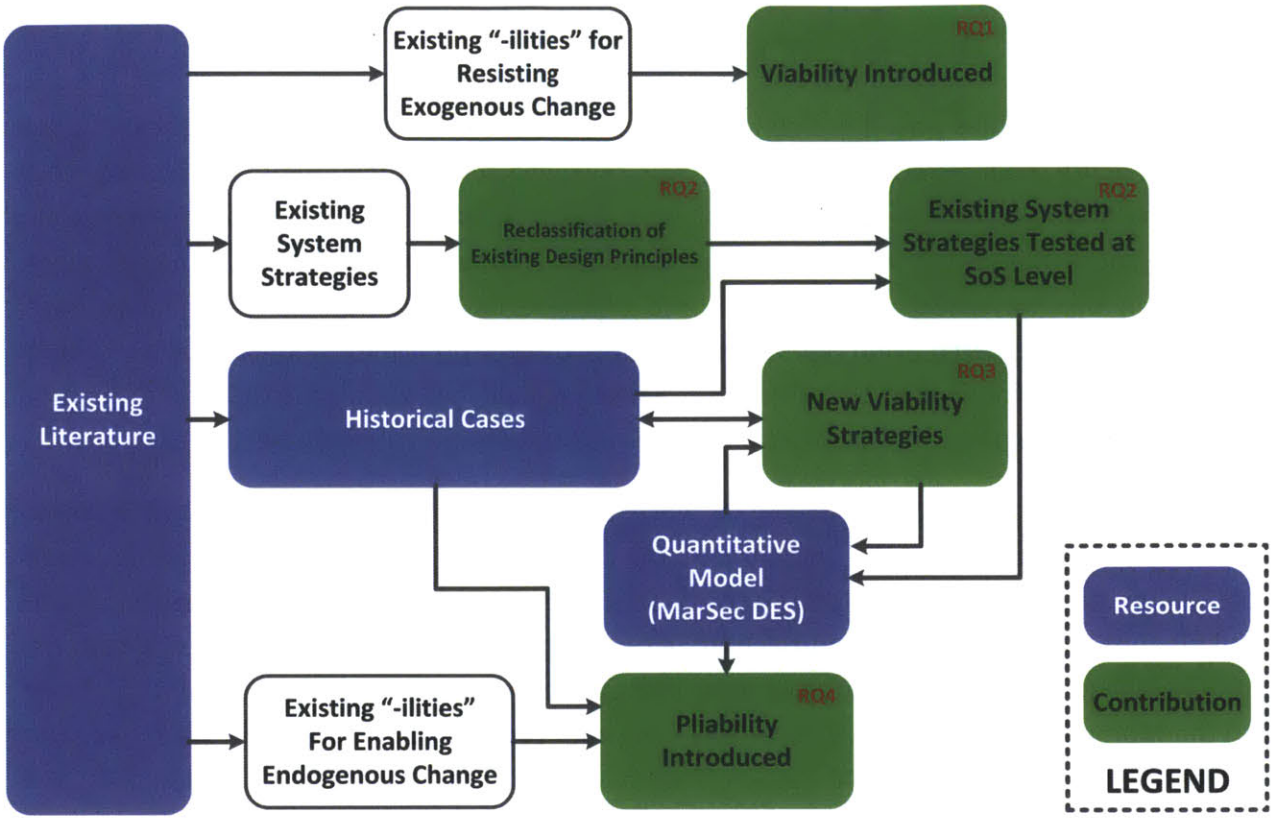

FIGURE 2-2: RESEARCH APPROACH

2.1 A Maritime Security System of Systems

In the research approach shown in Figure 2-2, historical case studies, such as the North American power grid and the Northeastern Blackout of 2003 are the primary empirical source used to test some relevant, existing "-ilities" and value sustainment strategies, as well as develop and empirically validate new "-ilities" and strategies developed in this dissertation. However, to strengthen the empirical evidence, and to help illustrate concepts described in this dissertation, a quantitative model of a system of systems was developed. A quantitative model allows for experimentation, and requires an analysis and specification of the system at a level of detail necessary to assess its performance. This level of depth, where the low-level interaction of constituent systems can be aggregated into higher-level functions, is necessary for validating certain concepts and is not easily obtained when looking at historical case studies.

2.2 Selection Criteria for SoS Used as a Quantitate Model:

The choice of SoS for quantitative modeling was not arbitrary. Rather, it was selected based on five criteria:

Strong Systems of Systems Characteristics. Part of the research objectives focuses on the differences between traditional systems and systems of systems as it relates to value sustainment, thus the quantitative model should be of a system that has strong SoS characteristics.

Dynamic Context. Much of the research in this dissertation involves contextual change. A suitable SoS would be one in which the context includes perturbations of varying type and duration.

Quantitative Performance Metrics. In order to assess the effectiveness of certain value sustainment strategies and to compare competing designs, the quantitative model must be able to produce quantitative performance data.

Precedence. The quantitative model should be based on a system of systems that has been previously identified and used in the literature as a SoS case study.

Representativeness. The design principles, methods and concepts developed in this research should be generalizable to other systems of systems that are similar to the one in the quantitative model.

2.3 Maritime Security System of Systems

Based on the criteria described in Section 2.2, the SoS selected to be modeled was a Maritime Security System of Systems (MarSec). This section will describe what a MarSec is, and how it satisfies the criteria described in Section 2.2.

2.3.1 Objectives

Not all maritime security system of systems are the same, but they typically share the goal of ensuring that ships entering a particular body of water are complying with appropriate rules and regulations. At a minimum, maritime security systems of systems are responsible for maintaining a certain degree of situational awareness over a particular body of water, known as the Area of Interest (AOI). Usually, the MarSec is required to know the location, identity and heading information of any ship within the Area of Interest (AOI). In many cases, the MarSec is

responsible for enforcing the laws and regulations governing the AOI as well, and this typically means tracking and intercepting any possible ships that may be violating those rules.

2.3.2 COMPONENTS

The primary effectors of the MarSec are manned and unmanned vehicles. All of the unmanned vehicles are aerial (i.e. UAVs) while the manned vehicles include both aerial and naval. These vehicles are not merely components of a larger system, but systems in their own right. In addition to the vehicles, there are supporting systems, such as command center, radar towers and ground control stations. The selection of components that constitute the MarSec is variable, and one of the ways in which strategies can be tested and induced, but design options for the MarSec in the quantitative model included the following:

Operational UAVs (OUAV). UAVs used for operational functions, such as detection of targets.

Tactical UAVs (TUAV). UAVs used for identification and observation of targets.

Manned Patrol Boats (MPB). Manned patrol boats used for detection, identification, observation and interception of targets.

Manned Helicopters (MHel). Manned helicopters used for identification and observation of targets.

Manned Patrol Aircraft (MPA). Manned patrol aircraft used for identification and observation of targets.

Command Center. Center used for coordination of tasks between the vehicles of the MarSec.

Airbases. The location where manned patrol aircraft reside when not in use as well as the location for the ground control stations controlling the UAVs. The distance between the UAVs and the ground control station impacts the quality of the communications.

Docks. The location where manned patrol boats reside when not in use.

2.3.3 SYSTEM OF SYSTEMS CHARACTERISTICS

A MarSec exhibits some of the key properties that distinguish systems of systems from traditional systems (Mekdeci, Ross, Rhodes, & Hastings, 2011a), including the following:

Component Independence. The vehicles themselves are capable of operating independently and providing value to their own stakeholders outside of the SoS. For example, a UAV that is part of the SoS, can be used outside of the SoS, for wildlife tracking or to fight forest fires, for example.

Distributed Authority. Managerial independence requires that the constituent systems have the ability to make decisions on their own, or through collaboration with other constituent systems. A MarSec can be configured such that decision making, such as task assignment, is done via a central authority, or done collaboratively by the vehicles themselves.

Geographical Separation. Maritime security systems of systems are typically made up of several vehicles, along with supporting components (e.g. radar towers, ground control stations, etc.). Unlike traditional, monolithic systems, most of the components within a MarSec are not co-located, but rather are distributed around a relatively large AOI.

Multi-Functionality. A simple, traditional system is more likely to have a single function or purpose, whereas a system of system is more likely to be multi-functional. In a MarSec, there are several tasks that need to be performed, i.e. detect, identify, and if necessary, intercept targets.

Increased Contextual Diversity. Since components in systems of systems are more likely to be physically separated than those in traditional systems, it follows that they will be more likely to be operating under different environmental conditions. While a UAV in one part of the AOI may be experiencing a relatively high load of including ships, other UAVs in other parts of the AOI may be idle. Certain vehicles in the MarSec, for example, may be operating on water, others on land, while others are in the air.

Decreased System Awareness. Since components in the MarSec are often operating under different contexts, they must share the contextual information with each other in a timely manner, for all of the components to have the same system awareness at any given time. In order for that to happen, three things must occur; (1) the important differences in context must be apparent, (2) stakeholders must be willing to share this information (not always the case, particularly if the contextual differences are the stakeholder preferences and policies), and (3) mechanisms must exist for this information to be shared in a timely manner. For these reasons, components within systems of systems that operate under different contexts may be operating under incorrect or incomplete information about the system itself, than components within traditional systems operating under the same context.

Evolutionary Development. Traditional systems are typically assembled during implementation, before the system is operated. Since components of systems of systems are often added or removed during the operation of the SoS, what the SoS is composed of may be continually evolving. Such is the case with a MarSec, which may change its configuration at any time based on evolving stakeholder needs and contextual considerations. An example of a

change might be to replace larger UAVs coordinated by a central authority with swarms of smaller UAVs that coordinate their tasks collectively.

Abstruse Emergence. In traditional systems, emergent behavior is often part of the design (or at least expected), and as such, is usually a benefit overall. In systems of systems, particularly those with evolutionary development, emergent behaviors are more difficult to predict and often end up being problematic. Such is the case with a MarSec, particularly when disruptions occur and there is a disconnect in the information sharing between the various components. This may lead to one component acting and re-acting based on a different set of information than another, leading to emergent behavior that is difficult to predict.

2.3.4 Dynamic Context

For this quantitative model, the area of interest is a littoral environment, consisting of narrow water passageway located between two landmasses (Figure 2-3). Typically, all ships above a certain size are required to have international transponders that pass the relevant identity, cargo and destination information to local port authorities, although additional confirmation by the MarSec is sometimes required. The MarSec normally has to identify smaller ships that typically do not have transponders. Methods of identification include marine radar, visual inspection and/or direct communication. Although there are many different types of ships that pass through this AOI, they can be divided into three broad categories; (1) compliant ships, (2) smugglers, (3) terrorists. Compliant ships are a general category for ships that are complying with the appropriate laws and regulations, and are not hostile towards the MarSec. Compliant ships include civilian sailboats, cargo ships, cruise ships, fishing vessels, police boats and friendly military vessels. Smugglers are non-compliant ships carrying contraband. A vessel of any size may contain contraband, but larger vessels have more capacity for smuggling. However, cargo size is not an indicator of cargo importance, as smugglers may hide some of the most dangerous contraband, such as weapons-grade uranium, aboard smaller vessels. Terrorists are non-compliant ships that wish to detonate high-impact explosives at the port. For the quantitative model, terrorists have a Modus Operandi (MO) based on a previous study on a low-probability, high-impact terrorist attack on a maritime port conducted by the Naval Postgraduate School (Buschmann et al., 2005) in which terrorists appear to be compliant civilian vessels until they are close to the port, at which point they head at full speed and detonate a bomb.

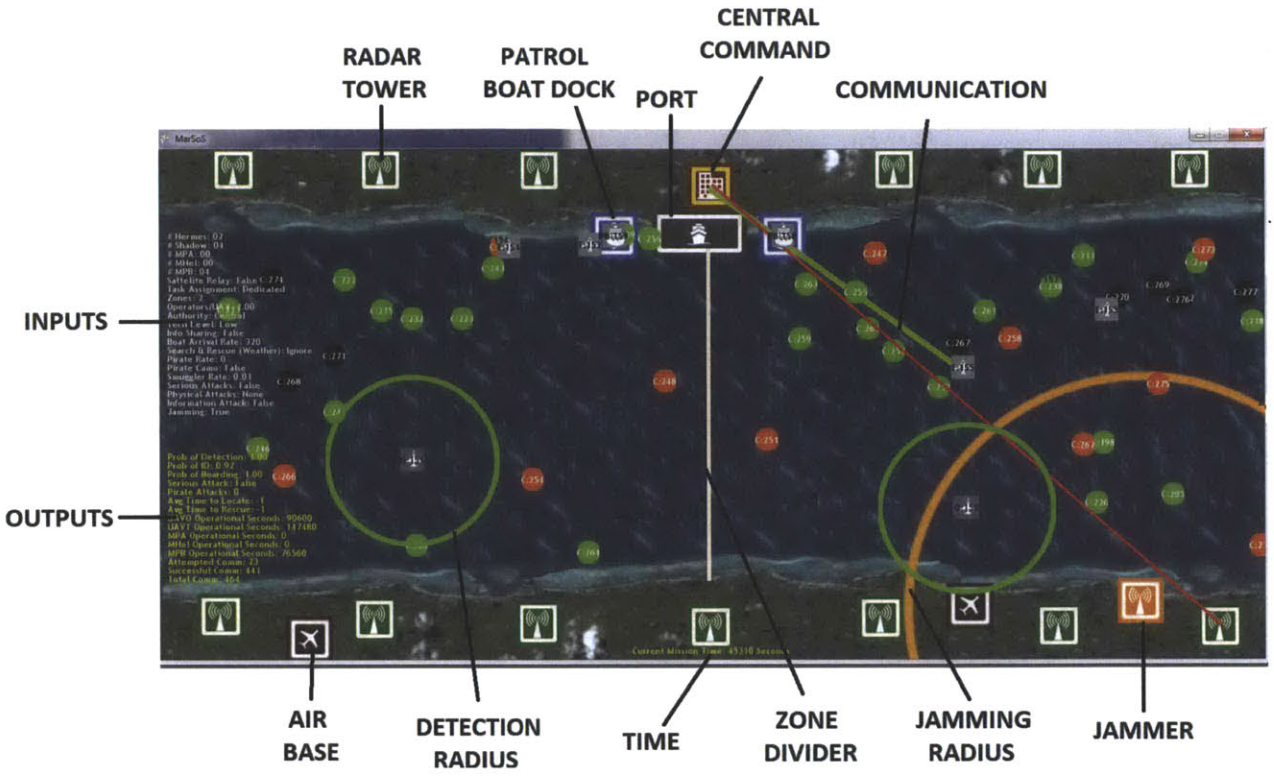

FIGURE 2-3: SCREENSHOT OF AOI USED IN THE QUANTITATIVE MODEL OF A MARSEC SOS.

2.3.4.1 CONTEXTUAL VARIABLES

There are numerous contextual considerations that can have a large impact on the performance and survivability of the MarSec. Some of these variables include:

Civilian Ship Arrival Rate. The number of ships that enter the AOI, per unit time. This is the main variable that controls the traffic volume for all the ships, friendly and hostile, that pass through the AOI.

Smuggler Rate. The percentage of ships entering the AOI that are smugglers.

Terrorist Scenario. Whether or not a terrorist will enter the AOI and attempt to detonate a bomb in the port.

Jammer. Whether or not a jamming device is located in the AOI, which may interfere with communication between entities in the MarSec.

2.3.5 QUANTITATIVE PERFORMANCE METRICS

The primary attributes for a MarSec are quantitative in nature. Specifically, performance is assessed by the following:

Identification Rate. The ratio of the number of identified ships to the total number of ships that entered the AOI. The identification rate is used to estimate the probability of identification, which is the probability that a ship entering the AOI is correctly identified before it reaches port or leaves the AOI.

Interception Rate. The ratio of the number of intercepted hostile ships (terrorists, smugglers) to the total number of hostile ships that entered the AOI. The interception rate is used to estimate the probability of interception, which is the probability that a hostile ship entering the AOI is intercepted before it reaches port or leaves the AOI.

Probability of Terrorist Bomb Detonation. The probability that a terrorist can successfully detonate a bomb. In order for a bomb to detonate in the port, a terrorist must have entered the AOI, and then the terrorist must not have been intercepted before he reached the port. Suppose *P(T)* is the probability that a terrorist enters the AOI over some period T that defines a scenario. The detonation of a bomb *(b)* is unique event and Boolean in nature. The probability of a bomb being detonated in a port given that a terrorist has entered the AOI *P(B|T)*, is determined by looking at *n* scenarios in which a terrorist has entered the AOI and counting the number of times a bomb was detonated.

$$P(B|T) = \frac{1}{n}\sum_{n} b_i \tag{1}$$

Using Bayes Theorem,

$$P(B|T) = \frac{P(T|B)P(B)}{P(T)} \tag{2}$$

rearranging we get

$$P(B) = \frac{P(B|T)P(T)}{P(T/B)} \tag{3}$$

Since $P(T|B) = 1$, we can determine the probability of a terrorist detonating a bomb in general *P(B)* by

$$P(B) = \frac{1}{n}\sum_{n} b_i\, P(T) \tag{4}$$

2.3.6 PRECEDENCE

Maritime security systems have been identified as a SoS in the literature (Gonzalez, Piel, Gross, & Glandrup, 2008), and have also been used by researchers in empirical case studies of systems of systems (Buurman, Zhang, & Babovic, 2009; Huynh & Osmundson, 2006; Monperrus et al., 2010).

2.3.7 REPRESENTATIVENESS

A MarSec is a system that shares many properties and characteristics with other large-scale, geospatial-centric system of systems including taxi services, emergency response services, and Joint Forces. These similarities are discussed below:

2.3.7.1 TAXI SERVICES

A major taxi service used previously as a case study for system of systems (Sauser, Boardman, & Gorod, 2008) is similar in many ways to a maritime security systems of systems. The major components of a taxi service are a central dispatch and the taxicabs themselves (Figure 2-4). Like maritime security, the constituent systems are geographically dispersed, they need to travel to certain locations (i.e. customers and destinations) which appear randomly, and time is off the essence. Also like maritime security, the constituent systems have operational and managerial independence. Although they belong to and benefit from being part of the overall taxicab service, individual taxicabs can choose to pickup/drop-off customers as they see fit, and choose to participate in various concepts of operations (such as using GPS positioning or not) to some extent. Customers may call central dispatch and request a cab or they may try hail a taxicab off the street. Either way, this information is shared between the central dispatch and the taxicabs, so that the constituent systems maintain situational awareness and the tasks (i.e. picking up customers) are shared properly. Exactly who determines which taxicab goes where is an operational consideration that the stakeholders in the taxicab service have to make. Many taxicab services are virtual systems of systems, in that the constituent systems are not working together for a common goal, but rather they are merely utilizing each other (in this case the cabs and the central dispatch) to achieve personal objectives.

FIGURE 2-4: TAXICAB SERVICE IN NEW YORK

2.3.7.2 Emergency Response Services

Emergency Response Services (ERS) are similar to taxicab services , in that they have been used as a case study for modeling systems of systems (Zhou et al., 2010) , and they have geographically dispersed constituent systems which must respond to spatially-diverse events in a timely manner. They consist of fire, ambulance and police services (Figure 2-5), and they also have a central dispatch (usually 911 emergency services) where calls typically originate and tasks are disseminated. One major difference between the taxicab service and an ERS, is that in an ERS, the constituent systems are heterogeneous similar to how they are in a MarSec. The heterogeneity of the emergency response system is required since different types of tasks are necessary and different organizations specialize in providing those services. Major accidents often require fire fighters to rescue victims from vehicles or buildings, Emergency Medical Services (EMS) crews to tend to the injured and take them to the hospital, and police to secure the accident scene, direct traffic and conduct an investigation. Most of these services are unique to their respective organizations; however, some duplication of services is expected. For example, fire crews often provide basic medical assistance if they are the first to arrive at the scene of the accident, and they will also assist in, or conduct their own investigation of the accident, particularly if it involves a fire, chemical spill or something of that nature. Like in a MarSec, which has vehicles that detect, identify, board, search and rescue, these dissimilar tasks and constituent systems have to be coordinated which makes the overall concept of operations more complex than that of a homogenous taxicab service. Also, ERS tend to be collaborative systems of systems, in that there is usually no central authority that commands all of the constituent systems, rather they combine their efforts to achieve common goals (i.e. maintain public health and safety).

FIGURE 2-5: POLICE, FIRE AND EMS VEHICLES AS PART OF AN ERS

2.3.7.3 Joint Forces

A Joint Force (JF), sometimes called a Joint Task Force (JTF), is a general term applied to a force composed of significant elements, assigned or attached, of two or more military departments (Figure 2-6), operating under a single joint force commander (Staff, 2010). JFs are typically used to conduct some major operation in a particular area of interest (i.e. a battlefield), and have been used numerous times as case studies for systems of systems (Biltgen, 2007; DiMario, 2006; Mittal, Zeigler, Martín, & Sahin, 2008). Similar to the maritime security SoS, taxicab service and ERSs, JFs are mostly composed of geographically separated constituent systems that must respond quickly to events that occur in disperse locations within the AOI. A JF is typically an acknowledged SoS, in that there is a central authority (the joint force commander), however, there is still some level of managerial and operational independence within the constituent systems.

FIGURE 2-6: A JOINT FORCE CONSISTING OF COMBINED NAVAL AND AIR FORCES

A summary of some of the SoS properties of these various systems is presented in Table 2-1: SoS Properties of Various Large-Scale Systems.

TABLE 2-1: SOS PROPERTIES OF VARIOUS LARGE-SCALE SYSTEMS

SoS Characteristic	Maritime Security	Taxi Service	Emergency Response (911)	Joint Task Force
Bounded Area of Interest	Yes (coastal area)	Yes (city)	Yes (city)	Yes (battlefield)
Components	Central authority, UAVs, helicopters, patrol boats, radar towers, etc.	Central dispatch, sedans, limousines, minivans	Central dispatch, police, fire, ambulance	Command center, troops, tanks, artillery, bombers, destroyers, etc.
Geographically Separate Constituent Systems	Yes	Yes	Yes	Yes
Location-Driven Events	Yes	Yes	Yes	Yes
Context changes	Traffic, weather, city bylaws, fuel prices, major events, politics, technology upgrades etc.	Traffic, weather, city bylaws, fuel prices, major events, etc.	Traffic, weather, city bylaws, fuel prices, major events, etc.	Weather, hostile forces, fuel prices, political events, technology upgrades, etc.
Heterogeneous Constituent Systems	Yes	No	Yes	Yes
Emergent Behavior	Yes	Yes	Yes	Yes

2.4 Modeling a Maritime Security SoS

2.4.1 Choosing an Appropriate Modeling and Simulation Technique

This research focuses on architecting systems of systems to maintain value delivery to stakeholders, throughout, and in spite of, endogenous and exogenous change. It is not practical to experiment with a large, real-world SoS, especially when testing survivability strategies where degradation or even destruction of the system is likely. Therefore, modeling and simulation is used in lieu of an actual system, as a proxy. There are many different ways to model a system, and a discussion of the choice to implement the MarSec as an agent-based discrete-event simulation follows.

2.4.1.1 Modeling of Systems

Models are used by researchers to give insight into various designs and contexts when it is impossible or impractical to use real systems and environments. Although the real systems and environments may be very complex, all models make some assumptions which may simplify the underlying relationships enough so that they may tractable. If simple, parametric models can be made that describe the real phenomenon with enough fidelity that they are useful to decision makers, then it is often possible to use numerical methods and computing solutions to explore a full tradespace of millions of designs and contexts. However, if it is necessary to capture emergent behavior that is not easily simplified, as in the case with many systems of systems particularly those that involve socio-technical aspects, then simulation is often necessary.

2.4.1.2 Simulation of Complex Systems

In choosing the type of simulation to be used, several considerations were made. First, since time is a critical property of a MarSec, dynamic simulations are required. This means static simulations, such as traditional Monte Carlo methods, are inappropriate (Law & Kelton, 2000). The next consideration to make is whether to use discrete or continuous models. Continuous models are more appropriate for modeling continuous phenomenon, such as electricity flow through a power grid or climate change caused by CO2 emissions. However, endogenous and exogenous changes in the MarSec are typically discrete. A ship randomly arriving in the AOI, a UAV being shot down, or a smuggler being identified as hostile entity are all examples of discrete-events that occur during the operation of a MarSec. It is at these discrete-events that the state of the system and/or context changes, the effects of which must be determined through simulation. The last consideration to be made is whether or not the simulation is deterministic or stochastic. As with many complex systems of systems, there is a large amount of uncertainty both in context and within the system. Inputs to the system are often random in both time and space. The arrival of boats in the AOI, for example, can be modeled as a Poisson process with a mean time between boat arrivals of $1/\lambda$, and a random location uniformly distributed along the border of the AOI. Endogenous events can be stochastic as well, such as the exact time it takes a human operator to identify a ship once he has a visual from a particular UAV. Since there is uncertainty in both exogenous and endogenous events, a stochastic simulation, particularly one that models the extreme conditions, is necessary particularly if survivability and robustness of the MarSec is a concern.

Discrete-event simulation is a popular technique that has been used to model large-scale systems of systems like maritime security systems of systems and ERSs (Bruzzone, Tremori, & Merkuryev, 2011; Gonzalez, 2009; Henderson & Mason, 2005; Leathrum, Mathew, & Mastaglio,

2011; Ng, 2007; Thomas, 2008). Discrete-event simulations are particularly well suited to models that have stochastic, dynamic and discrete-event properties, because they simulate discrete-events in a step-by-step manner, updating the state of the system and its context at specified time intervals, and allow random variables as inputs (Figure 2-7).

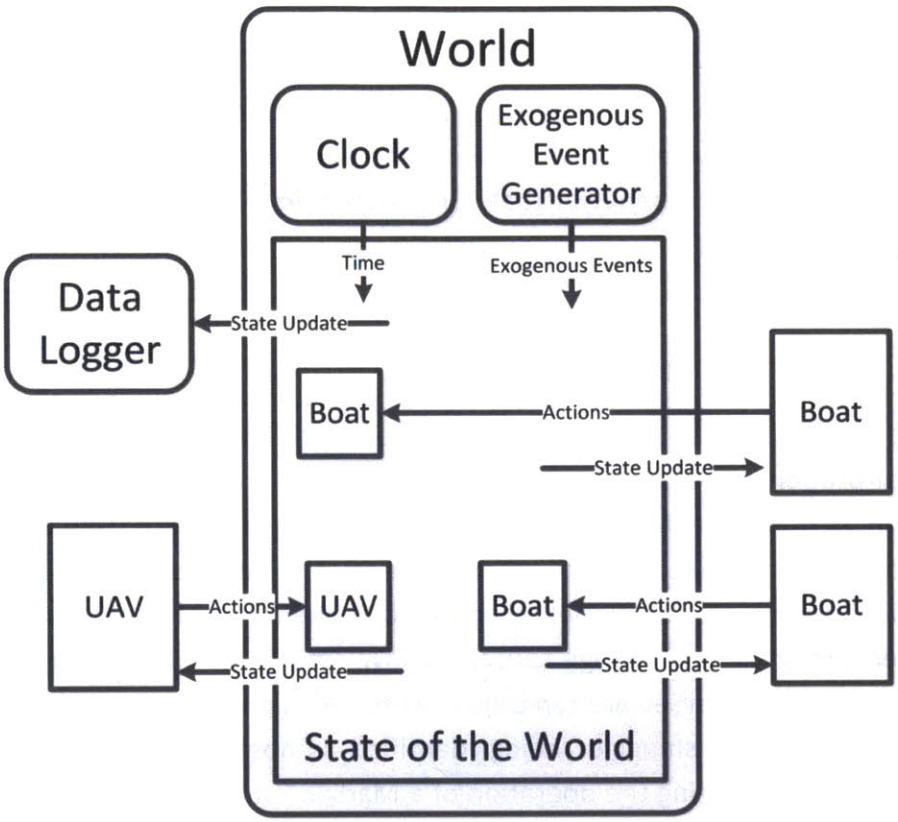

FIGURE 2-7: INFORMATION FLOW THROUGH THE DES

2.4.1.3 Agent Based Modeling

Systems of systems have many properties that make assessing value delivery non-trivial. Unlike many simple systems, equations alone do not adequately describe the relationship between components. Constituent systems have managerial and operational independence, which makes modeling of individual behavior and decision making necessary. Action and reaction to environmental changes is not simply a matter of physics. Instead, constituent systems perform actions based on the information they have at the time and their own priorities. If humans are operating some of the constituent systems, then there is typically going to be a stochastic behavioral element to be considered as well. When the individual behaviors of the constituent systems interact with each other, the behavior of the overall SoS may not function as expected.

As an example of the complexity of the overall SoS behavior, consider an ERS that responds to a large fire that is reported by a 911 call. A single police car, which happened to be in the neighborhood, is the first to arrive at the location and quickly discovers that it is a false alarm. All of the other vehicles, however, are still on route to the scene, because this local information (the fact that it was a false alarm) has not been shared. Whether or not the other constituent systems continue to the scene or turn back depends on a number of factors such as when the false alarm is reported, the priorities and protocols of the individual constituent systems (e.g. does the fire department trust a single police officer's assessment of the situation or do they need to conduct their own investigation), traffic conditions, and whether or not other events are happening in the city that require attention. The number of vehicles that still respond to the false alarm impacts value delivery immediately (such as fuel spent) but more importantly, may serious impact future events as well. The ability to respond to a major chemical spill that happens 20 minutes later at the other end of the city will likely depend on how many vehicles responded to the false fire alarm.

To assess the performance of a MarSec, an agent-based DES was implemented where individual constituent systems make tactical decisions based on their own behavior characteristics and the local information they have at some particular time. This way, the individual actions and reactions can be ascertained and overall behavior of the SoS can be assessed.

2.4.1.4 Limitations of Agent-Based DES

As with any modeling or simulation technique, agent-based, discrete-event simulations do have limitations and drawbacks. Almost all mathematical models require simplifying assumptions, which may affect the validity of the results, and agent-based discrete-event simulations are no exception. Also, as with any model, the quality of the outputs is strongly correlated with the quality of the inputs, e.g. if there is an error in the assumption used to determine an input such as the arrival rate of ships into the AOI, then there may be a significant error in one or more outputs, such as the probability of detection.

Due to the stochastic nature of the inputs and models, the outputs of the agent-based DES are typically stochastic themselves. This means that for a given set of inputs, the simulation must be run multiple times (e.g. "trials") to gather enough samples to estimate the true attributes of the system. Depending on the level of statistical confidence required, the number of trials required can significantly increase the computational load. This is a major concern, since perhaps the largest drawback to agent-based, discrete-event simulations is the considerable computational effort required to calculate the performance of even one design under once context. Determining the potential actions and reactions of every entity, for each small increment of time, is often cumbersome even for modern computers. If the behavior of each

entity depends on the behavior of all the other entities in the system, then in many cases the behavior f(n) for the entities in the system is O(n!) meaning that the computation effort required grows in a factorial relationship with the size of the SoS and context. This can become a very serious problem if a large tradespace consisting of thousands of designs and potential contexts is to be explored, and multiple runs have to be made for each design/context pair for statistical confidence.

2.4.1.5 Limitations of Observational Data

Simulations are subject to modeling error, and that is a primary argument against using data from simulations to make design decisions. Observational data from real systems in real contexts tend to be preferred, but for many even reasons, simulations are not only necessary, but sometimes more useful even if some observation data exists.

Dependent on real systems and real contexts. Unlike simple systems such as stereos, it is often impossible or impracticable to build prototypes of real, complex systems, particularly systems of systems. For example, it would be very unlikely that Sony would be able to build a prototype of the Sony PlayStation Network that included all of the different stakeholders and 70 million users and electronic devices around the globe. Furthermore, even if actual prototypes could be built, they cannot (and often should not) be exposed to the actual contexts that the real systems may encounter. For example, in addition to being extremely expensive and time consuming, it would be very immoral for stakeholders to attack their own military systems of systems with live ammunition if there is any danger to the human operators. Observational data can be still be gathered using prototypes of varying size and fidelity in contexts that are somewhat similar to actual conditions the systems may face, but this obviously introduces some uncertainty in the validity of the results. For example, observational data about a small, manned maritime security SoS operating in Maine might be available, but how does this data infer the performance of a large, mostly unmanned maritime security SoS defending the Japanese coastline?

Small sample sizes. Complex systems tend to operate in highly dynamic environments and often have their own stochastic responses, particularly if human decision making is part of the concept of operations. Even if observational data is available, there may not be enough available to accurately predict the system response to particular context. For example, data on how a system responded to a few, severe thunderstorms may be available, but is this enough to be able to say with confidence that the system is survivable (or not) to thunderstorms? Unfortunately, small sample sizes means that the estimates of the performance of the system will often be clouded by statistical error. To be able to account for the stochastic nature of the system and its environment, many samples have to be taken which is often not possible in the

real world, particularly for rare, but important contexts. Most simulations, on the other hand, can be run thousands of times, reducing the statistical error to trivial levels.

Measurement Error and Intermediate Metrics. Observational data relies on what was observed. Sometimes, it is difficult to measure certain metrics in the real world and there is a measurement error that needs to be considered. For example, a person with a stopwatch timing how long it takes an operator to identify a particular target introduces an measurement error that includes the ability to determine exactly when the process started and stopped, as well as a reaction time. Observational data also often doesn't measure intermediate metrics that may be important, particularly in determining why something happened. For example, an airplane may crash in a thunderstorm, but why it crashed may be difficult to assess without something like a black box. Even with a black box, forensic engineers often have to make inferences about the situational awareness of the pilots from their conversation or whatever actions were recorded, for example. In simulations, many of these variables can be directly measured and recorded for tracing purposes.

2.4.1.6 Verification and Validation of the Agent-Based DES

A maritime security, agent-based DES was implemented in software, based on a set of conceptual models of maritime security systems of systems. Verifying and validating the simulation involves proving that simulation accurately implements the conceptual model (i.e. verification), and proving that the simulation accurately represents the real system (i.e. validation) (Figure 2-8). Verification and validation of a simulation is a subjective process. What is considered valid and/or verified to some, is not to others. The purpose of verification and validation is to build confidence that the simulation accurately represents the real system, and that results and insights from the simulation can be applied to the real system. Verification and validation simply raises the confidence the user has of the simulation. However, verification and validation can be expensive process with diminishing returns. The relationship between building confidence in the simulation and the cost is not linear (Figure 2-9). As confidence is gained in the simulation, the next incremental increase in confidence gained by performing more verification and validation is vastly outweighed by the cost, such that it becomes infinitely expensive to gain 100% confidence in the simulation. Thus, a model builder should only do as much verification and validation necessary for the user to gain enough confidence in the simulation, and no more.

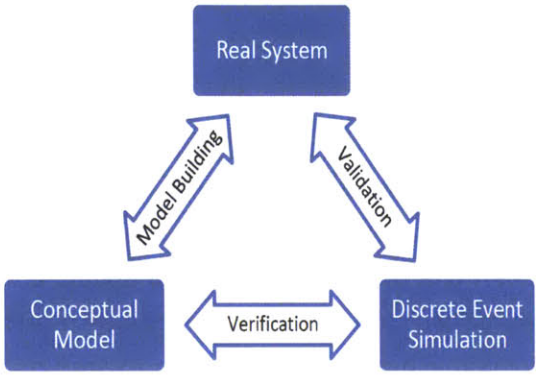

FIGURE 2-8: VERIFICATION AND VALIDATION OF A SIMULATION (BANKS & CARSON, 1984)

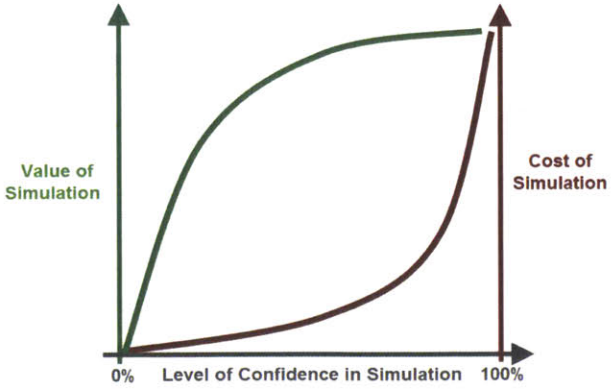

FIGURE 2-9: VALUE AND COST OF A SIMULATION VS. LEVEL OF CONFIDENCE IN THAT SIMULATION (SARGENT, 2005)

It is important to note that there is a risk to the model builder in the amount of verification and validation done on the model. If the model user is satisfied, then more successful verification/validation tests will simply improve the confidence. However, if any of these additional tests fail, then confidence in the model will likely decrease, perhaps enough to make the model user no longer confident enough in the model to accept it.

The goal of the verification is to make sure that there are no programming errors (bugs) and that the DES accurately implements the conceptual model correctly. There are several procedures that can build confidence that the model is verified. None of the procedures are essential, but all of them can help build (or reduce) confidence in the model. All of these procedures were performed on the simulation.

Modularity. The code was written in modules, using an object-oriented approach (Law & Kelton, 1997)

Tracing. A log of sequential events was kept as they occurred (i.e. a trace). This log was compared with a flow diagram of events from the conceptual model. (Banks & Carson, 1984) (Law & Kelton, 1997).

Input Integrity. Inputs including hardcoded parameters were verified to have remained constant after the simulation executed. (Banks & Carson, 1984)

Automatic Error Checking. There are over 100 automatic error checks in the code, to catch logic problems. For example, if an output of a function can only range between 0 and 1, an error flag is set if the function returns more than 1 or less than zero.

Visualization. A real-time animation of the output is generated to improve debugging and help determine if the DES model matches the conceptual model. (Law & Kelton, 1997)

Range Checking. For a given range of inputs, the simulation produced a reasonable set of outputs. (Banks & Carson, 1984)

Extreme condition testing. Some extreme values of inputs were used to test if the model. For example, boats arriving in the AOI every 10 seconds as opposed to every 320 seconds. The simulation did not break as a result. (C. Nehme, 2008)

Face Validity. Since no real system exists against which the outputs of the simulation can be compared, it is very difficult to validate the simulation. One of the best methods in lieu of comparing simulation data against actual data from a real system is to have subject domain experts take a look at the simulation and its outputs and demonstrate face validity. In the case of the simulation, the simulation was run for a group of system analysts that have built a similar MarSec and face validity was demonstrated.

2.5 SUMMARY

This chapter presented details on the research approach used to answer the research questions posed in Chapter 1. The literature was examined for existing definitions and strategies relating to viability, and gaps based on differences between the normative and descriptive research was identified. Those gaps were addressed through a combination of inductive and deductive reasoning, using historical cases and a quantitative model that was used to validate concepts and strategies, as well as to provide new ones. This chapter also justified why the MarSec SoS was chosen for the quantitative model, including the fact that it has strong SoS characteristics, is fairly representative of a class of important systems of systems, has quantitative performance metrics which can be easily modeled, and has been used in the past for similar SoS studies.

The next chapter will lay down the conceptual foundation for the remainder of the dissertation. Definitions of fundamental concepts will be presented, and there will also be a discussion on the current state of research in value delivery sustainment.

3 VALUE DELIVERY IN A DYNAMIC CONTEXT

This chapter begins by setting the basic definitions for many of the important concepts used in this research, such as system, context and value. Next, the chapter describes various value sustainment "-ilities" and discusses how they are defined in the literature.

3.1 DEFINING VALUE DELIVERY OF SYSTEMS

3.1.1 SYSTEMS

The term *system* describes an entity that is composed of other entities, i.e. "components", which interact with each other within some system boundary (Figure 3-1). A system boundary separates the system from its context. Context is everything outside the system that may influence the system, or be influenced by the system. Context includes such things as external entities, the physical environment and other conditions, with examples given in Table 3-1. An open system is a system that interacts with its context, whereas a closed system does not.

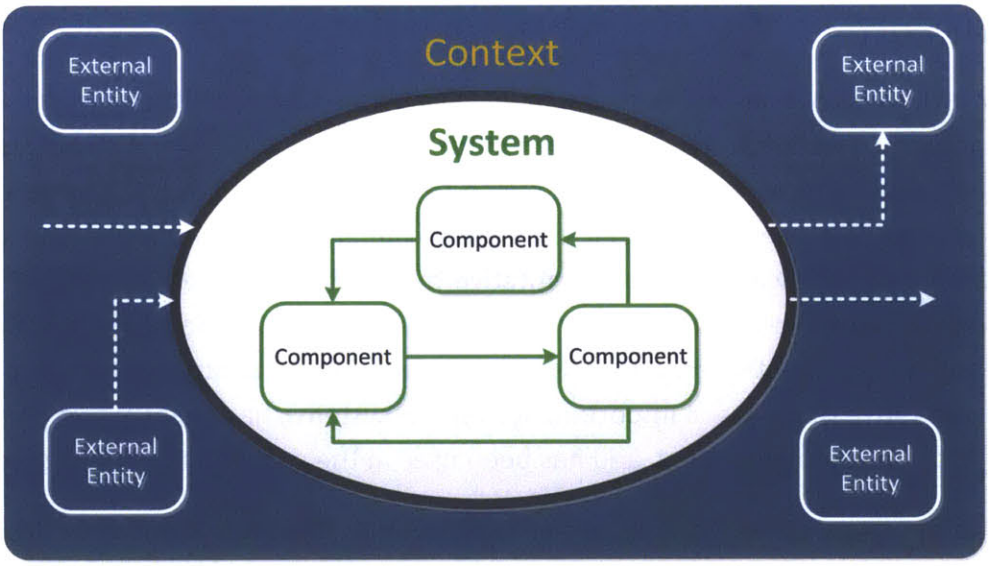

FIGURE 3-1: SYSTEM OPERATING IN ITS CONTEXT

TABLE 3-1: EXAMPLES OF CONTEXTUAL FACTORS

PHYSICAL ENVIRONMENT	EXTERNAL ENTITIES	OTHER CONDITIONS
Terrain	Customers	Current Date/Time
Weather	Wild Animals	Laws and Regulations
Precipitation	Enemy Units	Market Prices

There are many types of systems, including natural systems and engineered systems. Many natural systems, such as ecosystems, do not have a clear, identifiable purpose while others, such as the repository system found in the human body, do. *Engineered systems* are artificial systems designed to provide some value to stakeholders. Henceforth, unless otherwise stated, the term "system" in this dissertation will refer exclusively to engineered systems. A traditional system is one that is composed of components that do not produce value to its stakeholders on their own, but rather has components that must be combined to produce some useful effect. Table 3-2 provides a few examples of traditional systems.

TABLE 3-2: EXAMPLES OF TRADITIONAL SYSTEMS, COMPONENTS, AND BENEFITS

TRADITIONAL SYSTEM	EXAMPLE COMPONENTS	BENEFITS TO STAKEHOLDERS
Stereo	CD player, speakers, amplifier, wire, radio tuner	Ability to hear recorded sound stored on compact discs, ability to hear radio broadcasts
Laptop Computer	CPU, RAM, hard drive, motherboard, keyboard, monitor	Ability to read / write documents, perform scientific simulations, play games
Car	Engine, steering wheel, seats, wheels	Ability to transport passengers / cargo from point A to point B, fun to drive, status symbol

Engineered systems are not to be confused with *engineering systems*, which are systems that have both a high degree of technical and social complexity (de Weck, Roos, & Magee, 2012). Many engineered systems are also engineering systems and vice-versa, but there are differences. A laptop, for example, is an engineered system but not an engineering system, because it lacks social complexity. Similarly, there are technically and socially complex systems that emerge spontaneously but are not designed, such as the Internet itself, which can be considered an engineering system but not an engineered system.

3.1.2 VALUE

The focus of this dissertation is to study the ways that systems of systems can maintain value delivery over their lifetimes. Since systems will undoubtedly cost something to design, build and operate, the overall benefit that a system provides its stakeholders has to be weighed against its cost. For the purposes of this dissertation, value refers to the net benefit or utility (i.e. received benefits less costs for receiving those benefits) (Richards, Ross, Hastings, & Rhodes, 2008) an engineered system provides to its stakeholders, however value can have

other definitions that are used by decision makers to evaluate and compare system designs (Ross, O'Neill, Hastings, & Rhodes, 2010).

The design, behavior and context of a system all have an impact on its value delivery. The total area coverage of a MarSec, for example, is determined by design properties such as the technical specifications of the payload, behavioral factors such as the altitude and route that the UAVs are flying, and contextual factors such as the weather and terrain. Not all system properties or behaviors are relevant to all attributes. For example, the color of the UAVs has no effect on the total area coverage, but it does influence how visible the UAVs are. Similarly, while everything outside of the system boundaries can be considered context, very little of it is actually relevant to the value delivery of the system.

Value is subjective. What is valuable to one stakeholder may not be valuable to another. A car may be valuable to one person because it gets them from point A to point B, but the same car may be valuable to someone else because it is fun to drive. Stakeholders assess value by applying their preferences to system attributes. The attributes of a system are all the relevant aspects that the system has or does to provide value to the stakeholders. Table 3-3 provides some examples of stakeholder attributes that are mapped to system objectives. Preferences (sometimes quantified as "utility functions") are a measure of how well system attributes meet objectives and therefore provide value to stakeholders. Preference, or utility, curves typically normalize the utility associated with attributes to a numeric score between 0 and 1, facilitating comparison with each other when comparing designs.

TABLE 3-3: EXAMPLE OF MAPPING SYSTEM BENEFITS TO SYSTEM ATTRIBUTES

System Objective	System Attribute
Provide surveillance of enemy territory	• Minimum time between location revisits • Total area coverage
Fun to drive	• Maximum acceleration • Turning radius • Presence of a convertible roof • Noise
Able to transport goods	• Maximum carrying capacity • Maximum range • Maximum speed
Aesthetically pleasing	• Color • Shape

3.1.3 Stakeholders and Decision Makers

Stakeholders are those who are supposed to benefit from the outputs of an engineered system. A system may have many stakeholders, some of whom directly benefit from the system's outputs, and others who indirectly benefit. For example, suppose a company designs, manufacturers and sells stereos. Both the customers and the manufacturer are stakeholders of the system. The stereo will provide acceptable value to the customer if it does what the customer wants (e.g. provide good quality sounds), over its lifetime, and at a reasonable cost (to the customer). In this case, the customer is a stakeholder who directly benefits from the outputs of the system. If the stereo can meets the needs of the customers and it can be manufacturers, delivered and sold at a profit, then will provide acceptable value to the manufacturers. The manufactures indirectly benefit from the outputs of the system (e.g. the sound quality of the stereo) since if the stereo provides good sound, customers will be more likely to buy it and the manufacturer will be more likely to make a profit.

In addition to providing direct benefit to stakeholders, engineered systems are designed, manufactured, and operated by stakeholders. Many of these stakeholders may be the same, but sometimes they are different. For example, an amusement park ride is typically designed by one stakeholder, owned by another, operated by an employee and enjoyed by its rider. The system is designed to be enjoyable to the rider, be safe, operate effectively for a reasonable lifetime, and have a reasonable cost to the rider to enjoy, the owner to own and the manufacturer to sell.

Stakeholders are always decision makers, and they assess whether or not the system is meeting their value expectations. A Decision Maker (DM) is someone who has the authority to make a decision that can influence the system at the design, strategic or operational level. A Design Decision Maker (DDM) makes a decision during the design process that influences the capabilities that the system has. Typically, DDMs are the owners of the systems and the system architects who do the actual design. A Strategic Decision Maker (SDM) is someone that typically has a high-level of authority, and makes larger, strategic decisions after the system has been implemented, either during operations or between operations. Operational Decision Makers (ODM) typically have low levels of authority and make smaller tactical decisions during the actual operation of the system. If made during operations, then strategic decisions are often high-level and involve decision makers with high levels of authority. Decision makers are typically system architects, operators, owners, or other stakeholders that share common goals, and can directly influence the system. Example decisions and decision makers for a MarSec are presented in Table 3-4.

TABLE 3-4: EXAMPLE OF DECISION MAKING LEVEL AND DECISION MAKER FOR A MARSEC SOS

Example Decision	Type	Decision Maker
Adding the option to replace spare batteries on the UAVs with high-powered spotlights.	Design	Port authority, system architect
Removing the spare batteries on all the UAVs and replacing them with high-powered spotlights.	Strategic	Supervisor
Shining a spotlight on a target to confirm identity when visibility is low.	Operational	UAV operator

Decisions are hierarchal in nature, such that design decisions constrain strategic decisions, and strategic decisions constrain operational decisions. For example, shining a spotlight on a target (an operational decision) would not be possible if the field commander did not authorize that spotlights be placed on the UAVs instead of spare batteries. Similarly, the decision to use spotlights instead of batteries is only possible because of a design decision that allows the swapping of batteries and spotlights as an option.

Other entities may influence the system's value delivery, intentionally or not. Smugglers may try to outrun the patrol boats, unintentionally causing the MarSec to divert resources away from other parts of the AOI as they track and chase the smugglers. Similarly, a pirate ship may intentionally attack one of the patrol boats of the MarSec, if it feels threatened. Although both the pirates and smugglers made decisions that had an effect on the value delivery of the MarSec, they would not be considered decision makers because they do not have the authority to make voluntary changes to the SoS (even if they force the SoS to makes changes in response).

3.1.4 Epochs and Eras

Contexts can be temporally divided into distinct time periods known as epochs and eras. Ross and Rhodes (2008b) define an epoch to be a period of time for which the system has fixed context and fixed value expectations. By "fixed", it is implied that certain contextual factors are allowed to fluctuate in small ways that do not threaten the value delivery of the system. For example, a municipal power station may divide the year into two distinct epochs; the winter season and summer season. The key contextual parameter of interest is the demand for power. For this particular power station, the winter and summer seasons have significantly different demands for power, perhaps because the municipal is a summer vacation destination and the population doubles during those months. Demand for power is a continuous, stochastic variable and not likely to be the same at any given moment in time. Within each season, there are minor fluctuations in the power demand caused by such factors as the time of

day and the daily temperature. As long as these fluctuations are within reasonable limits, then the context can be considered "fixed". Eras are longer periods that are composed of several epochs in sequence. The entire system lifespan often consists of one era, however if the system undergoes a major change in its role or definition, then it may signal the end of one era and the beginning of another. An example of an era change would be when horse-drawn carriages on rails were replaced with electric trains.

3.2 Systems of Systems

The Department of Defense (2008a) defines systems of systems as "a set or arrangement of systems that results when independent and useful systems are integrated into a larger system that delivers unique capabilities". The independent and useful systems that make up a SoS are referred to as constituent systems. Not all of the components in a SoS are constituent systems; most systems of systems will still require some traditional components that are not systems themselves, but are required for the overall SoS to function properly. Some may argue that most "components" within a system, even within a traditional system like a stereo, are actually systems themselves. For example, a CD player can be considered a system because it is made up of components such as a motor, laser, and digital-to-audio-converter (DAC). Components of a traditional system that are composed themselves of subcomponents that interact with each other, are referred to as subsystems. Yet, what distinguishes a system of subsystems, from a system of systems?

There are a number of key system of systems properties that have been identified in the literature (Boardman & Sauser, 2006; Eisner, Marciniak, & McMillan, 1991; Ellison, Fisher, Linger, Lipson, & Longstaff, 1997; Ellison & Woody, 2007; Maier, 1998) which distinguish systems of systems from traditional systems. They are:

Operational Independence of the Elements. A system of systems is composed of constituent systems, which can independent and useful in their own right. They can exist and provide some value to stakeholders outside the SoS. In a traditional systems, components are not likely to be valuable by themselves or be able to operate outside of the system.

Managerial Independence of the Elements. The constituent systems have the ability to decide their own actions and behavior.

Evolutionary Development. A system of systems goes through evolutionary development where components and constituent systems are added, removed, and modified in response to changes in context.

Emergent Behavior. The DoD (2008b) defines emergent behavior as "behavior which is unexpected or cannot be predicted from knowledge of the system's constituent parts", even though it acknowledges that there is no single, universally accepted definition of emergence. Regardless, the reason why system architects construct a system of systems in the first place, is so that the SoS can perform higher-level functions that are not possible by any single constituent system.

Geographic Distribution. The span of the geographic distance between the component systems is so large, that they are usually only exchanging information and not useful quantities of mass and energy.

Connections. Constituent systems tend not to be totally independent or totally dependent when interacting within a SoS. Rather, they are inter-dependent and loosely coupled. In traditional systems, there tends to be tight coupling and strong inter-operations between components.

Multi-Functionality. The constituent systems within a system of systems tend to be able to perform multiple functions and roles within the SoS. Components within a traditional system tend to be uni-functional.

Contextual Diversity. Due to the increased geographical separation, constituent systems tend to have more contextual diversity in a SoS than components in a traditional system.

Unbounded. Systems of systems are often unbounded, meaning that components can be added, modified and removed independently of some central administrative control. Furthermore, the decision to add, remove or modify components is often done by stakeholders with a limited understanding of the entire SoS. Traditional systems tend to be bounded, where decisions about the addition, modification and removal of components are done by a central authority with complete knowledge of the system.

3.2.1 Types of Systems of Systems

Maier (1998) classified three major types of systems of systems, based on the task-sharing relationship between the constituent systems:

Directed. A directed system of systems is one that has a SoS management that sets a purpose for the SoS and manages the participation of the individual constituent systems. The constituent systems still retain some independence in their individual action, but must follow the overall SoS concept of operations and are considered subordinates to the SoS management.

Collaborative. In a collaborative SoS, there may not be a SoS management, but the constituent systems participate willingly and cooperatively to achieve common objectives. An example of a collaborative system of systems is would be the ad-hoc SoS that emerges when different emergency response agencies such as fire, police and ambulance respond to a major accident and work together to save lives and mitigate property damage.

Virtual. In a virtual system of systems, there is no central management or overall objectives. The constituent systems interact with each other independently, perhaps not even being fully aware of the overall ad-hoc SoS that they are part of. The Internet itself is a classic example of a virtual SoS.

Expanding on Maier's classification to include the types of systems of systems most often found in military organizations, Dahmann (2008) introduced another type of SoS, the acknowledged SoS.

Acknowledged. An acknowledged SoS is one where there is a central authority at the SoS level that sets policy and dictates an overall concept of operations, but the constituent systems have their own management, objectives and needs in parallel.

3.2.2 System Continuum

Mekdeci et al. (2011b) notes having one or more of the properties that characterize a system of systems, does not necessarily make a system a SoS. Similarly, most systems of systems do not necessarily exhibit every SoS characteristic that has been identified. Rather, complex systems, and systems of systems, fall somewhere along a system continuum (Figure 3-2) that ranges from traditional systems such as stereos and calculators, to complex system of systems such as Joint Task Forces and the Internet itself. Therefore, this dissertation will use caution when making claims about systems of systems in general, and instead use conservative language, i.e. "systems of systems tend to (be, have) ..." rather than "systems of systems (are, have) ..." where appropriate.

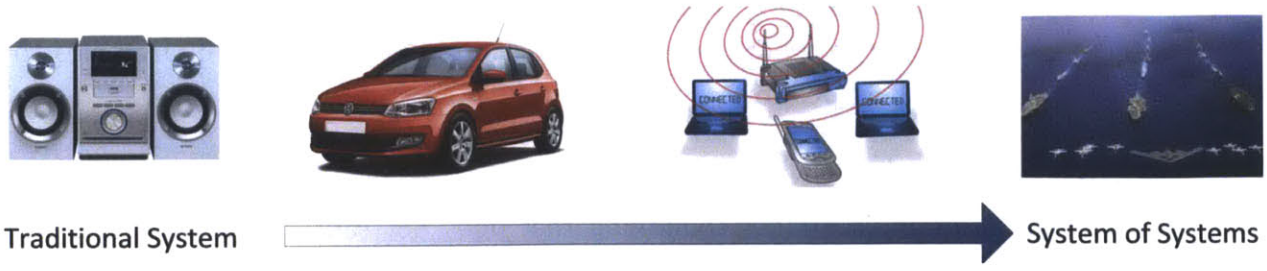

FIGURE 3-2: SYSTEM CONTINUUM

3.2.3 Global and Local Context

Not everything within a context is relevant to the system. Certain contextual factors are only relevant to certain systems. The market price of gasoline, for example, is probably relevant to a car, but not to a radio station. Relevancy of contextual factors is also due to differences between local contexts. For example, while the weather is relevant to an automobile, only the local weather is relevant, not what is happening on the other side of the Earth. Similarly, context can be relevant based on the roles and responsibilities a system assumes within a larger collection of systems. For example, while the presence of enemy troops are a concern to any unit in a Joint Task Force (including the MarSec), the location of an enemy submarine is more likely to more relevant to a naval ship than a foot soldier.

To distinguish between the overall context that exists, and the context that is relevant to a particular system, we define the global conditions, Φ, as everything that describes the state of the world at that particular moment, and the system context, φ_X, as everything that is relevant to a particular system X. For example, the global context includes the weather at every location on Earth, but a power plant in Boston is only concerned about the weather in its area. Therefore, if we let α_A denote a particular attribute of system A, and φ_A as the context that system A experiences, then we can say that

$$\alpha_A = f(A, \varphi_A) \tag{5}$$

3.3 Contextual Changes

In section 3.1.4, the concepts of epochs and eras were introduced, and it was stated that it is normal for systems to operate over a small range of contextual deviations within an epoch. Stakeholders should expect that minor fluctuations, such as the daily temperature rising and falling, will not cause the engineered systems operating in that context to fail. However, there may be certain events that can cause a rapid or significant change in context, which may reduce the value delivery of the system below stakeholder expectations. If these events are possible, then the system is vulnerable to them, and decision makers should implement strategies to reduce such vulnerability, where possible.

3.3.1 Significant and Insignificant Contextual Changes

Whether a change is insignificant (i.e. something that does not cause a context shift), or significant, is subjective. To illustrate the difference, consider a truck that delivers goods year-round, all over North America. From the perspective of the truck stakeholders, there are two epochs: winter and non-winter. In response to an epoch shift, the stakeholders either winterize

or de-winterize the truck. Winterization of the truck includes using special tires and anti-freeze windshield washer fluid. In this framing, each epoch has two distinct sub-contexts, whether it precipitates (snow/rain) or not. The "normal" subcontexts would be where there is no precipitation, in both the winter and non-winter epochs. The precipitation sub-contexts can be considered to be disturbances, since they are short, finite duration contextual changes that are significant enough to threaten the value delivery of the system (Figure 3-3). These changes are significant because there is a significantly higher chance of an accident during a precipitation sub-context, unless the truck changes delivery operations (i.e. slows down). Minor fluctuations, such as the daily temperature, can affect value delivery, but are usually not considered significant enough to describe as an epoch change. Note that even though only two epochs are shown in Figure 3-3, in the case of a truck, an era will often consist of many cycles of winter and non-winter epochs, and rain/snow disturbances that span the truck's lifetime.

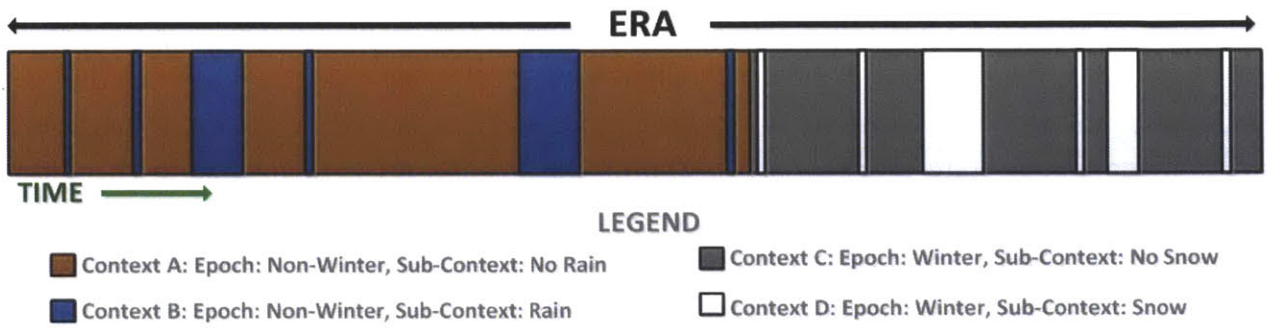

FIGURE 3-3: EXAMPLE OF AN ERA CONSISTING OF WINTER AND NON-WINTER EPOCHS

Suppose a crisis in the Middle East drives gasoline prices high enough that the owner can no longer afford to drive the truck as much as she wants. Does this change in gasoline prices mark the beginning of a new epoch? The answer is subjective, and largely depends on whether or not the gasoline prices return to an acceptable level within a "short time" or not. If the prices fall back down soon after they were raised, then that period of high gasoline prices can be considered a disturbance during an epoch of affordable gas prices. If the high gasoline prices remain for a long period of time, however, then the question of whether the prices will return to "normal" has to be asked. There are often cues that suggest whether a change will be short term, long term or even permanent. A rise in gasoline prices due to an unusually hot July will almost surely be a disturbance. A rise in prices due to a change in environmental law passed by Congress is more likely to be an epoch shift. In some cases, On the other hand, the contextual changes may be so significant that they may mark the beginning of a new era. For example, the owner may decide that it is no longer worth it to actually drive the truck around making deliveries, but instead decides to convert the truck into a mobile home.

3.3.2 PERTURBATIONS

There are many different terms in the technical literature for events that cause rapid or significant changes in context that threaten the value delivery of systems. Richards (2009) defines a "disturbance" as something which may asymmetrically degrade performance of a system. A sudden rise in gasoline prices that threatens the value delivery of the system would be an example of a disturbance. Jackson (2010) refers to external disturbances as "disruptions of input" and internal disturbances as "systemic failures". Mekdeci et al. (2012) refers to both disruptions and disturbances as perturbations, which is defined as an "imposed state change in a system's form, operations, or context which could jeopardize value delivery". In addition to system and contextual changes, Beesemeyer (2012) also includes imposed changes on stakeholder needs as a perturbations as well.

The duration of a contextual change is important. A lightning strike, for example, is an instantaneous event, whereas a thunderstorm has a finite duration. The key difference is that a system may have time to react to a finite-duration contextual change such as re-routing during a thunderstorm. From this perspective, Mekdeci et al. (2012) defines disturbances to be finite duration, continuous state changes, and disruptions as instant, discontinuous state changes. Beesemyer (2012) refers to a permanent, or very long (e.g. unlikely to revert), change in context or needs as an epoch shift. For the remainder of this dissertation, these definitions of disturbances, disruptions and perturbations will be used.

In addition to events which have already impacted the system, it is important to discuss events that merely threaten to impact the system. NIST (2009) defines a "threat" to be:

> "Any circumstance or event with the potential to adversely impact organizational operations (including mission, functions, image, or reputation), organizational assets, or individuals through an information system via unauthorized access, destruction, disclosure, modification of information, and/or denial of service."

The literature is somewhat confusing in the definition and classification of threats. If the NIST definition is used, a threat is always malicious. The International Standards Organization (ISO) defines threat more broadly as a "potential cause of an incident, that may result in harm of systems and organization" (International Standards Organization, 2011). The National Institute of Building Science (2011) and the Department of Homeland Security (Department of Homeland Security, 2012) distinguish between three types of threats; (1) natural events, (2) accidents, and (3) intentional acts to cause harm. Natural events are those that result strictly from natural forces, such as tornados, earthquakes and hurricanes. Accidents are those that result from unintentional human involvement, such as chemical spills and most fires. Intentional acts to

cause harm include both common criminal activity such as arson, and more serious attacks that would be considered terrorism. In her book Safeware, Leveson (1995) does not explicitly define a threat, but instead defines "hazards" to be a set of conditions inside a system that could negatively impact it. Construction engineering typically considers hazards to be both internal and external to the system, and either "dormant", such as poor soil conditions before construction has even started, or "armed", such as poor quality soil conditions upon which a building has been erected (MacCollum, 2007). For the purpose of clarity, this dissertation uses danger to mean a "set of conditions that have the potential to cause a perturbation, but have not yet impacted value delivery." Threats will mean dangers that exist outside the system, and hazards will mean dangers that exist inside the system.

Accidents, or "anthropogenic non-intentional events" according to FMEA, are a large concern for many sociotechnical systems. Humans play a large role in many complex systems. Even the unmanned vehicles used for maritime security have a significant human component. Although the word "unmanned" might give the impression that there are no humans involved, human operators perform command and control. Even in systems with a very high level of automation, humans have final decision-making authority. Since many stakeholders want to use unmanned technology as a force multiplier, the goal is to reduce the human-to-vehicle ratio as much as possible. However, Kilgore et al. (2007) points out that humans will still be required to perform contingency intervention, and will always be critical component of any "unmanned" systems in the foreseeable future. Ignoring the human aspect of these sociotechnical systems would be dangerous; some of the largest system accidents ever have been attributed, at least partially, to human error including Three Mile Island, the Challenger disaster, the Bhopal chemical spill, the Exxon Valdez spill, and Chernobyl (Woods, Johannesen, Cook, & Sarter, 1994). Even for unmanned aerial systems, it has been estimated that human operator error is at least partially responsible for between 60% and 80% of accidents (Leduc, Rash, & Manning, 2006). Many human factor researchers classifies system failures attributed to human errors using the Human Factors Accident Classification (HFAC) analysis (Shappell & Wiegmann, 2001). At the heart of this classification, is the human error itself, referred to as an "unsafe act". These acts are classified as being involuntary 'errors" (further decomposed into decision, skill-based or perceptual errors) and voluntary "violations" (either routine or exceptional). The errors themselves are often the result of "preconditions", the most common of which include the mental condition of the operators, procedural problems, supervisory problems and resource failures (Shappell & Wiegmann, 2002). Table 3-5 gives the result of a study, which classified the types of human errors and their causes for incidents involving the United States Marine Corps (USMC) and the United States Navy (USN) between 1991 and 1999.

TABLE 3-5: MISHAPS ASSOCIATED WITH EACH HFACS CAUSAL CATEGORY (FY 91-99) (SHAPPELL & WIEGMANN, 2002).

	USMC n=73	USN n=105
	Count (%)	Count (%)
Organizational Influences		
Resource Management	17 (23)	32 (30)
Organizational Climate	0 (0)	1 (1)
Organizational Process	19 (26)	39 (37)
Unsafe Supervision		
Inadequate Supervision	18 (25)	27 (26)
Planned Inappropriate Operations	9 (12)	11 (10)
Failed to Correct a Known Problem	4 (5)	10 (10)
Supervisory Violations	8 (11)	11 (10)
Preconditions for Unsafe Acts		
Adverse Mental States	57 (78)	79 (75)
Adverse Physiological States	18 (25)	27 (26)
Physical/Mental Limitations	7 (10)	11 (10)
Crew Resource Mismanagement	40 (55)	69 (66)
Personal Readiness	2 (3)	5 (5)
Unsafe Acts		
Decision Errors	36 (49)	64 (61)
Skill-based Errors	38 (52)	57 (54)
Perceptual Errors	23 (32)	28 (27)
Violations	22 (30)	33 (31)

3.4 "-ILITIES" FOR MAINTAINING VALUE

If we assume that complex systems, particularly systems of systems, operate in dynamic environments and have long lifespans, then we can also assume that they will encounter perturbations that threaten their value delivery. There are numerous non-functional requirements that may help systems maintain value delivery, in spite of contextual changes, over its lifetime including reliability (Downer, 2009), security (Furlani, 2009), safety (Leveson, 2002), survivability (Richards, Hastings, Rhodes, & Weigel, 2007), resilience (Hollnagel, Woods, & Leveson, 2006; Jackson, 2010; Westrum, 2006) and robustness (Smart, Amaral, & Ottino, 2008). Some of these qualities, such as resilience and robustness, are synonymous in the

English language, but researchers have defined them in the systems literature to describe distinct system properties. Furthermore, researchers have given different definitions to the same terms, such as resilience, creating inconsistency in the literature. In order for stakeholders to express what they need the systems to do, and for system architects to design systems that meet those needs, decision makers need to carefully choose which qualities they desire. Can a system be survivable in general or is survivability specific to a particular event or context? Should system architects design for survivability or robustness, and does it matter? Can a system be robust but not resilient? Is reliability necessary to achieve resilience? These types of issues will be explored in this section.

3.4.1 Resilience

Of all the "-ilities" relating to value sustainment, the term "resilience" is perhaps the most common. Sheard & Mostashari (2008) listed 35 different definitions of resilience, highlighting the fact that the definitions differ in the criteria they describe. Specifically, there are five criteria that may or may not be mentioned in the definition of resilience Figure 3-4. They are explained in more detail below:

System. Some definitions of resilience specify the type of system that the definition applies to. For example, Gunderson (2000) describes ecological resilience as the amount of disturbance that an ecosystem could withstand without changing self-organized processes and structures / without changing state. Whether or not that definition of resilience extends to other systems, depends upon how similar the other systems are to the ecosystem that Gunderson is referring to.

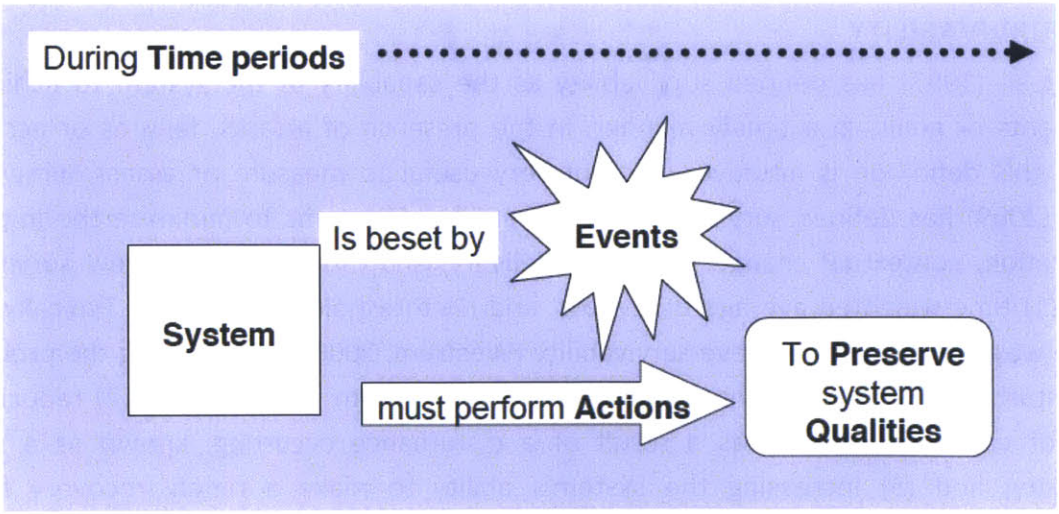

FIGURE 3-4: CRITERIA FOR ASSESSING RESILIENCE (SHEARD & MOSTASHARI, 2008)

Events. The definition of resilience may be specified to include only certain types of events such as whether they are endogenous or exogenous.

Actions. Certain definitions of resilience include specific actions that the system must take in order for it to be considered resilient. For example, the ability to change or adapt has been mentioned as a requisite for a system to be resilient (Hollnagel, et al., 2006).

Quality Preservation. According to some researchers, certain system properties or qualities must be preserved for a system to be considered resilient. For example, Allenby and Fink (2005) state that systems must retain their the capability of a system to maintain its functions and structure in the face of internal and external change and to degrade gracefully when it must.

Time Periods. The timing of the events, or the resilience to them, is often important to the definition of resilience. Some definitions of resilience specify a time period over which the resilience occurs, i.e. before, during, or after some event. Other definitions state the resilience is a time period itself. Gunderson (2000) defines resilience as "the time required for a system to return to an equilibrium or steady-state following a perturbation".

Sheard & Mostashari (2008) has shown that there is no consensus on the definition of resilience. In fact, certain definitions conflict with each other in that one might require a system to change to be resilient, while another forbids it. This means that it is not enough for a stakeholder to ask a system architect for a "resilient" system, since the meaning of resiliency is ambiguous at best.

3.4.2 SURVIVABILITY

Eillison et al. (1997) has defined survivability as the capability of the system to achieve its requirements or goals, in a timely manner, in the presence of attacks, failures or accidents. Although this definition is intuitive, it is not very useful to measure or assess survivability. Richards (2009) has defined survivability as the ability of systems to minimize the impact of finite-duration, contextual changes on value delivery, and introduced two new survivability metrics; (1) time-weighted average utility loss, and (2) threshold availability. Typically, there are three ways systems can achieve survivability (Westrum, 2006): (1) reducing the probability that a disturbance will impact the system, known as a system susceptibility; (2) reducing the amount of value lost directly as a result of a disturbance occurring, known as a system vulnerability; and (3) increasing the system's ability to make a timely recovery from a disturbance, known as system resilience. The three phases are also known as Type I, Type II and Type III survivability (Richards, et al., 2007) (Figure 3-5). Jackson (2010) refers to the same concepts as Richards, but instead refers to survivability as resilience and denotes the three

phases as (1) avoidance, (2) survival, and (3) recovery. However, the Oxford dictionary (2012b) defines resilience as "the capacity to recover quickly from difficulties" and "the ability of a substance or object to spring back into shape" which is more similar to Richard's definition than to Jackson's. To reduce confusion with the Oxford English definition, for the rest of this dissertation, resilience will refer to Richard's interpretation, i.e. the ability of a system to recover from a disturbance.

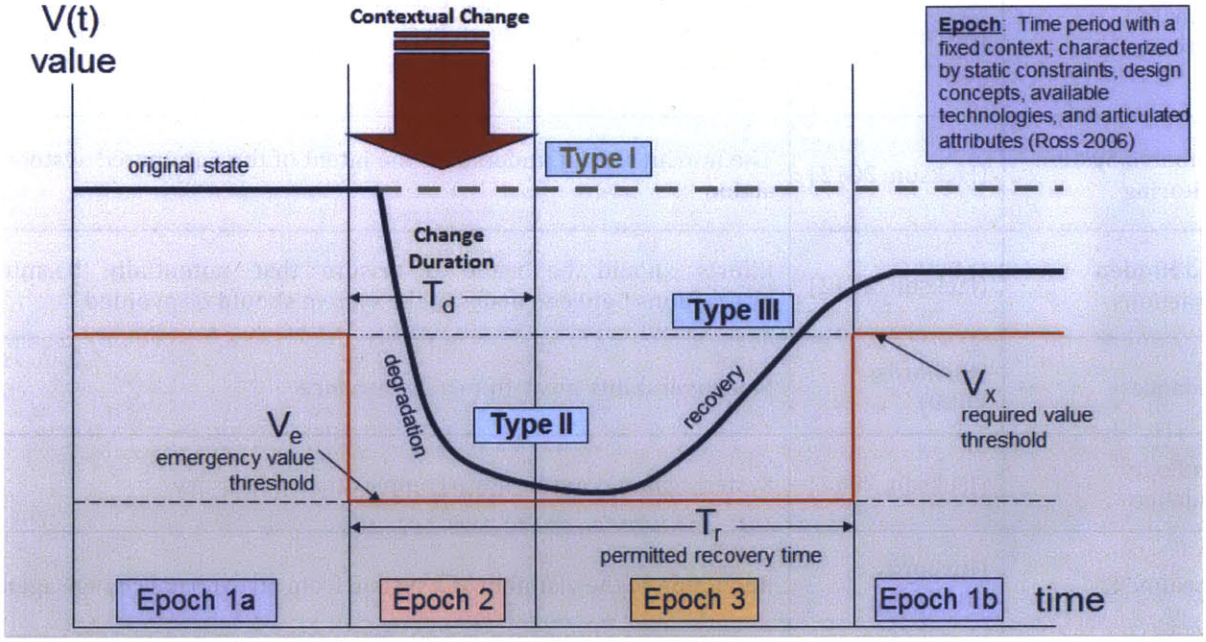

FIGURE 3-5: SURVIVABILITY DEFINED (RICHARDS, 2009)

3.4.3 SURVIVABILITY DESIGN PRINCIPLES

To give system architects and other decision makers tools to enable survivability in complex systems, researchers have identified and compiled lists of survivability design principles for systems. The design principles are listed in Table 3-6, and will be discussed more, along with the nuanced definition of what constitutes a "design principle" in Section 5.1.1.

TABLE 3-6: DESIGN PRINCIPLES FOR SURVIVABILITY

DESIGN PRINCIPLE	SOURCE	DESCRIPTION
Authority Escalation	(Jackson, 2012)	Authority over recovery operations would escalate in proportion to the severity of the threat
Automated Function	(Jackson, 2012)	Anytime there is a choice between having a machine perform a function or having a human perform it, the human is the preferable solution
Automated System Monitoring	(Jackson, 2012)	The human should understand the intent of the automated system's action
Avoid Hidden Interactions	(Jackson, 2012)	Efforts should be made to assure that potentially harmful interactions between nodes of the system should be avoided
Avoidance	(Richards, 2009)	Maneuverability away from a disturbance.
Complexity Avoidance	(Jackson, 2012)	Systems should not be more complex than necessary.
Concealment	(Richards, 2009)	Reduction of the visibility of a system from an external change agent
Containment	(Jackson, 2012; Richards, 2009)	Isolation or minimization of the propagation of failure
Context Spanning	(Jackson, 2012)	System shall be designed for both the maximum design level and the most likely disruption level.
Deterrence	(Richards, 2009)	Dissuasion of a rational external change agent from committing a disturbance.
Distribution	(Jackson, 2012; Richards, 2009)	Separation of critical system elements to mitigate local disturbances
Drift Correction	(Jackson, 2012)	If a system is drifting towards the boundary of survivability and there is evidence that a failure is approaching, then measures can be made either to avoid the threat or it can be diminished through corrective action.

Design Principle	Source	Description
Evolution /Reorganization	(Jackson, 2012; Richards, 2009)	Alteration of system elements to reduce disturbance effectiveness
Fail-safe	(Richards, 2009)	Prevention or delay of degradation via physics of incipient failure
Failure mode reduction	(Richards, 2009)	Elimination of system hazards through intrinsic design: substitution, simplification, decoupling, and reduction of hazardous materials
Hardness	(Jackson, 2012; Richards, 2009)	Resistance to a deformation
Heterogeneity	(Jackson, 2012; Richards, 2009)	Variation in system elements to mitigate homogeneous disturbances
Human in the Loop	(Jackson, 2012)	Systems should not have the final decision making authority in critical situations unless the pressure of time demands a quick decision
Human Monitoring	(Jackson, 2012)	An automated system should understand the intent of the human's action.
Independent Review	(Jackson, 2012)	Conduct an independent review to find latent faults in the systems that result in perturbations at a later time.
Informed Operator	(Jackson, 2012)	Humans should be informed regarding all aspects of the automated system.
Inter-node impediment	(Jackson, 2012)	All impediments to communication, cooperation and collaboration among nodes in the system should be removed
Inter-node interaction	(Jackson, 2012)	Every node, or element, of a system should be capable of communicating, cooperating, and collaborating with every other node.
Knowledge Between the Nodes	(Jackson, 2012)	All nodes of the system should be aware of what all the other nodes are doing.

Design Principle	Source	Description
Layered Defense	(Jackson, 2012)	System with two or more independent principles that address a single system point of vulnerability will be more survivable than a single principle
Loose Coupling	(Jackson, 2012)	System should have the capability to limit the ability of fail-ures to propagate from one component to the next in a system of many components.
Margin	(Jackson, 2012; Richards, 2009)	Allowance of extra capability for maintaining value delivery despite losses
Mobility	(Richards, 2009)	Relocation to avoid detection by an external change agent.
Neutral state	(Jackson, 2012)	States that human operators or other human agents delay taking action when there is an opportunity to survey the situation and make a more reasoned judgment about what the correct action might be
Preemptive Strike	(Richards, 2009)	Suppression of an imminent disturbance
Prevention	(Richards, 2009)	Suppression of a future or potential future disturbance.
Redundancy	(Jackson, 2012; Richards, 2009)	Duplication of critical system functions to increase reliability
Repair	(Jackson, 2012; Richards, 2009)	Restoration of system to improve value delivery
Replacement	(Richards, 2009)	Substitution of system elements to improve value delivery

3.4.4 Robustness and Value Sustainment

Robustness is another quality that is closely related to survivability. The Oxford dictionary (2012c) defines robustness of a system or organization, as *being able to withstand or overcome adverse conditions,* but like the term "resilience", there are numerous, different definitions in the technical literature. The "Taguchi" method, popular in industrial engineering for increasing

product quality, defines a robust design as one which delivers a strong "signal" regardless of external "noise" and with a minimum of internal "noise" (Taguchi & Cariapa, 1993). The concept of "signal" and "noise" relates to what the design is trying to do (the "signal") and external and internal disturbances that interfere with that ("noise"). During a workshop on systems engineering, Ross, Rhodes and Hastings (2008) noted that Dr. Marvin Sambur, Assistant Secretary of the Air Force for Acquisition, defined "robustness" as:

- Capable of adapting to changes in mission and requirements;
- Expandable/scalable, and designed to accommodate growth in capability;
- Able to reliably function given changes in threats and environment;
- Effectively/affordably sustainable over their lifecycle;
- Developed using products designed for use in various platforms/systems; and
- Easily modified to leverage new technologies.

According to this definition, a robust system also needs to be adaptable, scalable, reliable, sustainable, and modifiable. That definition is not only ambitious, but also somewhat ambiguous. For example, what does "easily modified" mean? Is it "easy" because it takes a small amount of time, or is it easy because it does not require much effort? Are changes that have to be made by external entities still considered "easy"? For this reason, Ross and Rhodes (2008a) defines value robustness as "the ability of a system to continue to deliver stakeholder value in the face of changing contexts and needs", which may be achieved through various sub-"ilities" such as modifiability, reliability, scalability, etc.

Beesemeyer (2012) examined the issue of value sustainment in his Master of Science dissertation at MIT, where he clarified the definition of robustness, while distinguishing it from similarly defined "-ilities" (Figure 3-6). According to Beesemeyer, a system *is* something, and *does* something, which can be described by a set of system parameters and outcome parameters, respectively. For example, a car can be described by a set of system parameters, such as the fact that it has a V8 engine, and output parameters, such as the fact that it can reach 150 mph. Robustness can be defined as a systems ability to maintain its output, regardless of change in the system or context. Suppose the car is designed to be able to reach 150 mph along a flat road. If the road becomes bumpy, and the car can still reach 150 mph, then it is robust. If one of the tires is replaced with a spare, and the car can still reach 150 mph, then it is also robust. Robustness makes no claim about value delivery, and the distinction between output and value is important. If the car owner suddenly decides that he/she needs to drive at 200 mph and the car cannot do this, one would not say it is because the car is not robust because the car can still maintain its output parameter, even though the stakeholders needs have changed. This distinction separates robustness from value robustness.

Beesemeyer (2012) defines "value sustainment" as the ability to maintain value delivery in spite of epoch shifts or disturbances.

3.4.5 Changeability and Versatility

If stakeholders require that a system does something it was not designed to do, either because the context has changed or because their needs have changed, then the system will need to be either versatile or changeable to be value robust. A system is said to be *versatile* if it can provide an output that it was not designed to provide, without changing its system parameters. For example, even with a static hardware configuration, most personal computers can be programmed to provide additional, high-level functionality through software. A system is said to be changeable, if its system parameters can be altered to achieve new output parameters.

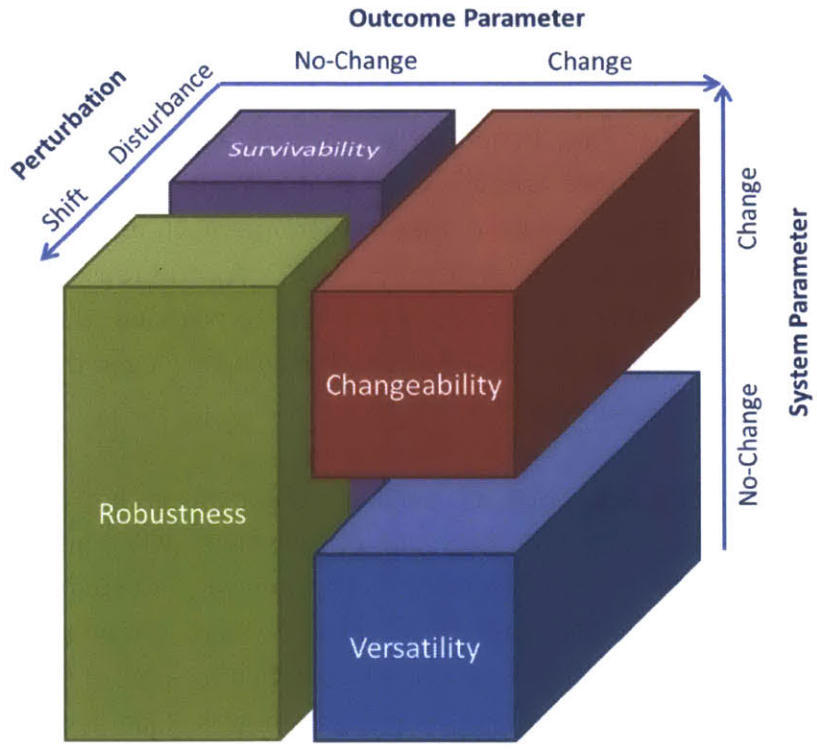

FIGURE 3-6: THREE-DIMENSIONAL PERTURBATION-SYSTEM PARAMETER-OUTCOME PARAMETER SPACE (BEESEMYER, 2012)

3.4.6 Robustness and Survivability

Both robustness and survivability describe a system's ability to prevent, mitigate and recover from harmful changes, but Richards (2009) explains that they differ only in the duration of the change. If the system maintains value delivery in spite of a perturbation of short duration, then it has exhibited survivability. If, on the other hand, the system avoids, mitigates or recovers from a long period between epoch shifts, then the system is robust. A system is considered

robust if it is effectively insensitive to changes in context, and survivable if it is effectively insensitive to disturbances and disruptions. The key difference between these two centers around the ambiguous difference between context change vs. disturbance or disruption. Of course, if the original context never resumes after the perturbation has occurred, then the system would be robust if it continues to provide acceptable value while it mitigates the effect of the "permanent" perturbation (Figure 3-7).

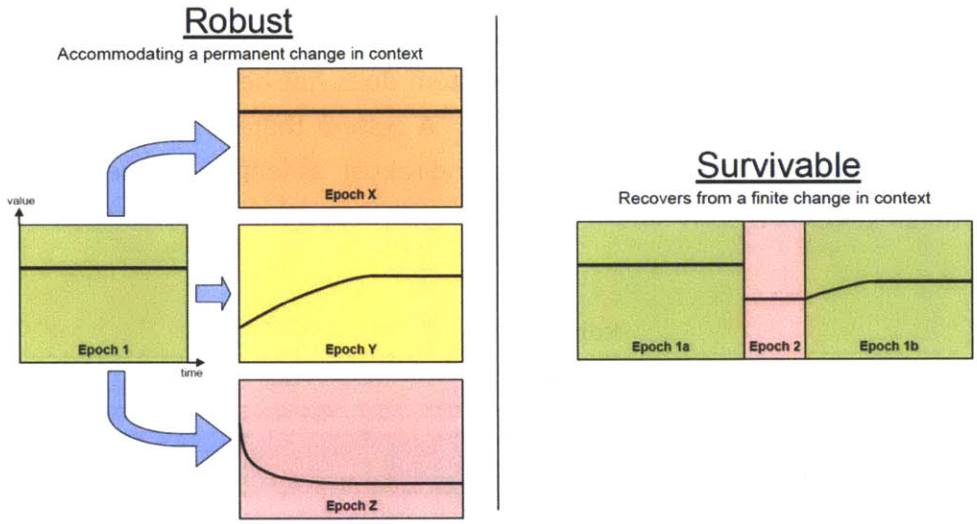

FIGURE 3-7: ROBUSTNESS VS. SURVIVABILITY
(ADAPTED FROM RICHARDS, 2009)

In his doctoral dissertation, Richards (2009) showed that survivability can be considered a special case of robustness, by demonstrating that if enough perturbations of short duration repeatedly occur, then the sum of these perturbations can be considered a new context itself, even if the context between perturbations temporarily resumes back to the original (Figure 3-8).

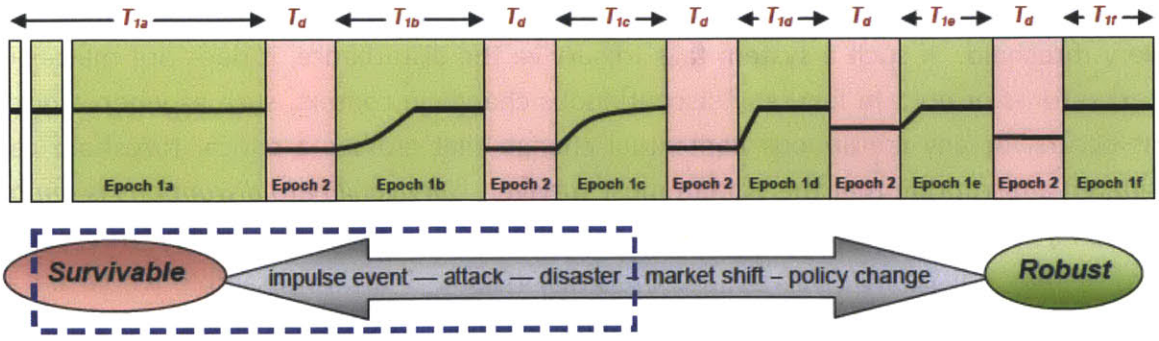

FIGURE 3-8: CONTINUUM BETWEEN SURVIVABILITY AND ROBUSTNESS
(RICHARDS, 2009).

While Richard's definitions are logically consistent, one could also argue that the argument is symmetric, that is, robustness is a special case of survivability. Suppose there is a perturbation of some duration T. If T is a context shift, (i.e. a change in context that is not temporary (Beesemyer, 2012)), then the system has exhibited robustness. However, there must exist some period τ, where $\tau < T$, such that τ is considered "short". Then, in order for the system to be robust during T, it must have first survived the context change that occurred during period τ (Figure 3-9). If the system did not survive the period τ, then it would not exist and be able to provide acceptable value to stakeholders during the period $T - \tau$ in order for it to be robust over period T. The corollary is not true however. A system does not need to be robust to some perturbation *X*, in order for it to be survivable to *X*. A system that can only withstand short duration perturbations can be survivable without being robust, as long as the perturbations are not frequent enough that they could be considered together as a new context with fluctuating conditions.

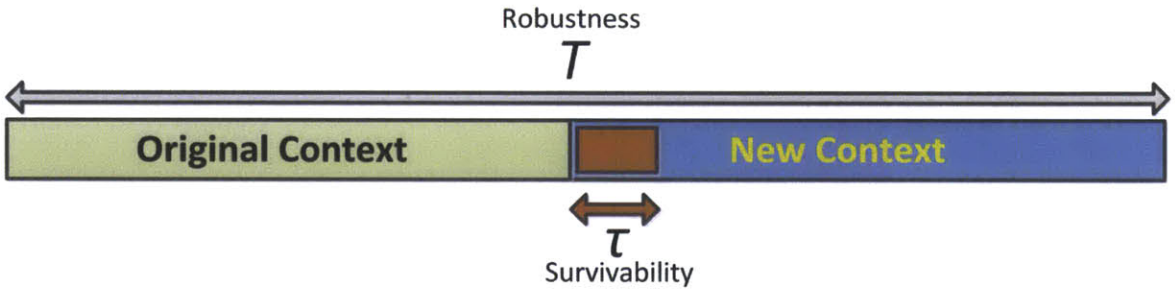

FIGURE 3-9: CONTINUUM BETWEEN SURVIVABILITY AND ROBUSTNESS

To illustrate this concept, consider the perturbation of a rise in the price of gasoline that threatens the value delivery of a system by increasing its operational cost. The rise in gasoline prices may be temporary or long lasting, depending on whether or not the prices eventually fall again, and this determines whether or not the system is considered to be robust to changes in gasoline prices or not. Regardless, while the price is high, the system has to survive that particular disturbance, i.e. the system must not fall below some minimum acceptable value delivery threshold. If such a system fails to survive the disturbance, it does not matter if the context returns or not. In fact, any discontinuous change in context, such as when a discrete-event occurs, or any continuous contextual change that crosses a critical threshold can be considered a disruption that the system must survive. *Survivability to disruptions is important since even if the system is able to provide value in a new, long duration context (thereby being robust), it may not be able survive the transition* (Figure 3-10). An example would be a 911 emergency response system that operates under a context where it gets positional information about its calls through the landline telephone system. As cell phones become more popular, the stakeholders demand that the system track the location of the mobile calls as well. This

requires the emergency response network to integrate with the cell phone carrier's GPS, which requires a fundamental change in networking components. This new demand may be a new context that requires a change in some of the networking technology. To make the transition to include mobile phone tracking, the entire system may have to go offline for several hours, possibly several days, while the underlying technology is upgraded. This is a temporary disturbance, but it may be unacceptable if the system cannot survive it. Thus, even though the system could have provided value under the new context and thereby be robust, it cannot survive the transition to that new context, and thus will never get the opportunity to provide value.

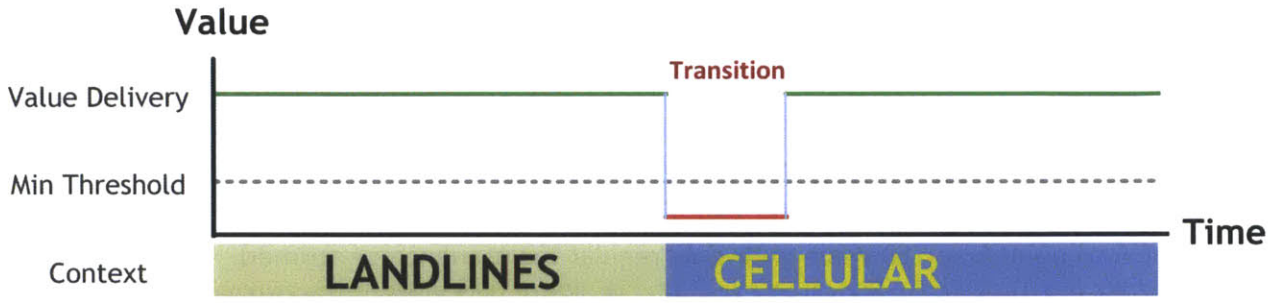

FIGURE 3-10: SYSTEM NOT SURVIVING TRANSITION BETWEEN TWO CONTEXTS

3.4.7 SURVIVABILITY TO PERTURBATIONS

Complex systems that exist over long periods are eventually going to encounter disturbances or disruptions that may threaten their delivery of value to stakeholders. For a maritime security system of systems, perturbations will likely include large increases in ships passing through the AOI, bad weather and labor shortages. Less likely, but perhaps more dangerous would be hostile attacks from criminals, terrorists or other hostile nations. To ensure that the systems continue to provide value to the stakeholders, the system architects need to make their systems avoid, mitigate or recover from these perturbations. Ideally, stakeholders want system architects to design a system that provides acceptable value during and after any perturbation it may encounter. Unfortunately, this is impossible. *There is no such thing as a system that is survivable in general, since there is no way of knowing that a system will be able to provide value no matter what.* According to a FEMA report, aircraft strikes were taken into consideration during the design of the World Trade Center, but reliably designing a system to survive an aircraft strike of the largest planes now or in the future, may be impossible (McAllister & Corley, 2002). At best, a system architect can state that a system will survive X, where X is some list of perturbations that are known in type and intensity. However, stakeholders and/or system architects must generate this list of X perturbations and accept the risks of not being survivable to any perturbation not included in X. Once the stakeholders

construct the list of X perturbation, system architects have the responsibility to implement and test appropriate survivability design principles and strategies and verify whether the system is survivable to all of the perturbations in X or not. Unfortunately, as X grows, various constraints will make it impossible to implement survivability strategies for every perturbation, so system architects and stakeholders must be willing to make tradeoffs and accept risks. However, the causes and effects of perturbations are typically not unique, and a strategy to survive perturbation X, may also help the system survive similar perturbations Y, Z. For this reason, research has looked at developing taxonomies of perturbations (Beesemyer, 2012; Mekdeci, et al., 2012).

3.4.8 RELIABILITY

The IEEE (1990) defines "reliability" as the ability of a system or component to perform its required functions under stated conditions (i.e. context and needs) for a specified period of time. Note that this definition does not mention changing conditions, only that the conditions are stated. A car would be reliable, for example, if it starts every morning and is able to drive from point A to point B for 10 years with only regular maintenance performed. If after only a year of driving the car fails to start, then the owner would probably consider the car unreliable. However, if the car fails to start because it suffered damage in a serious accident, then it would not be unreliable. Instead, the owner might say that the car did not survive the accident, and hence, the car was not survivable.

3.4.9 VIABILITY

The Oxford dictionary contains several definitions for "viability". It defines "viability" generally as *capable of working successfully; feasible,* and also defines viability relative to the following three life sciences disciplines:

> Botany: (of a seed or spore) *able to germinate.*
>
> Biology: (of a plant, animal, or cell) *capable of surviving or living successfully, especially under particular environmental conditions.*
>
> Medicine (of a fetus or unborn child): *able to live after birth.*

All of these definitions of viability are slightly different. Germination is the process of sprouting from a seed, which implies that the botanical definition of viability requires that the viable entity have the ability to change. The biological definition mentions that viability is "especially" relative to a particular context. The medical definition of viability implies that some level of autonomy is required, since after birth, the fetus is no longer a part of its mother's body.

3.4.9.1 Viable Systems in Management Cybernetics

The term "viability" has been described in the management cybernetics literature for organizations as "the survival or preservation of identity in a changing environment" (Dijkstra, 2007). Beer (1984) developed the Viable System Model (VSM), which describes how an organization can exist and maintain its identify in a dynamic environment.

The basis behind the VSM is that a viable organization must satisfy Ashby's Law of Requisite Variety (Ashby, 1962), by maintaining adequate control mechanisms to cope with uncertainty in the environment. Beer defines several criteria for viability, two of which are that a viable system is independent (i.e. autonomous) and has the ability to change (Yolles, 2000). The VSM applies to organizations, organisms and other entities that need to be self-sufficient, and whose stakeholders are part of the system itself. In an organism, for example, there is no distinction between the life of the "stakeholders" and the life of the organism itself, since they are one and the same thing. This is in contrast to an engineered system, which is built to provide value to stakeholders who are often external to the system. In the next chapter, the criteria of independence and the ability to change are examined to see if they are necessary for engineered systems to be considered viable as well.

3.5 Summary

This chapter began by defining some basic terms such as system, value and context, which form the foundation of many of the concepts in this dissertation. The usage of the term *viability* in the life sciences and management cybernetics disciplines was presented, and some related '-ilities" from the literature, such as survivability, reliability and resilience, were discussed. However, it was shown that there is a considerable amount of inconsistency in the literature as to precise definitions of some of these "-ilities". The concept of survivability was discussed in detail, since any system that is expected to encounter perturbations in its lifetime, will likely need to be able to survive to be viable. Since the eventual goal is to assist system architects make systems that provide acceptable value to stakeholders, existing survivability design principles from the literature were presented as well. In the next chapter, viability for engineered systems will be defined and compared with the related "-ilities" discussed in this chapter.

4 ENGINEERING FOR VIABILITY

In the previous section, the fundamental concepts of value delivery in systems and systems of systems was defined and explored, along with various system qualities, also known as "-ilities", for maintaining value delivery. Amongst the "-ilities" discussed, was "viability", which was defined for biological systems and organizations. In this chapter, the concept of "viability" is defined for engineered systems and compared with related "-ilities" already defined in the technical literature.

4.1 DEFINING VIABILITY FOR ENGINEERED SYSTEMS

The problem that needs to be solved, as described in Chapter 1, is that stakeholders need to design systems, particularly systems of systems, which will continue to provide acceptable value, for their intended life. While many of the "-ilities" described in the previous chapter may help the system achieve the goal of "staying alive", they all introduce a condition that may be unnecessary for certain systems and contexts. The closest "-ility" that would describe the quality that system architects are looking for would be value sustainment, which is the ability of a system to continue to provide acceptable value to stakeholders in spite of disturbances or epoch shifts. However, a simple engineered system, such as a stereo, that is not expected to experience changes in context or stakeholder expectations does not need to be value sustainable to provide acceptable value to its stakeholders, if those stakeholders are not concerned with perturbations or context shifts. Therefore, to provide value over a lifetime, a system must be value sustainable if it is operating within a dynamic context, or it does not have to be if the context remains fixed (either naturally or through some influence by other stakeholders). Furthermore, unlike other '-ilities" such as scalability, determining if a system will provide value in the future involves uncertainty. A system architect can prove that a design is scalable, for example, but for most systems, it is very difficult to be 100% certain that a particular design will provide adequate value over its entire lifetime, especially when the system is a SoS operating under dynamic contexts and diverse stakeholders. To address these gaps, *viability* for engineered systems is defined as the *likelihood that an engineered system will provide acceptable value to its stakeholders, over its life era*. The life era is both the expected time the system needs to last, as well as a sequence of epochs that it is expected to encounter (known as the "system era"). With this definition, we can say that an engineered system is *viable* if it is expected to provide acceptable value to its stakeholders, over its life era, and not viable otherwise.

There are several important concepts included in this definition of viability, which are important. They are:

Viability is Applicable to All Engineered systems. The concept of viability introduced in this chapter is applicable to all engineered systems, whether they are traditional, monolithic systems, or large systems of systems. However, this concept of viability is not intended for other systems, such as natural systems.

Viability is Subjective. Whether a system is viable or not, is determined by how well the outputs of the system are likely to satisfy stakeholder needs. Different stakeholders have different preferences, so the same system could be viable in the same context for one stakeholder, but unviable for another simply because the stakeholder needs are different.

Viability Is Dynamic. Viability is a prediction about whether the system will provide acceptable value to its stakeholders over its *life era.* What constitutes the life era is a prediction made by the stakeholders at the time viability is assessed. At the design stage, the life era may be determined to be a particular set of epochs, but as the system is implemented and exists, its life era (or what is left of it) may start to change, as the context and/or stakeholder needs change.

Viability is Relative. Although this research does not attempt to define metrics for viability, a system can be more or less viable than another system, or to itself if something changes, since viability is a likelihood. The more likely that a system will provide acceptable value to its stakeholders over its life era, the more viable it is. Viability can be *enabled* by something, if that something increases the likelihood that a system will provide value to its stakeholders over its life era. The objective of viability strategies, introduced in Chapter 5, is to enable viability.

Viability does not mean Existence. It is possible for an engineered system to exist, for a finite period of time, without being viable. This is because the system was viable at some point and implemented, but became unviable as time progressed. Meanwhile, the system might be producing value, perhaps even acceptable value, and therefore it "exists", but it is expected to fail before its life era ends, and is no longer considered viable accordingly. An example of the difference between existence and viability would be an SoS whose viability depends on a constituent system providing a critical service. The SoS could be viable and running successfully for years when one day the constituent system announces that it is planning to withdraw from the SoS in six months' time. Without a suitable replacement, this announcement means that the SoS will not be able to provide value after the constituent system leaves, and will effectively "die" as a consequence. In other words, the system became unviable when the announcement was made, even though it could exist and provide value for the next six months until the constituent system leaves.

Viability is a Prediction. Viability is a prediction about whether the system will continue to provide value to its stakeholders over its life era, which may involve a considerable amount of

uncertainty and risk. If the system is complex, there may be some uncertainty about its outputs. Similarly, there may be additional uncertainty about the context and needs of an engineered system that is expected to operate in a dynamic environment over a long period. When both of these sources of uncertainty are taken together, it can make assessing viability very difficult.

Viability is not Invulnerability. A system can be viable, even in a context where there are threats and perturbations that it may not be able to survive, as long as the stakeholders are willing to accept the risk. For example, a car can be considered viable if it has enough safety features that allow it to avoid and survive certain car accidents and weather disturbances, but that does not mean that cars have to be completely invulnerable to any and all possible problems. A car owner assumes that there is a certain level of risk in operating a car, and the manufacturers also assume a certain level of risk as well, and cars are designed, built, bought and operated with those risks being accepted. The Ford Pinto fuel tank controversy (Lee, 1998), is an infamous case where Ford executives were accused of knowing about and accepting a particular vulnerability in the design of one of its vehicles, by performing a cost-benefit analysis that included the risk of lawsuits resulting from deaths caused by the design flaw. This type of risk-taking may also highlight the fact that viability of a mass-produced system such as a car or cell phone, is likely to be different that the viability of a massive, unique SoS like Facebook or a MarSec. In mass-produced systems, stakeholders can spread the risk of fatal problems over many units.

Viability does not Require Independence. According to the Viable System Model (VSM), a system must retain its independent existence in order for it to be viable (Hildbrand & Bodhanya, 2011). However, independence is subject to interpretation. Most, if not all, systems require certain contextual conditions to exist and some of these conditions include the presence of, and actions by, certain entities. For example, businesses without clients or customers would not be able to exist, and therefore would not be viable. The VSM handles this by including assuming that adequate resources and services are part of the environment, at least to some extent. A viable system, according to cybernetic management, can handle perturbations in the environment, such as a reduction in available resources, but it cannot be viable if critical resources disappear entirely. If some of these critical resources are in fact other systems, then are the organizations described by the VSM independent? It depends on how independence is defined. In terms of engineered systems, there are ten levels of automation, ranging from a "dumb" Level 1 system where the human does all the decisions and actions, to a fully automated Level 10 system that has no human in the loop whatsoever (Parasuraman, Sheridan, & Wickens, 2000). The fact that Level 1 engineered systems exist and provide acceptable value to stakeholders, show that engineered systems can be viable without being

independent. For this reason, independence of components will not be a requirement for viability of an engineered system.

Viability does not Require the Ability to Change. Another requirement for a system to be viable, according to the VSM, is that it has the ability to change, or adapt, in response to changes in the environment. Like "independence", the exact definition of what constitutes a change or adaptation is not clear. If an owner replaces a working engine in a car with a more powerful version, did the car change? Yes. What if the working engine was replaced with another working engine of the exact same type and mileage, did the car change? That depends on how change is defined and who the stakeholder is. To the car manufacturer who provides a warranty on the car, then the car did change since the engine is not the original one that was manufactured with the rest of the car. To the driver, the car did not change, since the car will operate exactly the same way. While physical changes might be more obvious, changes in behavior are not. Suppose the car could switch between Front Wheel Drive (FWD) and All Wheel Drive (AWD), with FWD being more fuel efficient and AWD being safer when the roads are slippery. Is the ability to switch from AWD to FWD and vice versa depending on road conditions, count as change? Is this adaptation? According to the VSM, yes, because the system is altering its behavior is response to changes in the environment, in accordance with Ashby's Law of Requisite Variety. However, the VSM model assumes that there is variety in the context, for which the ability in the system is necessary. For a large SoS like Facebook, it's hard to imagine it being considered viable without the ability to adapt in some way to meet emerging technological and social changes. However, for simpler systems such as a stereo, it is not unreasonable to expect the context to remain fairly static and thus the requirement for change or adaption is not necessary for viability of engineered systems.

With viability defined this broadly, every system should be designed to be viable, but how do systems achieve viability? Does viability require robustness? How does this definition of viability relate to other definitions of similar properties such as robustness, survivability and reliability?

4.2 Viability and Survivability

As mentioned in Section 3.4.9, the biological definition of viability, according to the Oxford Dictionary is *capable of surviving or living successfully, especially under particular environmental conditions.* This definition suggests that a viable system is one that is survivable. This section looks deeper into the concept of survivability for engineered systems and defines its relation to viability.

4.2.1 General Survivability

The Oxford dictionary (2012d) defines survivable as being *able to be survived; not fatal*. An important distinction, however, that separates survivability from other "-ilities", is that the adjective "survivable" is defined relative to an "accident or ordeal", and not to an entity. In the English language, a car would not be considered survivable, however a car accident may be. This is in contrast to other, similar adjectives such as robustness. In the English language, one would say that a car is robust, not that the car accident is robust. For engineered systems, survivability has been defined as a system property, but it is relative to a particular disturbance or context. One can say, therefore that an engineered system, such as a car, is survivable, but only to some disturbance (or disturbances) in particular.

4.2.1.1 Survivability and the Disturbance Lifecycle

According to (Richards, 2009), survivability of an engineered system is defined relative to a disturbance lifecycle that consists of three stages. The first stage of the disturbance lifecycle is before the disturbance impacts the system, where survivability strategies are applied to reduce susceptibility to the disturbance. The second stage is when the disturbance has impacted the system, and the system is surviving by mitigating its effects. The final stage is after the disturbance has ended, where the system is surviving by recovering from the impact. Richards (2009) classifies survivability strategies according to the phase of the disturbance at which they are applied. Type I strategies are used to reduce susceptibility to the perturbation, Type II strategies are used to mitigate the perturbation, and Type III strategies are used to recover. At first glance, it might seem that classifying survivability design principles as Type I, II or III is somewhat arbitrary. Suppose there is a thunderstorm that is between a plane's current location and its final destination. If the plane continues on its original flight path, it will encounter the thunderstorm and possibly crash as a result (i.e. not survive). Not wanting to risk the crash, the pilots decide to fly around the thunderstorm. At first, this seems to be a textbook example of the *avoidance* survivability design principle as proposed by Richards. However, suppose that flying around the thunderstorm added an additional hour to the flight time, which reduced the value the plane delivered to its stakeholders. One could argue that the thunderstorm has already impacted the system, and thus flying around is simply mitigating the value loss, making this a Type II strategy. There seems to be ambiguity as to how and when the strategy is enabling survivability.

4.2.1.2 Non-survivable Events and Terminal Conditions

In the way survivability is currently defined in the literature, Type I survivability design principles also pose an interesting question. Type I survivability principles are designed to

reduce susceptibility to a particular disturbance. Some strategies, such as prevention, try to ensure that the disturbance never occurs at all. In some ways, it seems counter-intuitive that a system could survive something that has not happened. For example, if a country practices deterrence to a nuclear attack by stockpiling their own arsenal of nuclear weapons, can that country claim that they are "surviving" nuclear attacks? What if nuclear attacks are non-survivable? How does one survive something that cannot be survived? For any engineered system, there are non-survivable events. A non-survivable event is an event that marks the death of the system, i.e. the point at which the system has failed to deliver adequate value to its stakeholders and recovery is not possible. For example, direct rocket hit is often a non-survivable event for a small UAV since it will likely explode and not be recoverable. Similarly, a major terrorist bomb detonating inside of a plane is typically a non-survivable event for an airport security system. Richards implicitly described some non-survivable events when he introduced the concept of minimum value delivery thresholds. If a minimum value delivery threshold is reached, a non-survivable condition has been reached since the system has failed to meet stakeholder expectations.

A condition can be considered terminal if the system cannot survive the moment that condition is reached (called terminal events). The system may still technically exist in some form or another and even provide some value for a relatively short period of time, but because of the terminal event occurring, there is no hope of survival beyond a certain period of time. This means that terminal conditions cannot be mitigated or recovered from and instead, systems must try to prevent them from occurring.

As with value delivery in general, what constitutes a terminal condition for a system is subjective and determined by the stakeholders. For example, taking the entire system offline for repairs may be acceptable for cable company call centers, but it is likely a terminal event for 911 emergency call centers.

4.2.2 Inevitable Events

While some events cannot be survived, others cannot be avoided. The impacts of certain events are inevitable and these are called *inevitable events*. For example, no matter what a system or its stakeholders do, the day will turn into night, and the night will turn into day. Although the events themselves may be unavoidable, the system can still avoid its effects. Earthquakes are unavoidable for stationary buildings, but implementing certain survivability techniques can make buildings avoid their potentially terminal effects for all but the most severe cases.

4.2.3 The Single Perturbation Fallacy

A system cannot survive a non-survivable event. However, systems may be able to implement Type I survivability strategies to prevent these events from occurring, or avoid them all together. This may seem confusing, however, since someone may assume that by implementing a Type I survivability strategy to avoid a non-survivable event, the system is surviving it. The ambiguity lies with the fact that there is more than one disturbance affecting, or threatening the system and what the system is surviving is not the same disturbance as the one that it is preventing or avoiding.

Suppose a small UAV is flying over hostile territory. Due to its small size, being struck by an anti-aircraft missile is a terminal event. To compensate, the UAV operator flies the UAV at a high altitude, outside the range of the anti-aircraft system. The UAV successfully performs its mission over hostile territory without any incident. The UAV commander can confidently say the UAV survived and its survivability was (likely) due to the implementation of the avoidance design principle. The UAV did not survive an event that never happened, that is, it did not survive being struck by an anti-aircraft missile. Rather, the UAV survived flying a mission over hostile territory (disturbance X) due in part to the fact that being struck by a missile (disruption A) never happened. Flying a mission over hostile territory and being struck by a missile are related in that A is a possible effect of X, and X is a possible cause of Y, but they are not the same perturbation.

The example of a UAV flying a mission over hostile territory and avoiding being struck by a missile highlights a major source of confusion with the current definition of survivability and its related design principles. The current definition of survivability assumes a single disturbance that has a lifecycle as shown in (Figure 3-5). In reality, perturbations rarely occur in isolation. Instead, perturbations are caused by previous events and they themselves often lead to further perturbations (the effects). In general, survivability to perturbation X, is not about reducing susceptibility to X, it is about continuing to provide acceptable value once X occurs. Since a terminal event is defined by reaching a non-survivable set of conditions, one can say that surviving perturbation X is really about preventing all the possible terminal events that are possible effects of perturbation X. Perturbation X itself cannot be a terminal event, otherwise the system would not be able to survive it.

4.2.4 Multiple Causes and Effects

To illustrate an example of how a perturbation can have multiple causes and effects, Figure 4-1, shows some possible causes for why a perturbation might disconnect a server from the Internet, and some of the immediate effects that follow. Qualitatively, it would seem that as the number of causes and effects grows, so too does the likelihood and impact of that

perturbation, respectively. Perturbations often do not occur in isolation, and the effects of many perturbations are often the cause of others.

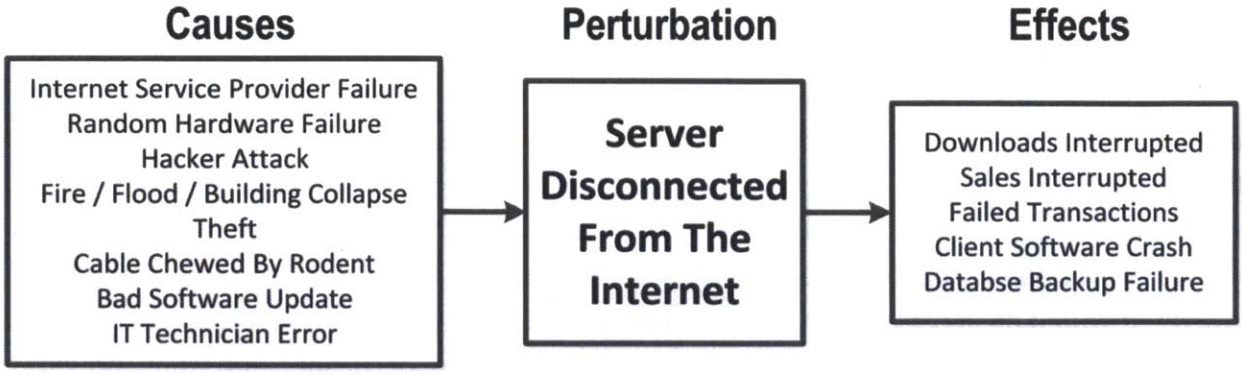

FIGURE 4-1: MULTIPLE CAUSES AND EFFECTS OF A SINGLE PERTURBATION

It is important to separate the immediate effects of a perturbation, from subsequent perturbations and effects. For example, lightning itself does not burn down forests, but the fires that the lightning starts, may. This distinction is important, because strategies to survive a forest fire are separate from the cause (lightning strike). Strategies for surviving a forest fire, such as using planes to fly overhead and drop fire extinguishing materials, will likely be effective regardless of whether the fire was started by lighting or some other perturbations (such as arson or careless camping). One strategy to avoid or survive a particular perturbation, may also help avoid or survive other perturbations as well (i.e. extinguishing a fire may help a system be viable towards both careless camping as well as lightning). However, system architects must take care that they are not wasting resources considering perturbations that are well outside the system boundary. A weak applicant pool may result in having incompetent operators, for example, but the job market is likely outside the system architect's control. Instead of trying to "avoid" an unfavorable job market, system architects should try to "survive" its effects (incompetent operators) by, perhaps, increasing automation or requiring closer supervision. Similarly, system architects should only consider perturbations whose effects are directly the result of the actions (or inactions) of the system. There are several effects of wasting fuel when UAVs are sent to the wrong location, for example. Some of them, such as the cost and the fact that the UAVs will be unavailable to perform their required tasks during that time, are of concern to the system architects because they affect the value the system delivers. Other effects, such as releasing CO2 into the atmosphere, are not of concern, unless they affect the system's ability to deliver value (due to some environmental constraints, perhaps).

4.2.5 VIABILITY AND SURVIVABILITY

Suppose system architects are tasked with designing a security system for an airport. A bomb brought aboard an aircraft is a terminal event, meaning that if it occurs, the security system will have failed to provide acceptable value to the stakeholders and will likely need to be replaced. Since a bomb brought aboard an aircraft is a terminal event, the system architects decide to prevent it from occurring by implementing various security measures such as having uniformed officers patrol the airport as a deterrence. To say that the airport is "survivable" because it has certain mechanisms to reduce susceptibility to an airport bombing is confusing. Obviously, a bomb has never been brought aboard, otherwise the system would not exist in its current form. However, it would be acceptable to say that the airport security system is viable since it has the necessary mechanisms to prevent this terminal event from occurring, and thus provides acceptable value to its stakeholders.

Viability is related to, but distinct from survivability. Survivability is about maintaining value delivery when the system experiences a disturbance, that is, a distinct change in context over a finite period. Viability is the likelihood of maintaining value delivery over a system lifetime, which typically includes multiple contexts and perturbations. If a system architect wishes to design a system that provides value over its entire lifetime, in spite of perturbations that may arise, then it will need to satisfy two criteria. First, the system must survive any inevitable events that threaten the value delivery of the system. Second, the system must prevent any terminal events from occurring. If the system fails to satisfy either criteria, then it can be considered unviable.

More often than not, a viable system will need to be survivable to perturbations that arise during its lifetime. However, it is possible that a system may be not to be survivable to any perturbations and still be viable, if those perturbations do not occur. A system is *fragile* to a particular context if its viability is dependent on the context remaining static. A fragile system does not need to be survivable or value robust, since those properties are defined by the ability to maintain value in spite of change. Shielding the system from contextual changes is a strategy for achieving viability, much in the same way a mother shields a newborn from perturbations it cannot survive on its own. This idea was first introduced by Ross (Ross, 2006), and will be expanded upon in more depth to produce new viability strategies in Chapter 7.

4.2.6 Viability and Other Value Sustainment "-ilities"

4.2.6.1 Viability and Value Robustness

Value robustness and survivability are very similar, and likewise, if the system can avoid contextual variations, then viability does not require robustness.

4.2.6.2 Viability and Reliability

Viable systems do not need to be survivable to a particular event, if they can avoid that event all together. However, do viable systems have to be reliable? The answer is yes, viability requires reliability. If a system is expected to provide value over its lifetime, then it need to work as intended over that period, meaning that it needs to be reliable.

4.2.6.3 Viability, Changeability and Versatility

Changeability and versatility are properties that allow a system the ability to meet new output expectations, should they be necessary. Neither property is required for obtaining viability during the design stage; however, both qualities may help systems remain viable if the context and/or stakeholder needs change. However, as Chapter 8 will show, care must be taken when designing a system that is changeable, since certain changes, even if made with good intentions, can cause a system to become unviable, particularly if the complexities of the component interactions are not well understood.

4.3 Viability of the MarSec

In this section, the concept of viability is applied to the MarSec to demonstrate how viability can be assessed for a system of systems and to set up a case study by which viability strategies can be examined and developed in later chapters.

4.3.1 Design Alternatives

When designing the MarSec, system architects will likely develop different design alternatives, some of which may include certain strategies for viability, and present them to the stakeholders for evaluation. Suppose there are four designs being considered:

Designs:

- Design I
    - 4 UAVs
    - 4 Patrol Boats

- Design II
  - 8 UAVs
  - 4 Patrol Boats
- Design III
  - 8 UAVs
  - 8 Patrol Boats
- Design IV
  - 12 UAVs
  - 8 Patrol Boats

4.3.2 Life Era Determination

The first step is assessing the viability of a particular engineered system, is to define the contexts that will form part of the expected life era of the system. This means defining "normal" operating conditions, if they exist, as well as contexts where there are perturbations that could jeopardize the value delivery of the system. Not every contextual factor is relevant, but those that may impact the attributes of the system should be specified. For the MarSec, there are numerous relevant contextual factors that will affect the attributes of the system, such as how many boats arrive in the AOI per unit time. There are also certain contextual perturbations that may arise, such as bad weather, which may jeopardize the value delivery of the system. It is not practical to consider every possible perturbation that may arise, so the stakeholders have to assume some risk by only considering certain perturbations within the life era. The choice of which perturbations should be considered is subjective and partially determined by the stakeholder's risk tolerance. By not including perturbations for consideration, stakeholders run the risk of approving a system that will not provide acceptable value when confronted or impacted by those perturbations. Similarly, by including perturbations that have a low probability of occurring, and are difficult to survive, stakeholders run the risk of rejecting systems that will provide acceptable value, and/or implementing unnecessary, but costly survivability strategies. Therefore, the consideration of which perturbations should be included is important, but beyond the scope of this research.

For the purpose of demonstrating the concept of viability, suppose that there are three stakeholders of the MarSec, named Sandy, Pat, and Carol. Also, suppose that there is a normal context that consists of boats arriving in the AOI every 640 seconds, on average. The stakeholders of the MarSec wish to consider the possibly of an epoch shift if a particular piece of maritime legislation is passed, which will double the traffic of ships through the AOI. In addition, Sandy and John wish to consider perturbations of political unrest, which may generate a terrorist attack that would threaten the value delivery of the system. Pat does not think that

political unrest will occur during the MarSec's life era. Enumerating the perturbations amongst the two epochs, yields the following four contexts:

- Context A ("Normal" conditions)
  - Mean boat inter-arrival period – 640 s
  - No political unrest
- Context B (High traffic)
  - Mean boat inter-arrival period – 320 s
  - No political unrest
- Context C (Political unrest)
  - Mean boat inter-arrival period – 640 s
  - Political unrest
- Context D (High traffic and political unrest)
  - Mean boat arrival period – 320 s
  - Political unrest

The expected life eras for the stakeholders are shown in Figure 4-2 and Figure 4-3.

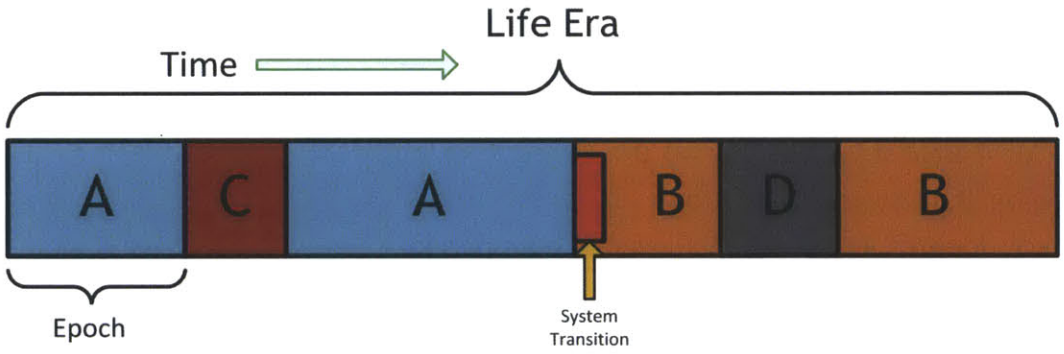

FIGURE 4-2: EXPECTED LIFE ERA OF THE MARSEC ACCORDING TO SANDY & JOHN

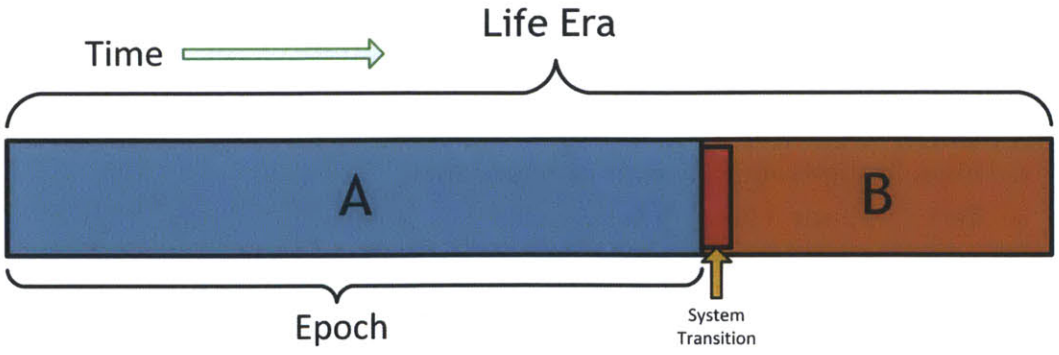

FIGURE 4-3: EXPECTED LIFE ERA OF THE MARSEC ACCORDING TO PAT

4.3.3 Threshold Determination

In addition to providing value under the contexts listed above, the all designs will require a network upgrade (system transition) in order to be able to provide value during contexts B and D. This is an additional, endogenous perturbation that must be considered. With this in mind, suppose that the stakeholders assign the following value thresholds to the four contexts described above.

Value Thresholds (Sandy):

- Context A, C
    - Identification rate ≥ 0.8
    - Terrorist Disarm Rate = 1.0
- Context B, D
    - Identification rate ≥ 0.7
    - Terrorist Disarm Rate = 0.7
- System does not need to provide any value during system transition

Value Thresholds (John):

- Context A, C
    - Identification rate ≥ 0.9 over 24 hour period.
    - Terrorist Disarm Rate = 1.0
- Context B, D, System Transition
    - Identification rate ≥ 0.8 over 24 hour period.
    - Terrorist Disarm Rate = 1.0

Value Thresholds (Pat):

- Context A,
    - Identification rate ≥ 0.8 over 24 hour period.
    - Terrorist Disarm Rate = 1.0
- Context B
    - Identification rate ≥ 0.7 over 24 hour period.
    - Terrorist Disarm Rate = 1.0
- System does not need to provide any value during system transition

4.3.4 Assessing Viability Over the Life Era

In order for the MarSec to be viable, it must be expected to provide acceptable value to its stakeholders over its life era. This involves not only being able to provide acceptable value

during each context (i.e. be survivable/robust for contexts A,B,C,D) but also being able to continuously provide value from one context to another including the system transition. This is one aspect where viability is distinct from survivability and robustness.

Suppose the performances of the three designs are assessed and are shown in Table 4-1.

TABLE 4-1: ASSESSMENT OF IDENTIFICATION RATE AND TERRORIST DISARMAMENT DURING EPOCHS

DESIGN	CONTEXT							
	A				B			
	ID Rate	Thresholds (Sandy, Pat/John)	Disarm Rate	Thresholds (Sandy, Pat/John)	ID Rate	Thresholds (Sandy, Pat/John)	Disarm Rate	Thresholds (Sandy, Pat/John)
I	0.88	0.9/0.8	1.0	0.7/1.0	0.78	0.8/0.7	1.0	0.7/1.0
II	0.95	0.9/0.8	1.0	0.7/1.0	0.86	0.8/0.7	1.0	0.7/1.0
III	0.99	0.9/0.8	1.0	0.7/1.0	0.96	0.8/0.7	1.0	0.7/1.0
IV	0.99	0.9/0.8	1.0	0.7/1.0	0.96	0.8/0.7	1.0	0.7/1.0
DESIGN	CONTEXT							
	C				D			
	ID Rate	Thresholds (Sandy, Pat/John)	Disarm Rate	Thresholds (Sandy, Pat/John)	ID Rate	Thresholds (Sandy, Pat/John)	Disarm Rate	Thresholds (Sandy, Pat/John)
I	0.88	0.9/0.8	0.88	0.7/1.0	0.75	0.8/0.7	0.88	0.7/1.0
II	0.95	0.9/0.8	0.95	0.7/1.0	0.95	0.8/0.7	0.95	0.7/1.0
III	0.99	0.9/0.8	1.0	0.7/1.0	0.99	0.8/0.7	1.0	0.7/1.0
IV	0.99	0.9/0.8	1.0	0.7/1.0	0.99	0.8/0.7	1.0	0.7/1.0

DESIGN	SYSTEM TRANSITION			
	ID Rate	Thresholds (Sandy, Pat/John)	Disarm Rate	Thresholds (Sandy, Pat/John)
I	0.88	None/0.8/0.7	0.0	0.0/1.0
II	0.95	None/0.8/0.7	0.2	0.0/1.0
III	0.0	None/0.8/0.7	0.8	0.0/1.0
IV	0.0	None/0.8/0.7	1.0	0.0/1.0

Viability of Design I. Design I is only viable to Sandy because of her relatively low thresholds. The system is not viable to the other stakeholders, highlighting the fact that designs have different appeal to stakeholders based on their preferences, and because of this, the viability of a system is not the same across stakeholders either.

Viability of Design II. Design II is not viable to Sandy and Pat, because it cannot provide adequate value during contexts C and D. However, Design II is viable to Pat because the expected life era for Pat does not include contexts C and D. This highlights the fact that the determination of life era can be stakeholder dependent, and distinct from value preferences.

Viability of Design III. Design III will provide acceptable value to all stakeholders in all contexts considered, and is therefore value sustainable to all stakeholders. However, according to John's preferences, Design III cannot survive the required system transition, during the epoch shift from A to B, and therefore is not viable. This distinction highlights one of the differences between being viable and being value sustainable.

Viability of Design IV. Design IV is viable according to all stakeholders.

4.4 SUMMARY

This chapter defined viability for engineered systems and described some unique characteristics that distinguish it from other "-ilities" already defined in the engineered systems literature and elsewhere. Although viable systems are often value sustainable, they can also be fragile, where they only provide acceptable value in contexts that remains fixed. The concept of "fragility" will be useful in Chapter 7, where new strategies for enabling viability will be developed. This chapter ended with an illustration of how viability can be assessed, and highlighted the difficulties associated with systems being able to provide acceptable value to multiple stakeholders and under multiple contexts. However, all systems should be designed to be viable and the focus of the next section will be to examine existing strategies in the engineered systems literature that may help enable viability for systems, and systems of systems in particular.

5 VIABILITY-RELATED STRATEGIES FOR SYSTEMS

Stakeholders want systems that are viable, that is, they will provide acceptable value over the system's lifetime. Since complex systems tend to have very dynamic environments, strategies for enabling viability of engineered systems will require that the systems survive certain perturbations and avoid others. Section 3.4.3 highlighted existing design principles that system architects can use to increase a system's survivability to perturbations that cannot be avoided, and reduce susceptibility to those that can. This chapter tries to explain why the literature is confusing as to what a design principle is, and resolves the issue by adapting the existing design principles into more useable design patterns and heuristics for enabling viability.

5.1 PROBLEMS WITH EXISTING VIABILITY-RELATED DESIGN PRINCIPLES

Systems that can expect to encounter perturbations, have to be survivable in order to be viable. In 2009, Richards developed a list of 17 design principles for survivability, which was followed by a list of "resilience" heuristics that Jackson compiled in his 2010 book *Architecting Resilient Systems*. Recently, Jackson transformed his list of heuristics into a hierarchy of about a dozen major survivability design principles and numerous sub-principles (Jackson, 2012). Since both Richards' and Jackson's design principles attempt to increase a system's ability to avoid, mitigate and recover from perturbations, they may help enable viability for systems that must encounter perturbations. Although there are a few design principles that overlap, the lists of design principles compiled by Jackson and Richards differ significantly (Table 5-1). The discrepancies between these lists show that there is confusion, or at least disagreement, in the literature as to what a design principle is, or which principles actually enable viability.

5.1.1 DEFINING DESIGN PRINCIPLE

There is not much consensus in the technical literature as to what exactly a "design principle" is. Fulcoly (2012) states that "a design principle is a principle that guides the design of a system or architecture in order to promote some quality (e.g. evolvability)". Wasson (2006) states that a principle is "a guiding thought based on empirical deduction of observed behavior or practices that proves to be true under most conditions over time". Richards (2009) refers to design principles as "concept-neutral strategies of architectural choice". Jackson (2012) uses the term "design principle" to refer to a broad range of abstract rules that when followed, produce concrete solutions which improve the system. Jackson quotes Lonergan (1992) who says that principles, like laws, are "invariant". Jackson further notes that design principles range in rigor from those that are mathematically provable to those that are just guidelines. Mathematically

provable design principles are relatively rare, and typically cover very specific types of perturbations. An example of a mathematically rigorous viability design principle would be the principle of redundancy. If only one operational component is necessary, then as the number of redundant components n increases, the probability of system failure, p_s, is related to the probability of each particular component failing, p_i, by

$$p_s = \prod_{i=1}^{n} p_i \tag{6}$$

TABLE 5-1: OVERLAP OF VIABILITY DESIGN PRINCIPLES IN THE LITERATURE

JACKSON (2012)		RICHARDS (2009)
DESIGN PRINCIPLES	SUB-PRINCIPLES	DESIGN PRINCIPLES
Absorption	Hardening	Hardness
	Margin	Margin
	Context Spanning	Not Represented
Physical Redundancy		Redundancy
Functional Redundancy		Heterogeneity
Localized Capacity		Distribution
Loose Coupling	Loose Coupling	Not Represented
	Containment	Containment
Repairability		Repair
Reorganization	Restructure	Evolution
	Regroup	Not Represented
	Authority Escalation	Not Represented
Layered Defense		Not Represented
Human in Control	Avoid Human Error	Not Represented
	Automated Function	Not Represented
	Human in the Loop	Not Represented
Complexity Avoidance		Not Represented
Drift Correction	Drift Correction	Not Represented
	Independent Review	Not Represented
Neutral State		Not Represented
Inter-Node Interaction	Inter-Node Interaction	Not Represented
	Inter-Node Impediment	Not Represented
	Informed Operator	Not Represented
	Knowledge Between Nodes	Not Represented

JACKSON (2012)		RICHARDS (2009)
DESIGN PRINCIPLES	SUB-PRINCIPLES	DESIGN PRINCIPLES
	Human Monitoring	Not Represented
	Informed Operator	Not Represented
	Knowledge Between Nodes	Not Represented
	Human Monitoring	Not Represented
	Automated System Monitoring	Not Represented
Avoid Hidden Interactions		Not Represented
Not Represented		Prevention
Not Represented		Maintenance
Not Represented		Preemption
Not Represented		Failure Mode Reduction
Not Represented		Mobility
Not Represented		Concealment
Not Represented		Avoidance
Not Represented		Fail-safe
Not Represented		Replacement

When system architects actually implement a design that has redundancy of components, it will work as intended, if the perturbation materializes. An example of a heuristic or a "rule of thumb" would be the principle of complexity avoidance, which states that a system can be made more viable if internal complexities are kept to a minimum.

The Oxford Dictionary (2012a) definition of principle is a "fundamental truth or proposition that serves as the foundation for a system of belief or behavior or for a chain of reasoning". A fundamental truth should be invariant. If the principle is not invariant, and the contextual condition is not explicitly stated in the principle itself, then there are two problems that arise. First, the number of possible "design principles" for a particular discipline can become quite large, as just about any solution to a problem can be formulated into a "design principle", no matter how specific it is to a particular system, context and perturbation. Second, practitioners may either implement the design principle incorrectly (i.e. under the wrong context) or wonder what additional caveats exist in order for the design principle to hold, thereby reducing confidence in the principle and limiting the chances of it being implemented.

5.1.2 Design Principle Invariance

The invariance of some of the current survivability design principles is questionable. If a design principle is invariant, it should be always be followed. If system architects need to be concerned with whether the design principle will help or not, then the principle is merely advice and not a rule. Many of the design principles suggested in the literature are very context-dependent. The principle of mobility, for example, states that viability can be increased by relocating the system to avoid detection by external change agents. The effectiveness of this strategy requires several contextual parameters to be true, including the fact that there is an external change agent who can and does wish to disturb the system, and the fact that the detection of the system by such an agent is location based. Furthermore, some of the design principles, if implemented, may actually reduce viability by making certain perturbations more likely to occur, or by reducing the system's ability to survive them. For example, the principle of concealment states that architects can enable viability of their system by reducing its visibility from external change agents. While following this principle may enable viability when considering perturbations that involve hostile external agents that are looking for the system, it may also hinder its viability in other ways. For example, painting a commercial aircraft black and having no lights would follow the concealment principle, but it would likely make the aircraft more susceptible to collisions with other friendly aircraft.

5.2 Transforming Survivability Design Principles into Design Patterns and Heuristics

All of the design principles for survivability mentioned in the literature are strategies for enabling system viability, but they are different in their approach and applicability. Broadly speaking, there are two types of strategies suggested in the literature for enabling viability in systems. The first type of strategy is a direct solution to specific types of perturbations. For example, the strategy of mobility suggests that a solution to location-based disturbances is to move the system away and avoid them altogether. The second type of viability strategy is a general approach that indirectly improves viability in certain ways. For example, the strategy of independent review that states that outside experts who did not design the system and are not stakeholders should review the design of the system to determine its viability. In an attempt to reconcile the literature, this research will propose that most of the survivability strategies proposed by Richards, Jackson and others as "design principles", can be re-classified as *design patterns* and *design heuristics*.

5.3 Design Patterns for Viability

Design patterns are general solutions to common problems that architects and engineers have used to improve their designs (Beck et al., 1996). Researchers have developed numerous design patterns in areas such as architecture (Alexander, Ishikawa, & Silverstein, 1977), software engineering (Wolfgang, 1994), security (Schumacher, Fernandez-Buglioni, Hybertson, Buschmann, & Sommerlad, 2006), and systems engineering (Cloutier & Verma, 2007) In the seminal book *A Pattern Language*, Christopher Alexander (1977) defines a design pattern as follows:

> Each pattern describes a problem, which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use the same solution a million times over, without ever doing it the same way twice.

Design patterns are specified in such a way that they can be used to solve a particular design problem. At a minimum, the users of a design pattern require a description of the problem that is solved by the pattern, a description of the solution, and a list of related design patterns. In addition to this core, researchers have often included additional useful information in the description of design patterns. For example, the software design patterns described in the highly-cited book *Design Patterns for Object-Oriented Software Development* (Wolfgang, 1994), include sample code so that software developers can more easily adapt the patterns to their own particular design.

Perhaps the most useful aspect of a list of design patterns is how they all relate to each other. Jackson developed a hierarchy of strategies where some strategies are more specific versions of larger, more general approaches. For example, while Richards lists both margin and hardness as separate viability strategies, Jackson lists them both as sub-strategies of a larger, overall absorption strategy. By recognizing that there may be more than one way to solve a particular problem, developing a mapping of viability solutions empowers designers with alternatives to consider. Furthermore, certain strategies enable, or depend on other strategies. Jackson refers to this as "interdependency" and lists numerous dependencies between his strategies. An example of an interdependency is the Human in Control strategy, which states that humans should have final decision making authority in critical situations. However, since humans can also make mistakes, the effectiveness of this strategy is also dependent on the successful implementation of the Avoid Human Error principle. Not all dependencies are as simple as that, however. For example, the Inter-Node Interaction Principle Interdependency involves no less than six different strategies (Figure 5-1).

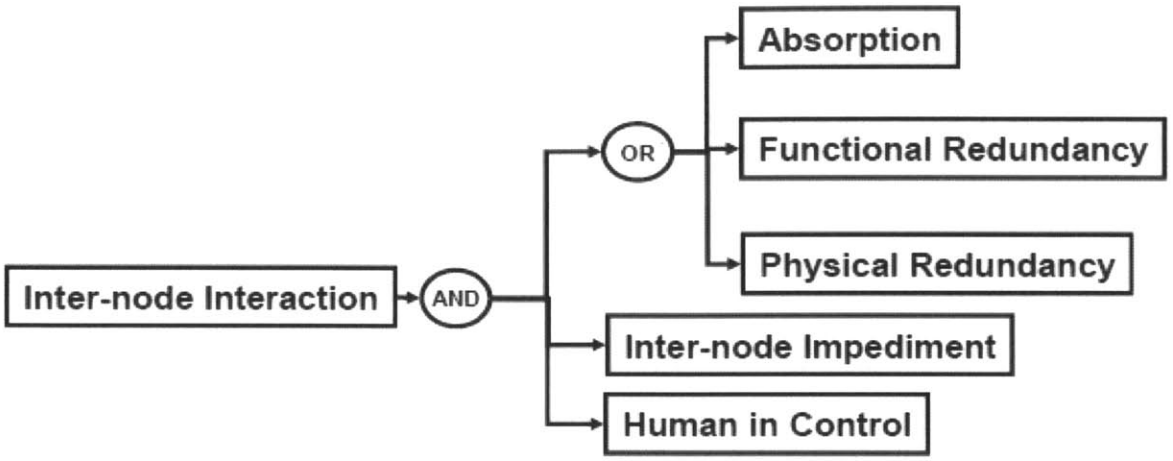

FIGURE 5-1: INTER-NODE INTERACTION DEPENDENCY DIAGRAM (JACKSON, 2012)

More importantly, and of specific interest to viability design patterns, is that certain solutions may conflict with each other. By following the link of related solutions, an architect may become aware that implementing a particular solution may actually decrease viability to different types of perturbations. In this sense, the mapping of design patterns allows designers to be selective amongst the alternatives and avoid unintended consequences of certain design implementations.

5.3.1 Structural and Operational Patterns

Design patterns for viability can be broken down into structural patterns and operational patterns. The implementation of structural design patterns result in specific structural properties, such as twin engines on an aircraft. Most passive viability strategies can be implemented solely through a structural design pattern. For example, the hardness viability strategy can be implemented by including armor, a structural feature that does not require an associated operational strategy.

The implementation of operational patterns involves the execution of a particular strategy for avoiding, mitigating or recovering from a perturbation. Mobility is an operational strategy because it involves relocating the system to avoid a disturbance. Operational patterns execute the active viability strategies, but they typically require structural features to be present. For example, for a system to relocate itself away from a disturbance (i.e. execute the strategy of mobility), it would first have to have suitable features to do so (e.g. wings, wheels, an engine, navigation system, etc.) However, not all operational patterns will be executed by the system and thus may be implemented without associated structural features. The strategy of

deterrence, for example, is often executed by entities outside the system and not the system itself. Laws, enforced by police or the military, deter many hostile actions that could be taken against systems. Many systems rely on the deterrence provided by outside entities as part of their own viability, without explicitly including structural or operational capabilities to enable it.

5.3.2 Elements of a Design Pattern

There is no single template for a design pattern that crosses all disciplines, since problems and their solutions tend to have unique characteristics. For example, in software engineering, design patterns typically include source code where the pattern in actually implemented. A software engineer can actually implement the solution by simply cutting-and-pasting the source code from the design pattern documentation and making a few minor adjustments to link the code with the rest of the software system. This is not possible with an architectural design pattern like "the door", for example, and those described by Alexander (1977). However, there are several key elements that should be included in a design pattern documentation for systems, which make the pattern useful for system architects. They are:

Name. The name of the pattern. The name should be unique and should clearly and concisely identify the solution.

Also Known As. Other names for this pattern. This typically occurs when more than one researcher develop the same solution to a specific type of problem.

Problem. The problem that the pattern is attempting to solve.

Solution. A description of how the pattern solves the problem.

Context. Contextual factors that are relevant.

Unintended Consequences. A description of potential side-effects and tradeoffs that may need to be considered.

Examples. Real-world examples of the pattern solving a particular problem.

Related Patterns. Before implementing a particular pattern, system architects should consult these patterns as well to ensure that a better solution does not already exist, or that the particular pattern does not interfere with other solutions already implemented or planned.

5.3.3 Example of a Design Pattern

An example design pattern for margin is shown in Table 5-2. Other design patterns for viability can be found in Appendix B.

TABLE 5-2: EXAMPLE DESIGN PATTERN

Name	Margin
Type	Operational
Related Design Principle	Margin (Jackson, 2012; Richards, 2009)
Problem	Capability is not being provided adequately.
Context	Perturbation causes the output of system to be inadequate.
Solution	Have system be capable of producing more output than necessary.
Unintended Consequences	May add require additional components, meaning unnecessary size (reducing the strategy of miniaturization), and/or unnecessary weight (making the strategy to mobility difficult to achieve).
Example	Extra-long wings on an A-10 aircraft, which generates enough lift in the event that part of the wings are destroyed or damaged.
Related Patterns	Heterogeneity, Physical Redundancy

5.4 Design Heuristics

Rechtin (1992) refers to heuristics as guidelines, or non-rigorous "rules of thumb". Some of the viability strategies are not specific solutions to perturbations, but general approaches to design that may result in systems that are more likely to be viable. Design Heuristics enable viability through indirect means. For example, the strategy of Failure Mode Reduction states that the designers should eliminate system hazards through intrinsic design. These design strategies do lead to better, more viable designs in general since certain hazards may be eliminated that would have jeopardized the systems viability. However, specific design features or operational strategies that address specific perturbations (e.g. operator error) are not clear. An example of a design heuristic is shown in Table 5-3.

TABLE 5-3: EXAMPLE OF A DESIGN HEURISTIC

Complexity Avoidance (Madni & Jackson, 2009)	**Description:** The system should not be more complex than necessary. **Example:** The Rule of Simplicity used in the Unix Operating System (Raymond, 2004).

5.4.1 Differences between Design Patterns and Heuristics

System architects should use both design heuristics and design patterns to increase the viability of their systems, as they are effective in different ways. For example, to solve specific problems, system architects should first look to see if a design pattern addresses that perturbation. However, to address unknown unknowns, the system architect may wish to follow design heuristics which may strengthen the design overall. The differences between design patterns and design heuristics are summarized in Table 5-4.

TABLE 5-4: DIFFERENCES BETWEEN DESIGN PATTERNS AND DESIGN HEURISTICS FOR VIABILITY

	Design Pattern	Design Heuristic
Strategy tends to	solve a specific perturbation	be general
	generate new features, strategies	help select amongst alternatives
	rigorous, verifiable	lax, unverifiable
	apply only to the design itself (and supporting enterprise)	also apply to the design process
	directly enable viability	indirectly enable viability

5.5 Summary

This chapter examined existing survivability design principles that have been identified in the literature and highlighted the fact that the principles do not exhibit invariance, and the lists of principles compiled by different researchers, are substantially different. In this chapter, the existing design principles are classified into heuristics and design patterns, in a prescriptive manner that makes them more useful for practitioners to actually implement. In the next chapter, these strategies will be tested to see if they are applicable to systems of systems as well.

6 Applying Viability Strategies to Systems of Systems

In the previous chapter, viability strategies for systems were identified in the literature and divided into design patterns and heuristics. However, it is not clear that system engineering techniques are also applicable to systems of systems as well, due to distinguishing characteristics that separate the two. The goal of this chapter is to determine if the existing strategies for viability can be applied at the SoS-level and how the viability of constituent systems relates to the viability of the systems of systems to which they belong. At the end of the chapter, a new strategy for enabling viability is derived from a historical case study of the 2003 North American Blackout is presented that demonstrates that there are other strategies for viability that are not included in the current literature, and a new

6.1 Stakeholder Needs

Value is a subjective measurement of how well the attributes of the system meet the needs and expectations of the stakeholders. Let Γ_A be the stakeholders of system A. If $V_{A|\Gamma_A}$ represents the value that A delivers to Γ_A, α_A is the attributes of A, φ_A is the system context of A, then

$$V_{A|\Gamma_A} = g(\alpha_A, \Gamma_A) \tag{7}$$

However, since $\alpha_A = f(A, \varphi_A)$ (Section 3.2.3), then

$$V_{A|\Gamma_A} = g(f(A, \varphi_A), \Gamma_A) \tag{8}$$

This means that value is a function of the system itself, the context relative to the system, and the stakeholders of that system. For a traditional system, this may seem trivial, but for a system of system, the relationship between the system's attributes, context, and stakeholder needs is much more complex. However, it will be necessary to explore these issues in detail at the SoS level, in order for viability of systems of systems, and system within systems of systems, to be defined.

6.2 Viability in a System of Systems

When examining the impact of applying vitality strategies to systems of systems, it is important to consider the following question. Can a SoS still be viable even if none of the systems within a SoS are viable with respect to a particular set of contexts under consideration? To answer this question, consider the fact that an engineered system is made up of components, which need

to do something useful *for the system*, in order for the system to do something useful for its stakeholders. For example, a hard drive may need to read and write data, a UAV may need to detect targets in the water, and a Facebook user may need to log in every now and then and interact with other users. If none of the components do something useful for the system, then their outputs cannot be combined into something useful, and the overall system will not be able to deliver adequate value to the stakeholders and will therefore be unviable. Therefore, we can state the following:

Axiom of Component Viability: In order for a system to be viable, a necessary and sufficient set of its components must be viable with respect to the system and contexts under consideration.

6.2.1 System Viability within a SoS

The key phrase in the Axiom of Component Viability is that the component must be viable "with respect to the system", not necessarily to itself. This is critical in a system of systems where a constituent system may be viable to its own stakeholders and context, but not to the SoS. To illustrate this concept, consider a MarSec that includes an UAV (system U), a ground control station (system G), and a satellite that links the two (system S) (Figure 6-1). The UAV and GCS are owned and operated by one set of stakeholders (Γ_M), who lease the services of the satellite from a satellite communications provider (Γ_S). The satellite is part of the SoS that includes the UAV and the GCS (SoS M), but the satellite stakeholders also lease bandwidth to cell phone companies (Γ_C). Thus, the satellite is also part of cellular phone SoS as well (SoS C).

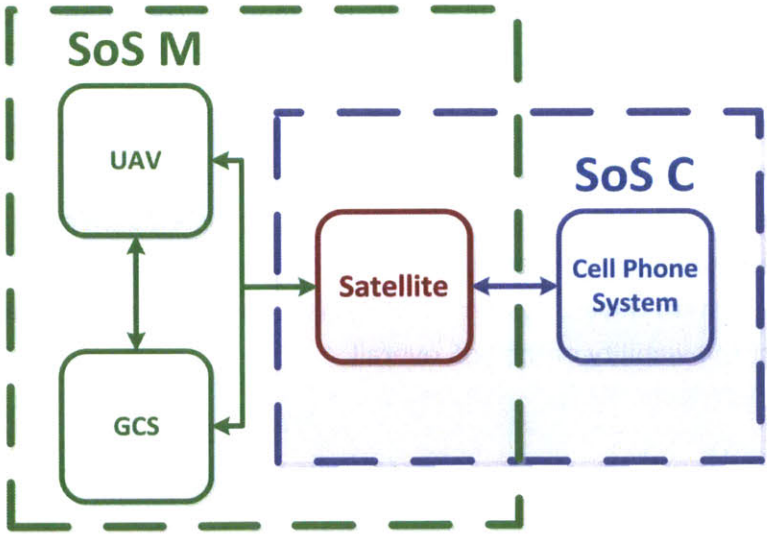

FIGURE 6-1: EXAMPLE OF A SYSTEM BELONGING TO MULTIPLE SYSTEMS OF SYSTEMS

Let $V_{S|\Gamma_S}$ be the value that the satellite delivers to its stakeholders (Γ_S). This value is different, but related to the value of the satellite to the cell phone operators ($V_{S|\Gamma_C}$) and the value of the satellite to the MarSec stakeholders ($V_{S|\Gamma_M}$). The viability of the satellite to its stakeholders ($\psi_{S|\Gamma_S}$) is simply determined by whether or not $V_{S|\Gamma_S}$ exceeds some minimum threshold set by its stakeholders ($T_{S|\Gamma_S}$). $V_{S|\Gamma_S}$ will be some combination of the value the satellite bring to its stakeholders by participating in the MarSec ($V_{M|\Gamma_S}$) and the cellular SoS ($V_{C|\Gamma_S}$).

6.2.2 Example of Viable System, Non-viable SoS

Suppose there is a component failure that makes the satellite communications noisy. Unfortunately, because the satellite is already deployed, a repair is impractical. Since the MarSec only uses the satellite for low bandwidth C2 communications, the noisy channel is not a problem and the MarSec is still viable. However, if the customers of the cell phone company are not willing to tolerate the noisy channel, then the cell phone SoS is no longer viable, meaning that it will likely be shut down and not produce value to the satellite providers either (i.e. $V_{C|\Gamma_S} \to 0$). However, if the value to the satellite stakeholders generated by participating in the MarSec exceeds the minimum threshold to the satellite stakeholders, i.e. if $V_{M|\Gamma_S} \geq T_{S|\Gamma_S}$, then the satellite is still viable overall

6.2.3 Example of Non-Viable System, Viable SoS

Suppose instead that the noisy communications caused by the component failure in the satellite was not acceptable to the MarSec. In this case, the satellite would no longer produce acceptable value in either SoS and would therefore be unviable as a result. However, if the MarSec is able to use a different satellite provider (i.e. replace the failed "component" with another), then the MarSec can remain viable to the random component failure, even if the satellite itself did not remain viable.

System viability strategies applied at the system level will help enable viability at the SoS level, however viable constituent systems do not guarantee that the systems of systems they belong to will also be viable are a result. Similarly, the systems of systems themselves may be viable, even if (certain) constituent systems are not. Therefore, before viability strategies are applied at the constituent system level, system architects should carefully examine how these strategies will impact the viability of the SoS overall.

6.3 Applying Viability Strategies at the System Level vs. SoS Level

Certain system viability strategies can be applied to systems of systems either at the system-level (i.e. bottom-up) or at the SoS-level (i.e. top-down). To illustrate the difference, consider a MarSec where UAVs are performing identification on targets that enter the AOI. The identification process can be approximated as a queue, where the targets that need to be identified arrive according to a random process specified by a mean arrival rate λ. There are k UAVs in the SoS, and each UAV identifies a target according to a random process with a mean service time of μ. As long as $k_\mu \gg \lambda$, then all the targets should get identified before they cross the AOI. Roughly speaking, k_μ represents the capacity of the system to identify targets, and λ represent the demand. Suppose there is a perturbation where a sudden, large influx of targets arrives in the AOI according to some new arrival rate λ'. For this disturbance, $k_\mu < \lambda'$ meaning that all the UAVs will likely be busy for duration of the disturbance and some targets will leave the AOI unidentified as a result. To make the SoS viable in this context, the principle of margin (Richards, et al., 2009) can be applied to either the UAVs or the SoS itself. For example, if the payloads on the UAVs were improved such that the number of targets identified per unit time (i.e. μ) increased to meet the required capacity, then the SoS would be viable in case that perturbation arose. This is an example of applying the principle of margin at the system-level, by increasing the technology level from low to high, and reaping the benefits at the SoS level. To apply the principle of margin at the SoS level, a system architect could simply increase the number of UAVs in the AOI (i.e. increase k) until there was enough capacity to satisfy the demand created by the perturbation.

6.4 Applying the Strategy of Margin to the MarSec

The MarSec DES can be used to demonstrate and assess how the strategy of margin can be applied at both the system and SoS-level. Suppose that the stakeholders agree that in order for a MarSec SoS to be viable, it must able to identify at least 90% of the ships that enter the AOI, on average. A system is proposed, that has 14 vehicle assets (i.e. 8 UAVs and 4 manned patrol boats) and the MarSec DES simulation is run under normal contextual conditions. Since the context and system behavior is highly stochastic, the output of any particular trial is a random variable and therefore multiple trials must be run to estimate the true mean of the probability of ID. Since the simulation runs fairly quickly, a total of 10,000 runs are executed and the results are shown in Figure 6-2.

The mean of the probability of ID was estimated to be 0.941 with a very narrow confidence interval, due to the large sample size. This system is viable, however it is important to note that there are certain times when the system may not be able to meet its viability criteria, due to some of the random effects of both the context and the system itself. This can be seen in the left tail end of the distribution of values shown in Figure 6-2 that fall below the minimum threshold of 0.9. Although this system can be expected to have a probability of ID > 0.9 under normal conditions, there is a chance that the probability may fall below at times. This is part of the stakeholder risk in assessing viability.

Now suppose that the traffic in the AOI is doubled, perhaps due to a temporary disturbance such as an evacuation of a nearby country due to an earthquake, or a conflict that causes major shipping lanes to be re-routed through the AOI. To see if the SoS remains viable, the MarSec DES was run 10,000 times with the same SoS design but twice the ship traffic. As shown in Figure 6-3, the mean probability of ID for the SoS under high traffic conditions was found to be 0.867, which is below stakeholder thresholds. Again, due to the stochastic canture of the inputs and system response, there are certain cases where the SoS meets or exceeds the threshold as shown by the right-tail of the distribution in Figure 6-3, but on average, the system does not met the requirements and would therefore not be considered viable.

To be able to handle disturbances where the traffic through the AOI is doubled, the SoS can implement the principle of margin at either the system or SoS-level to increase the throughput of the system. At the system level, the designers can use better technology (e.g. more advanced payloads) that reduce the amount of time it takes to identify a target by ½. At the SoS-level, designers can simply double the number of vehicles in the SoS. To assess the impact of both of these changes, the MarSec SoS was run again 10,000 times for each strategy under the heavy traffic conditions (see Appendix D). The system-level implementation of the margin strategy (i.e. increasing the technology level of the payloads) exceeded the performance threshold under heavy conditions (probability of ID = 0.960) and even surpassed the performance of the original system under regular traffic conditions. However, the probability of ID for the SoS that had double the number of vehicles (i.e. the SoS-level implementation of the margin strategy) only reached 0.885, doubling the number of vehicles) and failed to meet the viability threshold, although it did improve performance over the baseline SoS, slightly.

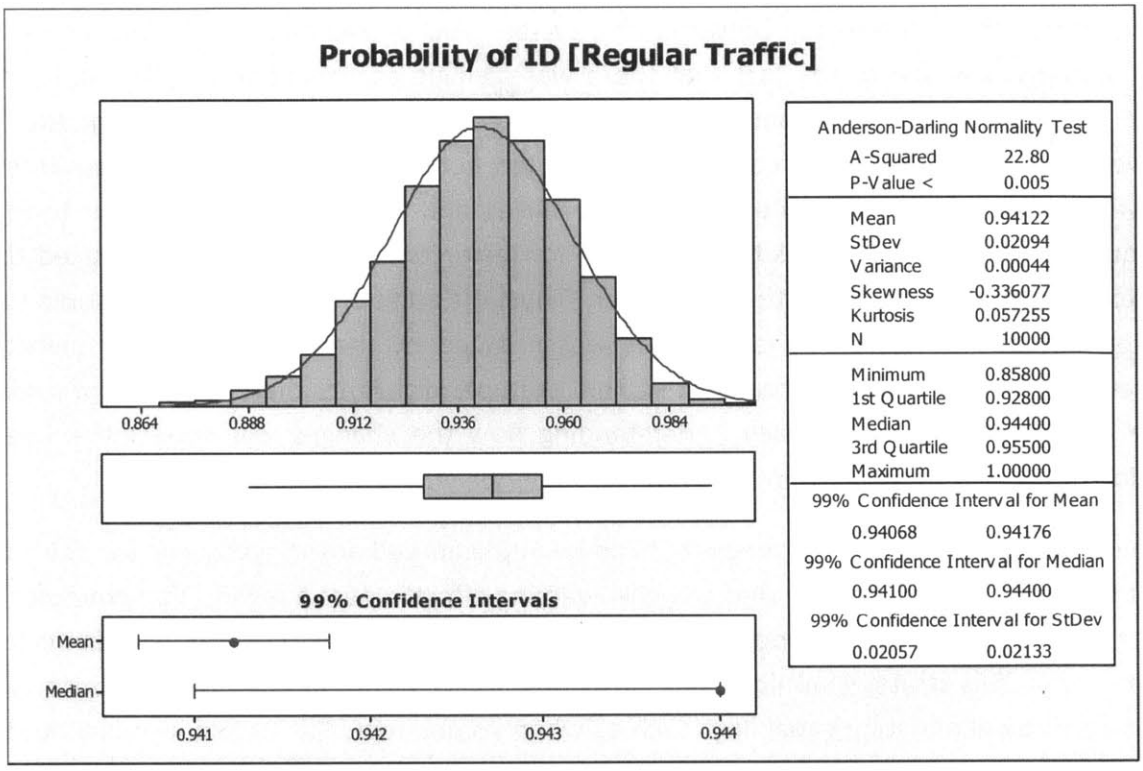

FIGURE 6-2: PROBABILITY OF ID WITH BASELINE SOS EXPERIENCING REGULAR TRAFFIC

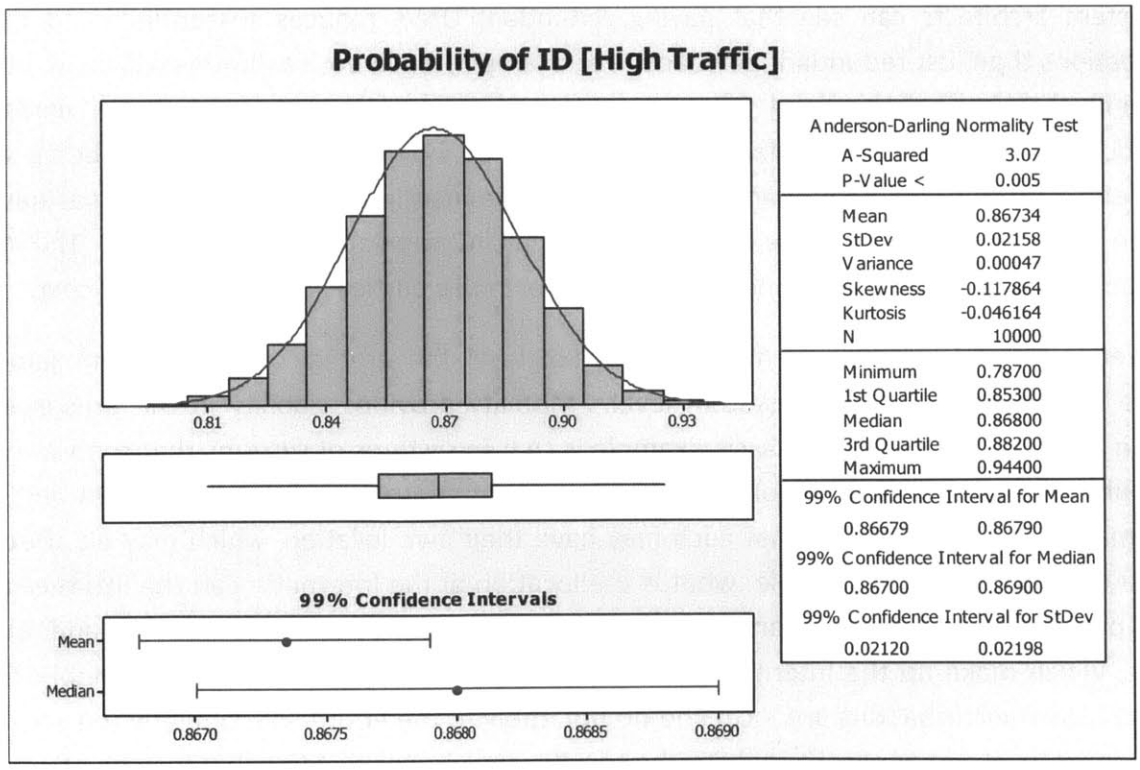

FIGURE 6-3: PROBABILITY OF ID WITH BASELINE SOS EXPERIENCING HIGH TRAFFIC

The reason for the discrepancy between the system and SoS-level implementation of the margin strategy was due to the fact that there was a single command center that delegated tasks to the vehicles, and that command center operated similar to a single server queue. As more vehicles were added, the command center did not have the resources to be able to effectively communicate with the additional vehicles and communication queues formed. These queues caused the vehicles to wait longer for instructions and as a result, reduced their ability to perform their tasks in a timely manner. Thus, the additional capacity that should have been realized by having additional vehicles was negated by the additional communication overhead that did not scale accordingly. This example highlights the difficulties of making change in one area, without really understanding how the changes will impact the overall operation of the rest of the system (or SoS).

Margin is not the only viability strategy that can be implemented at the system or SoS-level and it is not the only viability strategy that can have varying effectiveness a result. For example, the strategy of redundancy can be applied at the SoS level by having multiple UAVs that perform the same task. This strategy would reduce susceptibility to perturbations that reduce capacity of the system by eliminating capability, such as when components fail, or are incapacitated by hostile actions. Redundancy can be applied to the system level, for example, by having multiple payloads onboard the same UAV. By performing a cause-effect mapping, described in Section 7.1, system architects can see that having redundant UAVs reduces susceptibility to more perturbations than just redundant payloads. For example, while both solutions will be viable in the event that a single payload fails, only the redundant UAV solution will be viable in numerous events such as engine failure, an operator not showing up for work, UAV being shot down, etc. Furthermore, redundant UAVs also provide margin, whereas in most cases having multiple cameras of the same type onboard the same UAV will not improve capacity. The main disadvantage to applying redundancy at the SoS level in this particular case, would be cost.

While redundancy may be more effective at the SoS level, the strategy of mobility, for example, may be more applicable at the system level. Mobility provides viability in the presence of location based disturbances. A classic example is that in systems of systems that tend to have geographic distribution of their components, there is often not one SoS location. Rather, the SoS is made up of components that each may have their own location, which may be affected by local disturbances. For example, what is the location of the Internet? Can the Internet as a whole physically move away from disturbances? Routers, personal computers and other devices, which make up the Internet can move perhaps, but is the Internet itself mobile? As location-based perturbations are local and do not apply to the entire SoS, similarly the solution of mobility is local and applicable only to the specific constituent systems that may be affected.

6.5 Sufficiency of Existing Viability Strategies

Research question 2b asks if there are additional strategies that enable viability in systems, particularly systems of systems?. To answer that question, one of the largest SoS failures ever, the Northeast Blackout of 2003, will be examined and scrutinized from the perspective of the existing viability strategies.

6.5.1 North American Power Grid

Many experts believe that the North American power grid is the largest and most complex system of the technological age, with approximately 15,000 nodes (power stations), and 20,000 transmission lines (edges) (Albert, Albert, & Nakarado, 2004) (Figure 6-4). The primary purpose of the power grid is to ensure that power is safely and effectively transferred from generators to end users, in a complex network of interconnected nodes that span a large geographical area and include many different independent power operators. The North American power grid is divided in the Eastern, Western and Texas regions, which are tightly coupled within each region, and loosely coupled between.

FIGURE 6-4: THE POWER GRID LIGHTS UP NORTH AMERICA AS VIEWED FROM SPACE

The most obvious type of failure in such a system is when a node in the chain of transmission between the generator and end user fails and the two end points become disconnected. In accordance with the localized capacity heuristic, the topology of the power grid is such that a single node failure will simply cause power loads to shift through different transmission paths until it eventually reaches its intended destination. In fact, an analysis by Albert et al (2004), showed that networks such as the North American power grid are fairly robust to multiple node failures if they are random in nature, but quite vulnerable if a few high-load nodes fail. Failure of 4% of the high-load nodes in a network similar to the North American power grid, results in 60% connectivity loss. If 8% of the highest-load nodes fail in order, which might be case if the grid was under a directed attack, then complete connectivity loss can occur.

6.5.2 Chain Reactions and Cascading Failures

Disconnections affect the nodes that are no longer connected to the network in obvious ways. However, in networks such as the North American power grid, disconnections also impact nearby nodes that are still connected to the network. Since high amounts of electrical power is not easily stored, the power grid must carefully balance the demand of electricity to its supply. In power grids, failure of an edge transfers the load being carried by the edge to other edges in the network. If those other edges have the additional capacity to carry the load (i.e. margin), then there is not a problem. However, there are limits to how much load an edge can carry and if those limits are approached, then the new edge may fail itself either involuntarily, or by being shut down to prevent damage. When this new edge subsequently fails, then other edges must take both its load as well as the load of the original failing edge as well. This can cause a chain reaction, where failures lead to further failures that propagate throughout the system. Cascading failures are a special type of chain reaction where the small failures eventually lead to major events, e.g. a nuclear power operator spilling coffee on a command console causing a chain reaction that ends in a Chernobyl-level. Preventing chain reactions and in particular, cascading failures, should be a priority for system architects. Figure 6-5 illustrates how chain reaction of small events can grew into the cascading failure that resulted in the US Northeast Blackout of 2003. As stated in Chapter 3, the initial incident was an untrimmed tree in Ohio that severed a power line. This link failure burdened other lines with an extra load, causing some of them to fail as well. Each failing power line placed additional load on the remaining lines until they succumbed as well, leading to a chain reaction of power line failures. Eventually, enough power line failures caused entire power plant generators to shut down as a precaution, in a cascading failure that eventually lead to a complete grid failure covering the Northeastern United States and some parts of Canada. There were multiple incidents and circumstances along the way that contributed to this unfortunate outcome. One such perturbation was the failure to recognize and mitigate the failing power lines in Ohio, due to a

software bug affecting the control rooms of one of the Ohio power plants (Zhivich & Cunningham, 2009).

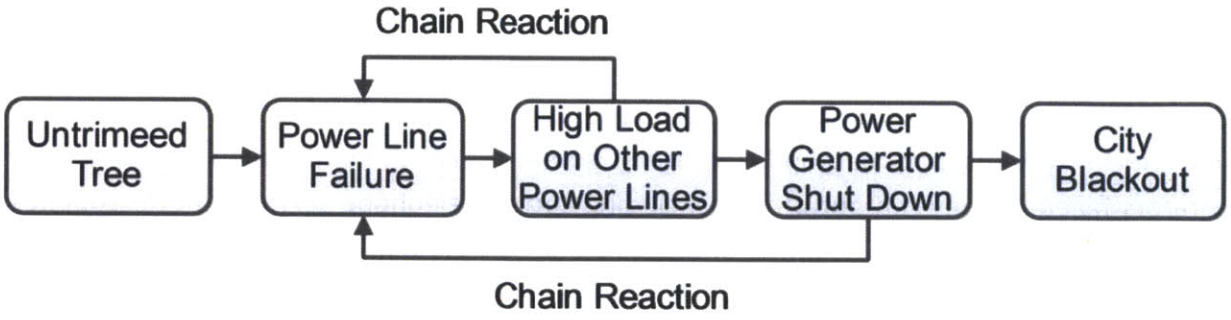

FIGURE 6-5: CASCADING FAILURE LEADING TO NORTHEAST BLACKOUT OF 2003

Cascading failures are system failures that result from component failures that cause other components to fail, which cause further components to fail, and so forth. Cascading failures are a major concern for connected networks like a power grid, but also occur in other systems as well. When the financial crisis of 2008 happened, certain banks were deemed Too Big To Fail (TBTF) and bailed out to prevent cascading failures from crippling the entire economy. In his address to the Independent Community of Bankers on March 20, 2010, the Chairman of the Federal Reserve said that to avoid a collapse of the financial system, the government now only has to bail out banks that are too big to fail, but those that are too interconnected to fail as well (Bernanke, 2010).

6.5.3 Northeast Blackout of 2003

On August 14, 2003 at approximately 4:10pm EDT, the second largest blackout ever affected the large parts of the Northeastern United States and parts of Canada (Figure 6-6). The following is a condensed timeline of the events that occurred on August 13, 2003 (Lerner, 2003; U.S.-Canada Power System Outage Task Force, 2004):

- 13:31 (EST) – FirstEnergy shuts down a generating plant in Eastlake, Ohio.
- 14:02 (EST) – Overgrown trees in Ohio fail when it comes in contact with transmission line
- 14:02 (EST) – Alarm in FirstEnergy fails due to computer bug
- 14:32 (EST) – More power lines in Ohio fail, due to power loads being placed on them. FirstEnergy notices problem, but fails to inform other power operators in nearby states.
- 16:10 (EST) – Power lines in Michigan start to fail. Eastward flow of power cut off. Demand surges and overloads power station causing them to go offline, starting the blackout.
    - Large metropolitan areas start separating from the grid

    - Blackout spreads in a cascading failure.
- Last of the affected power stations separate from the grid and cascading failures stops.

By the end of the day, more than 508 generating units at 264 power plants shut down, after a 3.5 GW power surge affected the transmission grid. The blackout lasted between 7 and 16 hours, and resulted in a 20% increase in fatalities during that period (Al-Aufi, 2012). Although many critical systems had backup generators, some critical systems failed because their generators failed or they ran out of fuel, including certain telephone services, water supplies, gas stations, border crossings, hospitals and television/radio stations

FIGURE 6-6: AREAS AFFECTED BY THE NORTHEAST BLACKOUT OF 2003

6.5.4 Systems Affecting Other Systems within a SoS

One of the largest problems in a SoS is that the environment in which a constituent system interacts often forms part of the context of one or more other constituent systems (Figure 6-7). That is, the outputs of a constituent system, whether voluntary or involuntary, often become the inputs of another system within the SoS. This is the case with the North American power grid, which allowed the cascading failure to emerge. Certain power systems intentionally shutdown to prevent damage to their own system, which caused the excess current to be sent to adjacent systems. This meant that the actions of one system caused a perturbation in the context of another.

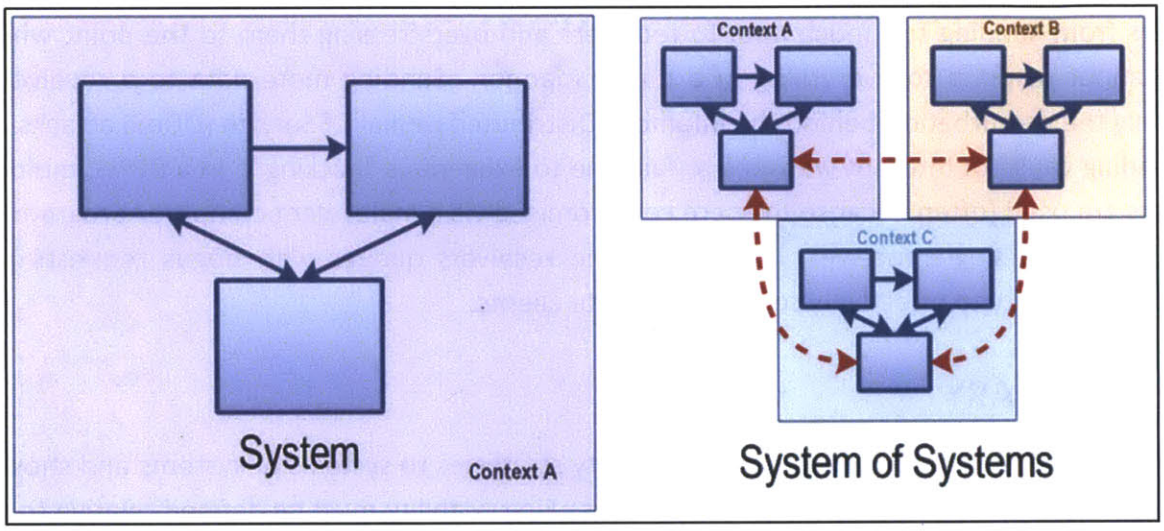

FIGURE 6-7: CONTEXTUAL DIVERSITY WITHIN A SOS

In order for a SoS to be viable, strategies must be implemented where the constituent systems ensure that their actions do not negatively impact the contexts of other constituent systems, if those changes may not be survivable. In other words, they must not disrupt the context of other systems, if those systems are fragile to those disruptions.

6.5.5 Intensity Regulation

In the case of an intensity-based disturbance like that involving current, the intensity regulation of the system's output is an example of a strategy that can help enable viability of systems, but was not discussed by Richards or Jackson. Throttling is when a system intentionally reduces its output, to increase its own survivability to a perturbation it is experiencing, or help other constituent systems avoid future perturbations by not disrupting their context. In the case of power grids, throttling is achieved in part through "shedding load", i.e. eliminating power to some areas ("rolling blackouts") and/or reducing the voltage provided ("brownouts").

There are other examples of intensity throttling used in both traditional systems and systems of systems as well. In most modern laptop computers, the CPU and GPU produce the lion share of the heat. Heat is an unintentional interaction effect that threatens the component within the laptop. If the internal temperature of the laptop rises above a critical threshold, then certain components, such as the RAM or hard drives, may fail as a result. To prevent this, CPUs and GPUs in most modern laptops will down-clock themselves, i.e. operate at a lower speed and produce less heat, for a period of time when the overall internal temperature is too high. On the other end of the SoS spectrum, throttling is specifically implemented as part of the Transfer Control Protocol (TCP) protocols that forms part of the operational backbone of the Internet itself. Specifically, TCP contains rules for end-to-end flow control that prevents individual senders from sending too much data to receivers and overstressing them to the point where they cannot respond to everything in a timely manner. Sending more data to a receiver is precisely the perturbation behind the infamous Distributed Denial Of Service (DDoS) attacks are the leading cause of Internet websites to fail, due to exogenous "hacking". In a DDoS, multiple senders are used (often because they are compromised via a malevolent computer program) to send data to a single receiver, overflowing the receivers queues with bogus requests and reducing its ability to effectively service legitimate clients.

6.6 SUMMARY

This chapter applied some of the existing viability strategies to systems of systems and showed that they are still applicable, with several caveats. First, viability must be defined relative to the SoS, not the system. Additionally, a system may be viable, even if the SoS is not viable, and a SoS may be viable, even if one or more of its constituent systems are not. Thus, viability of constituent systems and viability of systems of systems are not the same. Additionally, viability strategies for systems of systems can be applied at the system level or SoS level, with different results. It is for these reasons that care must be taken by decision makers when applying viability strategies to systems of systems, since they are more complex than traditional systems, and have more unintended consequences as a result. This chapter ended by providing a historical case study that demonstrated the need for an additional viability strategy (intensity regulation) that was not identified in the systems literature. In the next chapter, additional viability strategies using the concept of system context and cause-effect mapping of perturbations, will be developed.

7 ADDITIONAL VIABILITY STRATEGIES

In the previous chapter, viability strategies applicable to systems, that were previously identified in the literature were found to be applicable to systems of systems as well. Additionally, a new viability strategy, intensity regulation, was discovered after examining the Northeast Blackout of 2003 and how the North American power grid handles power perturbations. In this chapter, additional viability strategies for engineered systems not identified in the literature will be presented. This will be accomplished by first examining the MarSec in detail using a cause-effect mapping to determine a set of perturbations that may cause the SoS to lose its ability to provide value. Then, possible points of intervention by system architects and operational decision makers will be highlighted. Finally using various historical examples and examining the perturbations from a nested contextual viewpoint, methods for avoiding and surviving these perturbations are identified and formulated into general viability strategies. A selection of these new strategies was implemented and validated through experiments in the MarSec DES, which is discussed at the end of this chapter.

7.1 CAUSAL DIAGRAMS OF CAUSE AND EFFECTS

To develop a list of perturbations of interest, and to determine possible points of intervention where viability strategies can be applied, a cause-effect mapping is performed where the multiple causes and effects of perturbations are linked from spontaneous events to terminal conditions. Spontaneous events are those outside the system's control. In many cases, spontaneous events are exogenous (i.e. outside the system boundary), such as changes in the weather or the action of an outside entity. Spontaneous events can also be endogenous, such as a random component failure or operator error. Spontaneity is not absolute, but rather it is relative to the system. A chicken crossing the road would likely be a spontaneous event to a car[2], but not to a farm. The chicken may be lured by the smell of a particular crop. Once the chicken crosses the road, it may start to eat the crop on the other side, which is a disturbance to the farm. Forcing the chicken to cross the road to get to a particular crop, may be a viability strategy of the farm if the crop the chickens are eating is relatively worthless compared with the crop on the other side of the road.

[2] The traditional answer to the infamous joke "Why did the chicken cross the road" is "To get to the other side". The punch line is anti-humorous because it highlights the fact that we really do not know the reason, even though the listener imagines all sorts of wild causes.

A cause-effect mapping begins with a terminal event (Figure 7-1). The causes of that event are added to the map as perturbations of interest and a single arrow from each of the causes to terminal event is drawn. If any of the causes of the terminal event are not spontaneous events, then their causes, along with appropriate arrows, are added to the map as additional perturbations of interest. Then each perturbation of interest is examined for additional causes and effects. Arrows are drawn to any causes and effects already on the map and new perturbations are added, as necessary. This process continues until each perturbation originally started with a spontaneous event, and eventually results in a terminal event. If a perturbation cannot eventually lead to a terminal event, then it is not worth considering and should be removed from the cause-effect mapping.

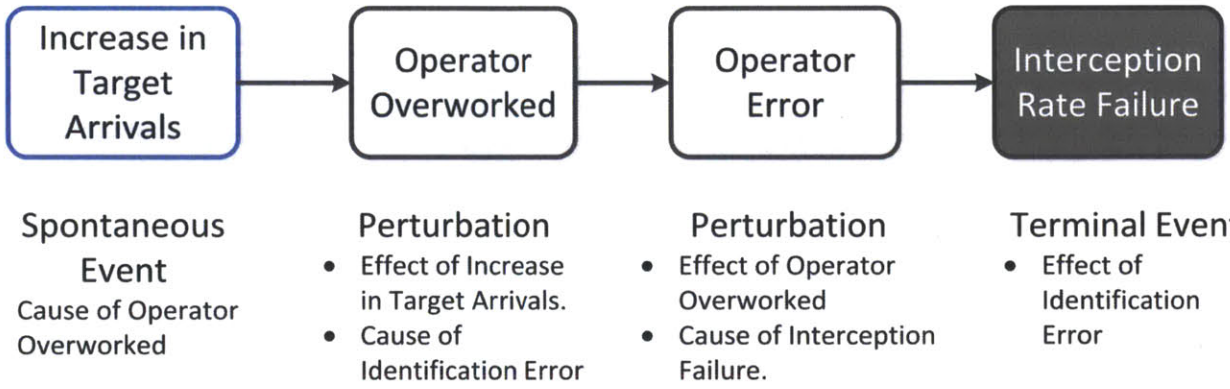

FIGURE 7-1: SIMPLE CAUSE-EFFECT MAPPING

7.2 Strategies as Intervention Points

The cause-effect mapping is useful for being able to highlight areas of intervention, where strategies can be implemented to increase viability by mitigating or recovering from causes by preventing their effects from happening. For example, in Figure 7-2, operators tend to perform more errors when they are under a high workload (Wiener & Nagel, 1988). This perturbation can lead to a terminal event (e.g. interception rate failure), so reducing the probability of operators being overworked is one strategy for increasing the viability of the system overall. This can be achieved, for example, by incorporating automation into the identification process that double checks operator's actions for common mistakes. This type of solution implements the avoid human error strategy as discussed by Jackson (see Section 3.4.3).

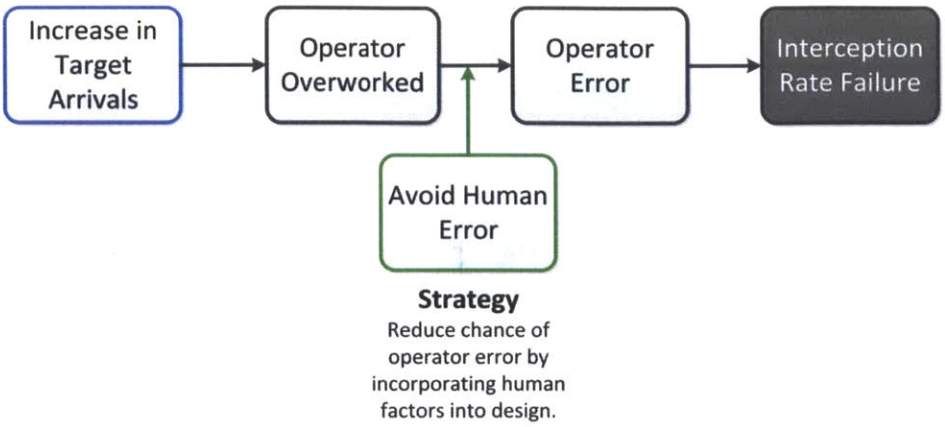

FIGURE 7-2: APPLYING A STRATEGY AS AN INTERVENTION

7.3 Cause-Effect Mapping for MarSec

A cause-effect mapping was performed for the MarSec (Figure 7-3). Not all perturbations are shown in Figure 7-3, rather only a very small subset is shown to highlight where new strategies for enhancing viability might be made. For example, a UAV may not be available due to a random component failure. However, random component failures are not shown in Figure 7-3 because they can be addressed using the existing drift correction strategy. The details of the cause and effect mapping are discussed in Table 7-1 .

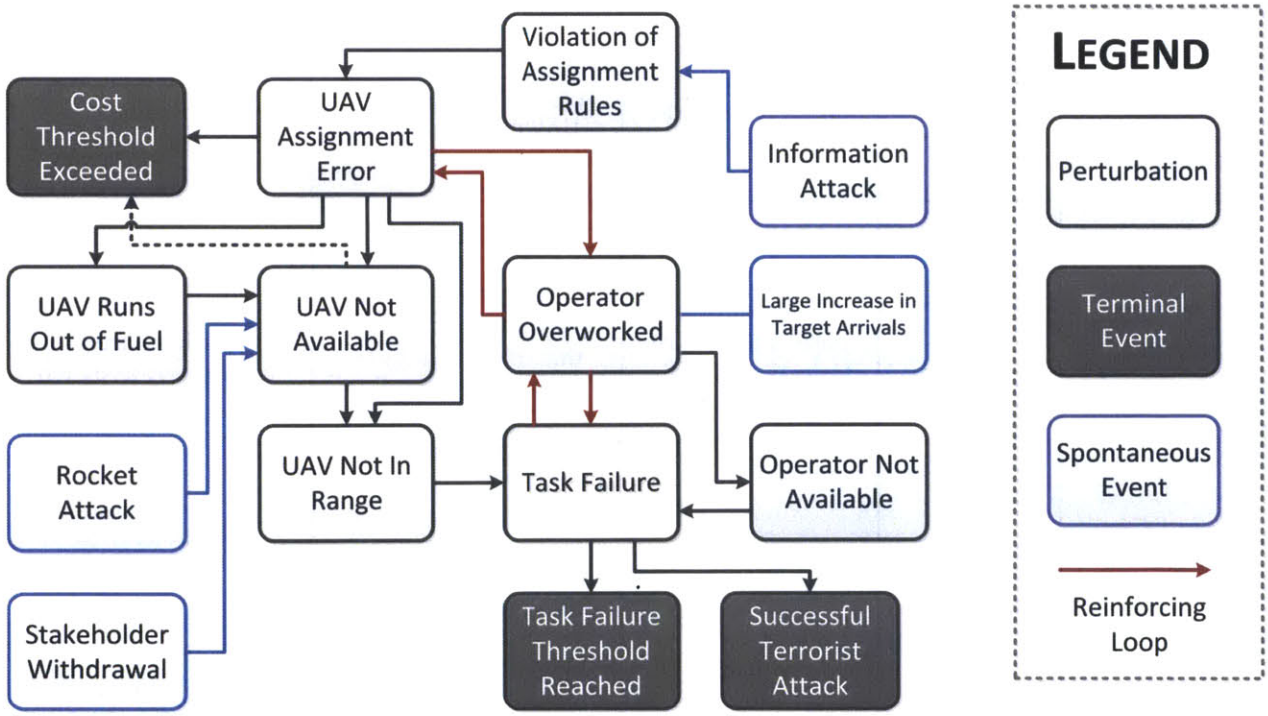

FIGURE 7-3: CAUSE-EFFECT MAPPING FOR MARSEC

TABLE 7-1. DESCRIPTION OF PERTURBATIONS IN CAUSE-EFFECT MAPPING OF MARSEC SOS

EVENT	TYPE	DESCRIPTION	CAUSES	EFFECTS
Task Failure Threshold Reached	Terminal Event	Too many task errors cause performance to drop below value threshold.	Too many task failures.	System no longer viable.
Successful Terrorist Attack	Terminal Event	Terrorist successfully detonates bomb in harbor.	Terrorist wasn't intercepted before it detonated bomb.	System no longer viable.
Cost Threshold Exceeded	Terminal Event	Cost of system exceeds acceptable limit.	Resource price increase, operational costs (e.g. excessive travel burns fuel), capital costs (e.g. UAV repairs after being attacked)	System no longer viable.
Task Failure	---	Required task (e.g. identify target) not performed, or not performed properly.	Operator error, available UAV not in range, operator busy.	Task failure threshold reached, successful terrorist attack.
Operator Not Available	---	Operator not available to perform required task.	Operator busy, operator shortage.	Task failure.
UAV Not In Range	---	UAV not in range of target to perform task.	UAV not available, UAV sent to wrong location, UAV not assigned to task.	Task failure.
Operator Overworked	---	Operator performing more tasks than he/she should for a given period of time.	Too many targets in AOI, previous task failures that make additional work and improperly assigned UAVs that have to be re-assigned.	Task failures, improperly assigned UAVs.
UAV Not Available	---	UAV is not available.	UAV damaged/destroyed by rocket attack, stakeholder controlling UAV withdraws support, UAV runs out of fuel.	Cost threshold exceeded if damaged/destroyed UAV has to be repaired/replaced, UAV not in range to perform task.

EVENT	TYPE	DESCRIPTION	CAUSES	EFFECTS
UAV Runs Out of Fuel	---	UAV runs out of fuel	Unnecessary travel due to UAVs being sent to the wrong locations.	UAV not available.
UAV Assignment Error	---	UAV sent to wrong location, UAV not tasked at all.	UAV assignment rules violated intentionally or unintentionally (e.g. to operator overworked).	Cost threshold exceeded if unnecessary travel consumes too much fuel, UAVs not being in range or available.
Violation of Assignment Rules	---	Assignment rules (such as targets go to closets UAV) are not followed.	Operators involuntarily assign UAVs when they should not, information attack.	UAV assignment error.
Information Attack	Spontaneous Event	Hostile entity takes control of UAVs.	---	Violation of assignment rules.
Large Increase in Target Arrivals	Spontaneous Event	Too many targets arrive in AOI over a specific period.	---	Operator overworked.
Stakeholder Withdrawal	Spontaneous Event	Stakeholder withdraws it support from the SoS	---	UAV not available.
Rocket Attack	Spontaneous Event	Hostile entity attacks UAV with rocket.	---	UAV not available.

The causal diagram highlights several interesting dynamics associated with how perturbations in this particular system propagate. System architects are constrained in how they can intervene to solve some of the problems caused by the perturbations in Figure 7-4. There is a limited amount of time, money and other resources to implement solutions to every perturbation. There are also design constraints, such as size and weight, as well as boundary constraints that limit the influence system architects have. For these reasons, system architects must be selective in the strategies they chose to implement. A cause-effect mapping enables system architects to visually identify possible points of intervention where the system or supporting enterprise can do something to avoid, mitigate and recover from certain types of

perturbations. In particular, system architects should try to intervene to avoid perturbations that have multiple effects, known as common causes, and survive perturbations that have multiple causes, known as common effects. It is particularly important to intervene against perturbations that are part of a reinforcing loop, since they are both the sources of multiple effects and multiple causes simultaneously, and lead to the often devastating cascading failures seen in large-scale systems of systems. Cause-effect mappings also help system architects deal with "unknown unknowns", or perturbations that no one thought of. Sometimes the solution to a known problem is also the solution to unknown problem. Common causes and common effects are probably more likely to be causes and effects of unknown perturbations as well.

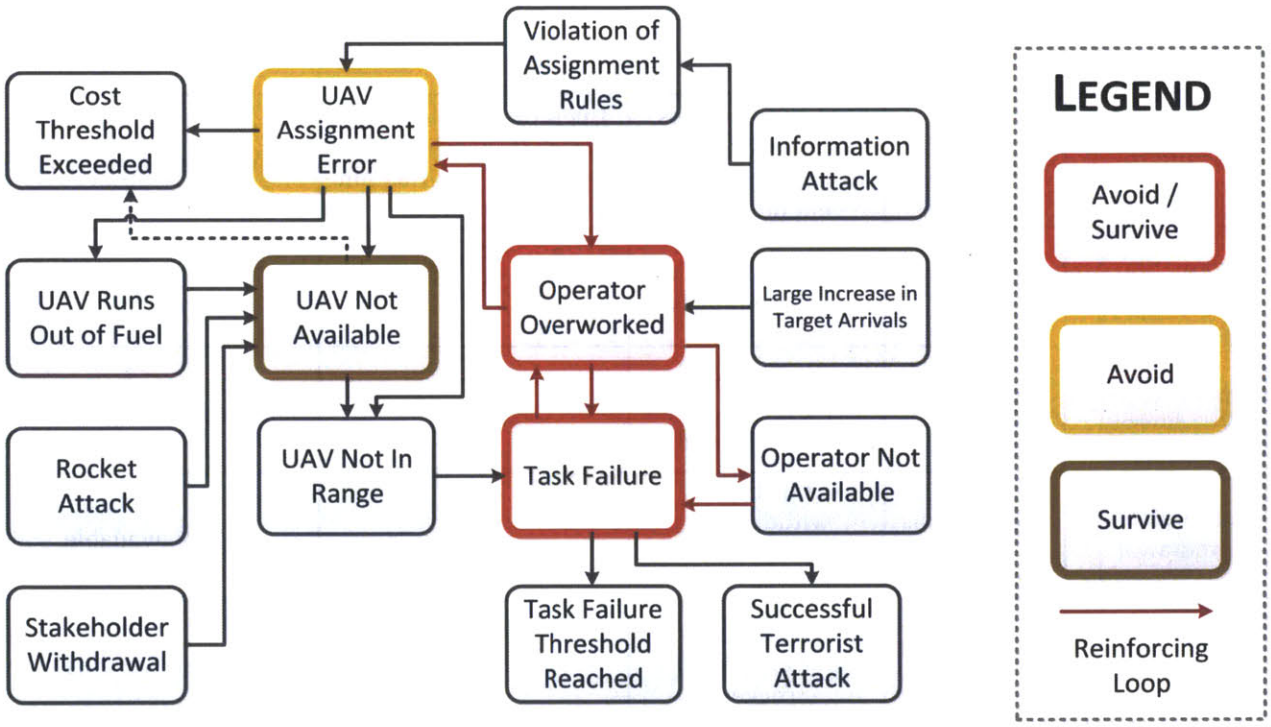

FIGURE 7-4: GOOD CANDIDATES FOR POSSIBLE POINTS OF INTERVENTION

7.4 Strategies for Intervention in the MarSec example

By using the cause-effect mapping to generate perturbations of interest (Figure 7-5), recognizing the differences between system and global contexts, and looking at some related historical cases, some interesting strategies for enabling viability emerge. These new viability strategies are discussed in this section.

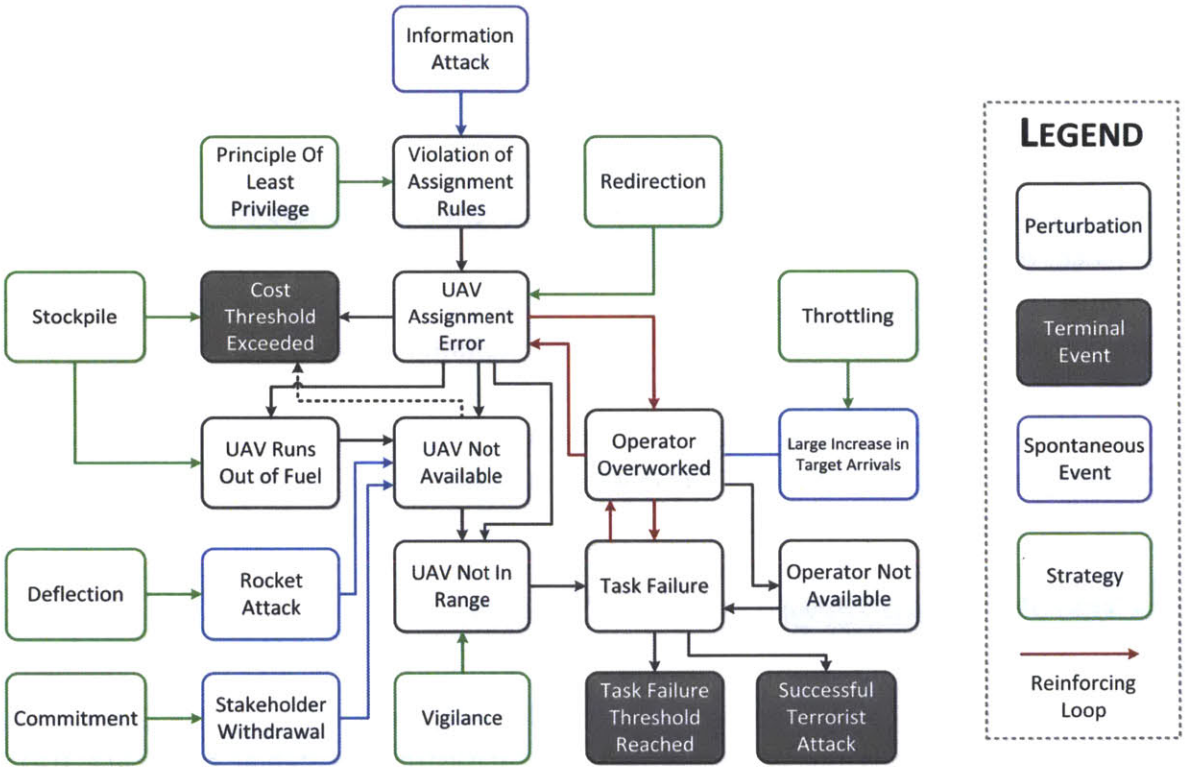

FIGURE 7-5: CAUSE-EFFECT MAPPING FOR MARSEC WITH VIABILITY STRATEGIES APPLIED

7.4.1 Commitment

One of the key characteristics of systems of systems is the fact that they are never fully formed and have what is referred to as evolutionary development. This means that stakeholders and consistent systems often come and go and their participation is not always guaranteed. Furthermore, if their values are not in line with those of the overall SoS, they may simply change the way their constituent systems operate in a way that disrupts the concept of operations for other systems in the SoS. For example, the satellite provider may unilaterally implement security features that reduce the effective bandwidth and unintentionally delay certain communications between the UAVs and ground control station.

Suppose system *A* belongs to a system of systems (SoS *X*) that includes system *B*. System *B* provides a critical functionality to system *A*. In general, if the critical functionality to system *A*

is being provided by system *B* that is controlled by its own stakeholders (i.e. Γ_B), then the viability of system *A* does not depend on the viability of system *B*. It is not necessary that system *B* remain viable. System *B*'s stakeholders can replace system *B* with something else that can provide the critical functionality. However, to buffer the system context of *A*, stakeholders of system B should be committed to providing the critical functionality to system *A*.

A common way of ensuring commitment to a SoS, and thereby protecting the local contexts of constituent systems from internal, stakeholder withdrawals, is to require Service Level Agreements (SLA) between the constituent systems (Bianco, Lewis, & Merson, 2008). A SLA is a contract between the stakeholders, outlining the required services each stakeholder is responsible for, and penalties for not providing such services when needed. By setting appropriate penalties, system architects can either prevent stakeholders from withdrawing in the first place, or use the penalties to help mitigate or recover from the withdrawal. The SLA protects both the content provider and the content consumer. A constituent system in a SoS wants to ensure that its context is protected from voluntary stakeholder withdrawals and service disruptions. Similarly, it may want to discontinue providing services to other constituent systems, if that service is causing other unintended interactions that are disrupting the system context. The case of Wikileaks and the Amazon EC2 cloud highlights the importance of the SLA in protecting the system in scenarios similar to the latter.

7.4.1.1 Historical Case: Amazon & the Wikileaks

On November 28, 2010, the website Wikileaks began to publish the first set of confidential diplomatic cables from US embassies on its site, causing a diplomatic crisis in the United States and front-page news across the world. After a set of Distributed Denial of Service (DDoS) attacks on its Swedish servers, Wikipedia decided to move its servers to Amazon as part of their EC2 cloud solutions. Almost immediately, WikiLeaks was subject to further DDoS attacks, which Amazon proudly claimed to have thwarted. However, a few days after being contacted by Senator Joseph Lieberman asking why the American-based company would host services for a website that was illegally distributing stolen and confidential materials, Amazon voluntarily terminated their contract with Wikileaks and stopped hosting their website (Benkler, 2011). There were many speculations as to why Amazon had to do this, but Amazon claimed it was because Wikileaks violated the SLA (Babcock, 2010). As part of the SLA between Amazon and its customers, the customers agree not to host content that they do not legally own, nor post content that could potentially harm someone. While these terms may seem quite vague, Amazon had the legal right to terminate the arrangement and protect themselves from further hacker attacks and unwanted Federal investigations.

7.4.2 STOCKPILING

To perform an action, systems typically consume resources. At a minimum, some amount of energy is always expended to perform work (at least from a physics perspective), so energy is typical consumable. Energy is usually provided to systems in the form of direct AC/DC current, batteries or fuel. Other common consumables include raw materials like iron and oxygen, as well as money to pay salaries and purchase whatever is needed, when it is needed.

Unless the system is self-sufficient, consumables need to be acquired from external sources and brought into the system context. However, due to a variety of perturbations, the system may not be able to get the consumables as easily as expected. Fuel prices may rise dramatically, external funding may be cut, or power may be unavailable, for example.

The ability to survive, despite disruptions in external sources of consumables, is often a number one priority for many systems. In fact, for many living organisms, survival is almost exclusively dependent on the ability to get energy from food sources to meet its daily demands. Similarly, system architects should ensure that reserves of critical consumables are kept and made available to mitigate perturbations in the global conditions, from affecting the system context. This is known as stockpiling and has been used in many systems, in a variety of ways.

7.4.2.1 HISTORICAL EXAMPLE: THE UNITED STATES STRATEGIC PETROLEUM RESERVE (SPR)

Oil is a vital resource for any industrialized nation and it is one of the most traded and volatile commodities on the free market. The extraction, production, sale and transport of oil are truly global endeavors that are subject to wide variety of contextual perturbations. Changes in weather, environmental laws, politics, and economies all have had major impacts somewhere along the supply chain.

Not surprisingly, the United States is the largest consumer of oil in the world, with an estimated consumption of 18,690,000 bbl/day (Central Intelligence Agency, 2012b). Perhaps surprising to some, the United States also produces 9,056,000 bbl/day, which makes it the third largest producer globally. Clearly, like China, Japan, the UK, Germany and many other powerful nations, the United States does not produce as much oil as it consumes, and must import oil from other parts of the world. A quick glance at the top 10 oil exporters in the world (Table 7-2), shows that the most of the US's oil imports either come from the volatile Persian Gulf region (Saudi Arabia, UAE, Iran, Kuwait, Iraq) or from countries that not overly friendly to US interests (Russia, Venezuela). For this reason, it is not surprising to think that something could happen where these countries do not supply the level of oil that is needed for the US to

continue operating at a military or economic level that the government expects. Thus, one can say that the US is not viable to its current expectations, without an adequate supply of oil.

TABLE 7-2: LIST OF TOP OIL EXPORTERS (CENTRAL INTELLIGENCE AGENCY, 2012A)

RANK	NATION	OIL EXPORTS	DATE OF DATA REPORTED
1	Saudi Arabia	8,900,000 bbl/day	2007
2	Russia	5,430,000 bbl/day	2009
3	United Arab Emirates	2,700,000 bbl/day	2007
4	Iran	2,400,000 bbl/day	2010
5	Kuwait	2,349,000 bbl/day	2007
6	Nigeria	2,327,000 bbl/day	2007
7	Venezuela	2,182,000 bbl/day	2007
8	Norway	2,150,000 bbl/day	2009
9	Canada	2,001,000 bbl/day	2008
10	Iraq	1,910,000 bbl/day	2009

A catastrophe caused by a shortage of oil may be avoided in several ways. One way is to increase internal oil production. The problem with this approach is that there is only a finite amount of oil in the United States as well as an environmental impact that limits the oil production. Other strategies, advocated by President Jimmy Carter in response to the 1979 energy crisis, are for Americans to reduce their energy consumption and for the United States to discover alternate means of energy production (Carter, 1979). However, in the more than 30 years since that crisis, there have not been any major engineering breakthroughs that have significantly reduced the United States dependence on oil and it is clear that there will be a significant gap in oil consumption vs. production for the foreseeable future. Therefore, the United States, for the time being, is dependent on oil imports and is subject then to external perturbations as a result.

Although there are many causes (war, political pressure, environmental laws, oil spills, natural disasters, etc.) there are two terminal events for the U.S. The first is that the cost of oil rises beyond the critical threshold set by the stakeholders. At this point, Americans can no longer afford the oil and either must take additional measures to continue the status quo (such as borrowing money) or reduce their consumption. Changes in oil prices are common and a critical rise in the price of oil is probably the most common threat. The second terminal event is the supply of oil just does not meet the demand. If for some reason, there just are not enough

barrels of oil available to satisfy the demand of the U.S., then the U.S. has a problem that money cannot solve. The threat of there not being enough oil at all to meet demand is a much more serious, but less likely event, at least for the near future. However, the amount of oil available from fossil fuels is limited and this problem may in fact end up being very real in a generation or two (Figure 7-6).

To reduce the impact of external perturbations on the supply of oil to the United States, the American government has adopted two major strategies. First, the Carter Doctrine follows the strategy of preemption by openly declaring that the US will use military force to protect its national interests in the Persian Gulf region. This threat was a direct response to the energy crisis caused by Iran hostage situation in 1979. The second strategy is the Strategic Petroleum Reserve (SPR). According to the U.S. Department of Energy (www.fossil.energy.gov), the SPR is the largest oil reserve in the world, capable of storing 727 million barrels of oil. Currently, the SPR holds around 694 million barrels of oil, or about 95% of its capacity. At current US import rates, this translates into a supply that would last 80 days.

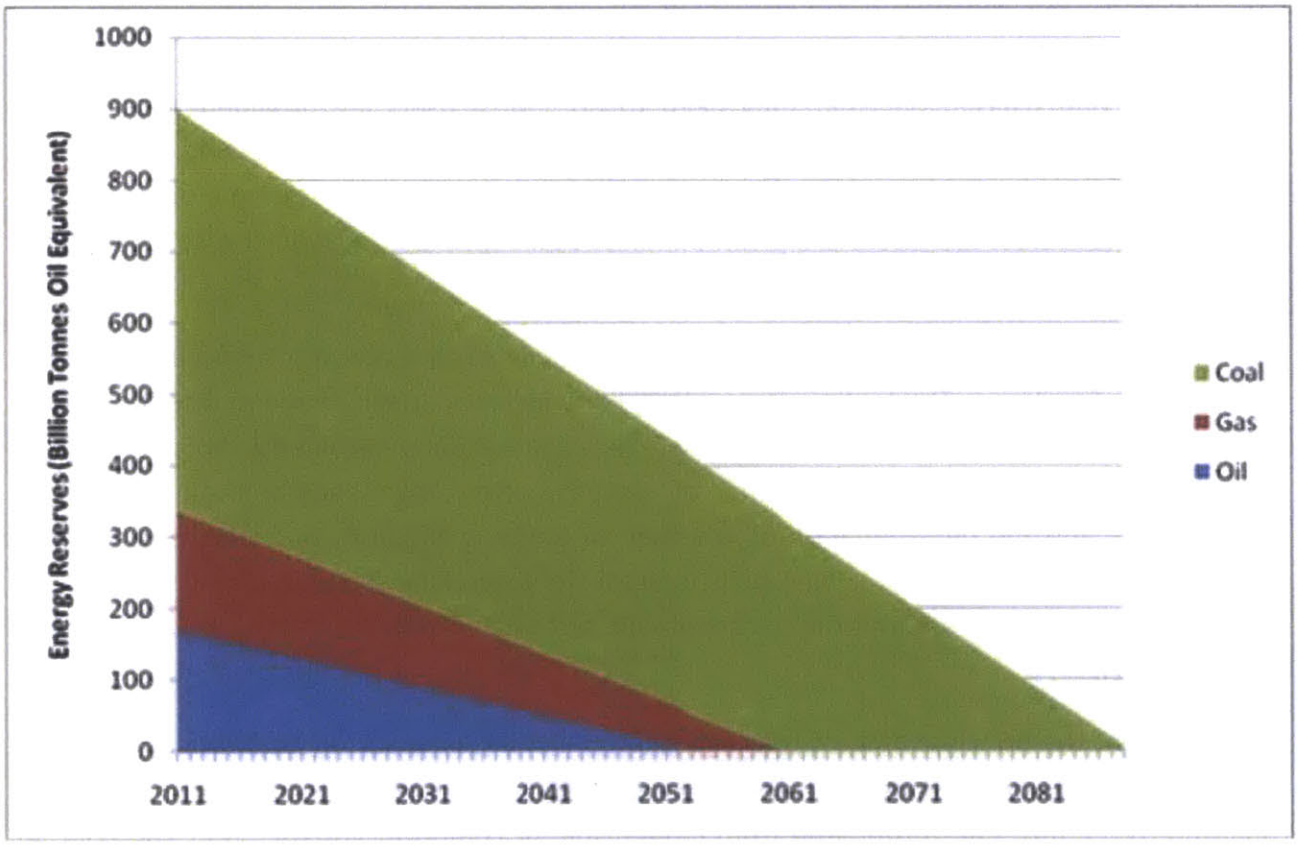

FIGURE 7-6: GRAPH SHOWING THE DECLINE OF AVAILABLE FOSSIL FUELS (WWW.ECOTRICTY.CO.UK)

7.4.3 Strategy of Least Privilege

For engineered systems to be viable, their components need to be able to interact and influence each other, to some extent, and produce high-level behavior that provides benefit to the stakeholders. Sometimes, however, components of a system can be influenced to perform actions that threaten the viability of the overall system. This is particularly a problem in systems of systems because of the independence and geographical separation of components. For example, many systems of systems allow constituent systems to interact with each over the Internet, or similar networks. These interactions are necessary; for example, Facebook would not be viable without users being able to connect with each other through the Internet, but the interactions can also result in disastrous consequences if not managed properly.

The strategy of least privilege (more commonly referred to as the principle of least privilege), was first introduced by Saltzer and Schroeder (1975), and generally accepted as being a fundamental principle to system security (Yee, 2003). It can be summarized as the following: Constituent systems should only have the minimum privileges necessary to perform their intended tasks. A privilege is the ability to influence the system in some way. Implementation of the principle of least privilege can be most easily seen in computer systems where users are assigned accounts based on what they need to do. Most users are provided with a basic account, which restricts their privileges to simple tasks such as word processing, e-mail and document printing. Typically, executing these tasks does not alter the system's operation for other users. A select few users are given administrator rights that include the ability to alter the system by installing and removing software components.

The principle of least privilege addresses two vulnerabilities that are particularly problematic for systems of systems. The first obvious vulnerability includes attacks made by hostile entities spoofing consistent systems, and performing nefarious actions within the SoS. This is a problem due to the fact that systems of systems often have geographically separated components, and the determination of whether an entity is within or outside the SoS is not necessarily trivial. The second vulnerability is to accidental actions and unintended interactions caused by legitimate users and other constituent systems with good intentions (Anderson & Mutch, 2011).

To implement the principle of least privilege, there must be some sort of authentication where the constituent systems identify themselves, and some level of restriction by which unauthorized entities are prohibited from influencing, or compromising the SoS. The issue of restricting certain entities from being able to change the system, is typically referred to as security. Depending on the system, there numerous security strategies ranging from physical locks and keys, to passwords, virtual firewalls and encryption (Stallings, 2003). However, there

are no perfect solutions, and the ability to make a large system of systems completely secure, particularly those that are open to the public such as Facebook, is non-trivial.

7.4.3.1 Historical Example: Data Breach at the Massachusetts Unemployment Agency

On April 20, 2011, a computer worm known as W32.QAKBOT infected as many as 1,500 computers in the Departments of Unemployment Assistance and Career Services at the Massachusetts Unemployment Agency (MUA), stealing extremely sensitive personal information such as addresses, social security numbers and bank information of potentially hundreds of thousands of individuals and employers. Like many computer worms, W32.QAKBOT has to be installed to run its nefarious instructions which requires administrator rights, and could have been prevented by implementing the strategy of least privilege (McFee, 2011).

7.5 Vigilance

The boundaries between what is internal and external to a complex system, and systems of systems in particular, are often not distinct. In many traditional systems, the components are together inside a physical enclosure that clearly separates what is inside the system, from what is outside. In highly complex systems, such as networks or power plants, something can be inside the system, even though it is exogenous to it. A person may be walking around inside an airport, interacting internally with various components of the system, before they board a plane and depart. These hostile entities may harm the system once they are inside, unless the principle of least privilege restricts their ability to do so. Passengers must always check in, show ID and be subject to potential search. However, even after a passenger has been authenticated, care must be taken to ensure that the authentication was not premature. Terrorists, for example, may have slipped past previous authentication checkpoints for any number of reasons. While they are waiting for their opportunity, however, they may appear as simple passengers. Only when the opportunity presents itself, they may change their mode of operation and expose themselves. Such was the case with the hijackers aboard the aircraft on 9/11, in which they passed by initial security measures at Logan and other airports, and appeared perfectly normal right up until they initiated hostile action aboard the aircraft. It is for this reason that many airports have multiple checkpoints where authentication is required (Hogan, 2001) .

Hostile actions are not limited to external entities, such as terrorists, thieves and hackers. According to a survey done by InformationWeek, authorized users and employees are the

sources of 64% of the security breaches that pose the greatest threat to small and medium-sized businesses (Figure 7-7).

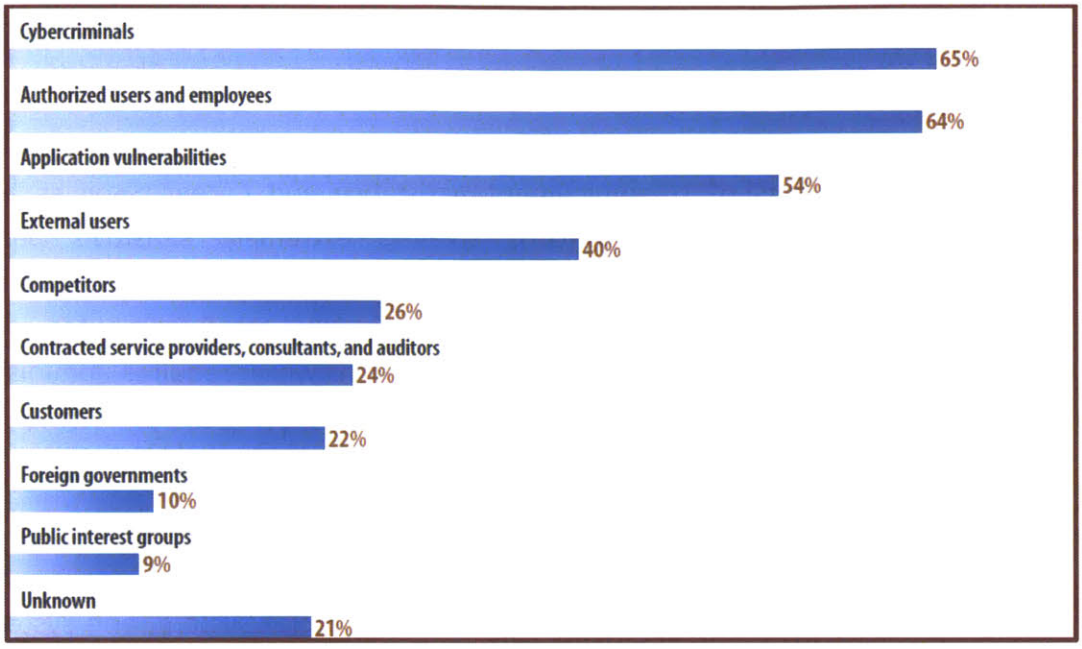

FIGURE 7-7: SOURCES OF SECURITY BREACHES (ROITER, 2011)

Since hostile actions may be taken by previously authorized entities at some later point, it is important that a system practice the strategy of vigilance. Vigilance is defined as the ability to continuously monitor internal interactions for operational violations.

7.5.1.1 Empirical Case Study: Applying Vigilance To the MarSec

The concept of vigilance can be demonstrated by the MarSec DES described in Chapter 2. In the terrorist scenario, the terrorist ship appears as a fisherman when enters the AOI, typically passing through the normal detection/identification phase as any other civilian ship would. When the terrorist is close to the port, he/she speeds up and ignores coastal regulations on its way to detonate a bomb in the port. If the MarSec is not vigilant, then by the time it notices the infraction and sends a patrol boat to intercept, the terrorist has already reached the port and detonated the bomb (a terminal event). However, the strategy of vigilance can be applied where friendly vehicles heading to port are tracked by patrol boats after they have been identified.

Performing a 2x2 DOE, where the vigilance strategy is applied against the control group, and looking at both a light traffic (λ = 11.25 ships / h) and heavy traffic (λ = 22.5 ships / h) context,

the following two contingency tables were generated after running each treatment 10,000 times Table 7-3, Table 7-4.

TABLE 7-3: CONTINGENCY TABLE OF BOMB DETONATION FOR LIGHT TRAFFIC

	Bomb Not Detonated	Bomb Detonated	Total
No Vigilance	2783	7217	10000
Vigilance	6334	3666	10000
Total	9117	10883	20000

TABLE 7-4: CONTINGENCY TABLE OF BOMB DETONATION FOR HEAVY TRAFFIC

	Bomb Not Detonated	Bomb Detonated	Total
No Vigilance	1833	8167	10000
Vigilance	3756	6244	10000
Total	5589	14411	20000

Graphing the results shows a strong difference in both contexts when the vigilance strategy is applied (Figure 7-8). Using a Fisher exact t-test, there is a significant difference between the vigilant strategy and the control in both the light traffic and heavy traffic contexts ($p < 0.0001$).

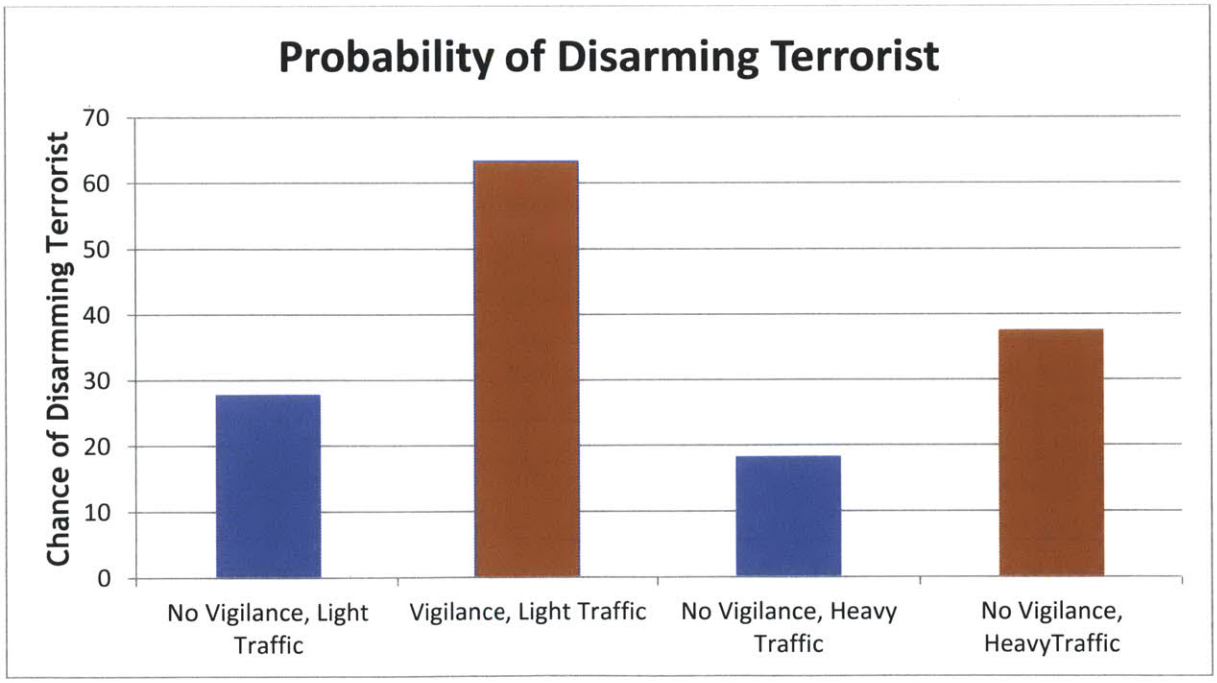

FIGURE 7-8: RESULTS OF VIGILANCE STRATEGY ON THE MARSEC DES

7.5.1.2 Difference between Redundancy, Margin, Vigilance, Layered Defense.

The strategies of redundancy, margin, vigilance and layered defense may seem similar at first and can be implemented together in various ways, but they are distinctly different. To illustrate the difference, consider the scenario of an airport trying to avoid the terminal event of a passenger boarding an aircraft with a gun. At the airport is a security clearance checkpoint, where passengers have to pass through. There are numerous security mechanisms that the airport can use, with varying levels of success (Table 7-5). Suppose a manual pat-down can detect weapons on a passenger 90% of the time, and this is the minimum requirement according to regulations. An x-ray machine has a 95% chance of detecting a weapon, but is more expensive since it requires both equipment and an operator. When used together in series, the pat-down and the x-ray machine is 99% effective at detecting weapons, and 97% effective when used in parallel.

Margin. Although a manual pat-down would satisfy regulators currently, the airport decides to use the x-ray machine for the 5% margin, just in case regulators decide to raise the minimum threshold at a later date.

Physical redundancy. The airport decides to use two x-ray machines, so that if one fails, the other can still provide adequate value. Note, having an additional x-ray machine at the security point clearance does not improve the probability of detection over having a single x-ray machine.

Functional redundancy. The x-ray machine is used as the means of detection, however if the x-ray machine becomes unavailable for any reason, or if a passenger strongly opposes, there are staff that can perform manual pat-downs on the passengers instead.

TABLE 7-5: HYPOTHETICAL DATA FOR SECURITY MECHANISMS AT AN AIRPORT

Mechanism	Probability of Detection
Manual Pat-down	90%
X-ray machine	95%
Two Manual Pat-Downs (serial)	97%
Two Manual Pat-Downs (parallel)	95%
X-ray machine and Manual Pat-Down (serial)	99%
X-ray machine and Manual Pat-Down (parallel)	97%
Two X-ray machines (serial)	95%
Two X-ray machines (parallel)	95%

Margin & physical redundancy (Layered Defense). Two pat-down inspections in serial would be an example of both margin and physical redundancy, however, it is important to note that in this case, the margin itself is not redundant. The 97% success rate (2% margin) is only achieved when both pat downs are performed. If either pat down is not performed, then the margin disappears and the probability of detection falls to just 95% (the minimum).

Margin & functional redundancy (Layered Defense). Having an x-ray machine and a manual pat-down provides both margin and functional redundancy, provided that both mechanisms are functioning. Like physical redundancy, the margin is not redundant.

Vigilance. Before boarding the aircraft, a member of the crew performs a manual pat-down on the passengers. From the perspective of the airline, this is considered vigilance because they were protecting their own context within the SoS by not assuming that the passengers crossing into their context were ok just because they were already in the secure part of the airport. From the perspective of the airport, this is a strategy that could be considered redundancy, as well as vigilance, since the inspection is done at a different time and place than the original inspection. The distinction in time and area is important, since things could have changed since the initial checkpoint. For example, a "passenger" may not even be a passenger at all, but rather someone who used a back door and an employee accomplice to enter the secure area. Another example would be that after passing the security checkpoint, the passenger may have gone to a restroom, disassembled his laptop, phone and other electronic devices and created a shank[3] from the bits of metal and glass.

7.6 DIVERSION

Some components are more vulnerable to certain perturbations than others. Often, this vulnerability is constant, such as electronics being vulnerable to electrical surges, while in other cases it is matter of circumstance, such as the influx of tasks exceeding a component's capacity. In either case, if the perturbation cannot be avoided by the SoS, then the system should protect the context of the vulnerable components by diverting the perturbation away. There are two types of diversion strategies; redirection and deflection. If perturbations are generated internally, as in the case with clients requesting operations from servers, then this strategy can be considered redirection, where the constituent systems redirect their outputs away from vulnerable (e.g. overworked) components. Part of the TCP/IP protocol includes TCP packet

[3] Since prisoners are obviously prohibited from possessing traditional weapons, a shank is a prison term for a make-shift knife, usually formed by filing a piece of scrap metal into a sharp blade.

redirection where loads on servers are mitigated by redirecting packets (i.e. information transmitted through the network) towards less utilized servers. If the perturbations originate externally, then this strategy can be considered deflection. A classic example of defection is the use of lightning rods on buildings to divert electrical energy away from critical components and into the ground where it can dissipate safely. One of the most famous lightning rods it the Empire State Building in New York. According to its official visitor website (http://www.esbvisit.com/index.html, accessed November 25, 2012), the Empire State Building is designed to serve as a lightning rod for the surrounding area and is struck by lightning about 100 times per year (Figure 7-9).

FIGURE 7-9: EMPIRE STATE BUILDING STRUCK BY LIGHTNING

7.6.1 Historical Example: Using Decoys in WWII.

Military deception is just one method of deflecting perturbations away from critical systems. Perhaps one of the greatest examples of diversion occurred during the Normandy Invasion of 1944. "Operation Titanic" involved the Royal Air Force (RAF) dropping hundreds of fake paratroopers away from where the actual invasion was occurring (Vego, 2009). The deception worked, as the dummies (Figure 7-10) successfully diverted German forces away from the actual landings used by the British and Americans on D-Day (Barbier, 2007; Ramsey, 1995).

FIGURE 7-10: DUMMY USED IN OPERATION TITANIC, 1944

7.6.1.1 Empirical Case: Applying Redirection To the MarSec

As discussed in Section 6.4, the MarSec is similar to a queuing system, where each vehicle in the MarSec acts like a server with its own service time for each task (i.e. detection, identification, observation and interception). Under normal operations, the command center assigns tasks to the vehicle that is closest to the target, regardless of whether it is busy. Unfortunately, this means that the task has to wait for some finite period before the vehicle is available. If a string of tasks, all in the same area arise, then it is possible that a significant queue will arise, resulting in a potential perturbation. A new strategy of redirection was implemented, where the command center assigns tasks to the nearest, non-busy vehicle. The strategy was implemented for the SoS in Figure 6-2 and Figure 6-3, under both regular and heavy traffic conditions. Compared with baseline SoS described in Section 6.4, the redirection strategy improves the performance of under normal conditions (probability of ID = 0.972) , but more importantly keeps the SoS viable when the heavy traffic disturbance is encountered (probability of ID = 0.

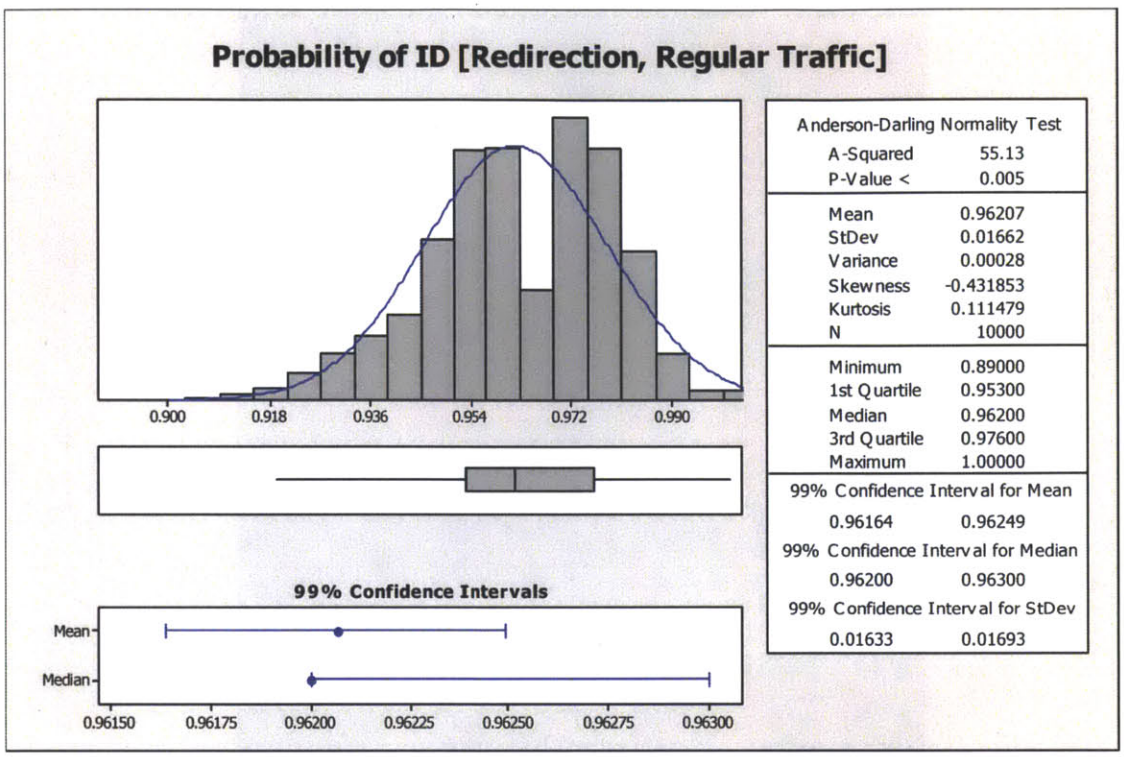

FIGURE 7-11: PROBABILITY OF ID WITH REDIRECTION STRATEGY APPLIED TO SOS EXPERIENCING REGULAR TRAFFIC

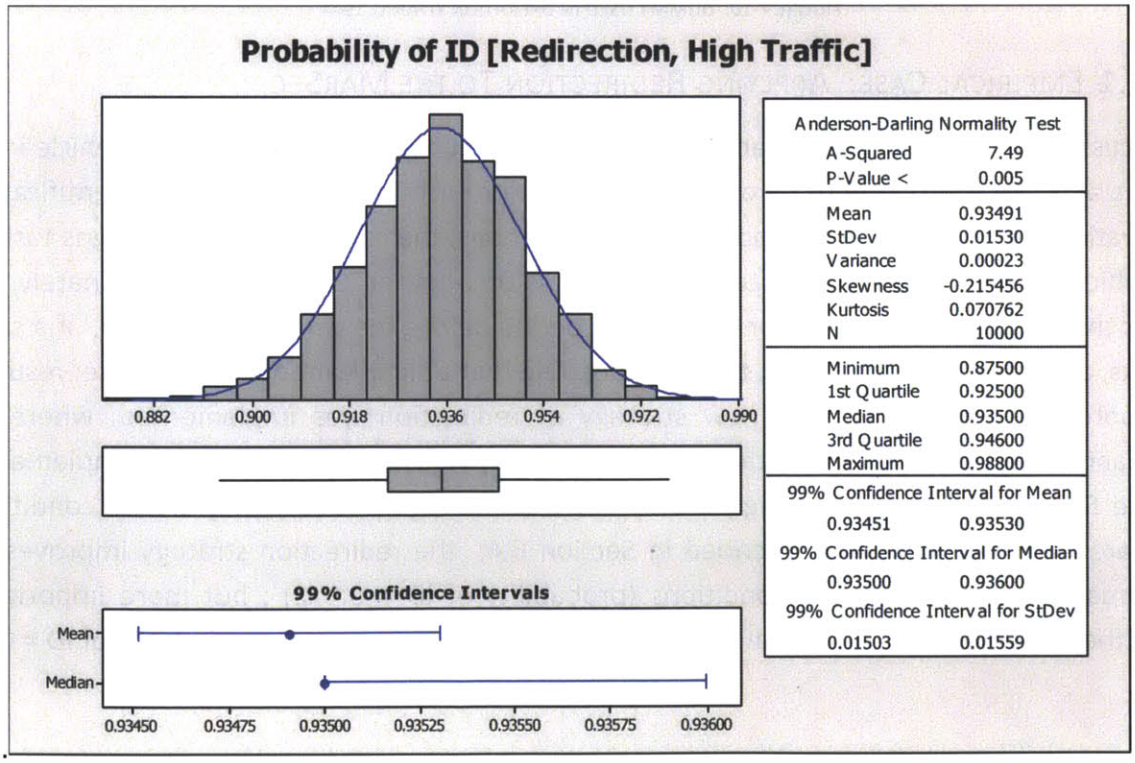

FIGURE 7-12: PROBABILITY OF ID WITH REDIRECTION STRATEGY APPLIED TO SOS EXPERIENCING REGULAR TRAFFIC

7.7 SUMMARY

Using the concept of local context and cause-effect mapping, eight additional strategies for enabling viability were introduced in this chapter. All of the strategies were empirically validated through historical cases, and two were also validated through the MarSec DES. In the next chapter, problems associated with endogenous change will be discussed, and the need for an "-ility" that restricts change, for the purpose of increasing viability, is described. Appendix A summarizes the new viability strategies introduced in this chapter.

8 ENDOGENOUS CHANGE

Up until this point, most of the perturbations that have been considered have been exogenous in nature. In some cases, such as the development of the commitment strategy, it was acknowledge that systems of systems in particular are never quite fully formed, and undergo a considerable amount of internal changes during its lifetime. While the ability to change has been identified in the literature as something to be desired, this chapter argues for the need to restrict endogenous change by providing several historical examples of voluntary changes that actually made the systems of systems they affected worse and not better.

It is likely that large-scale, complex systems will need to change at some point in their lifecycle, particularly systems of systems that have numerous, diverse stakeholders and operate in dynamic contexts.

There has been increasing research studying how, when and why systems need to change in response to shifts in context (McManus and Hastings, 2006). In particular, research includes areas such as:

- Designing latent capabilities that grant the ability to change at a later date (de Neufville & Scholtes, 2011)
- The ability to change during operational phases (Gupta & Goyal, 1989)
- The ability to change easily / rapidly (McGaughey, 1999)
- The ability to change in size only (Elkins, Huang, & Alden, 2004)

The terms "flexible", "adaptable", "agile", and "modifiable" are some of the labels that have been used to describe the ability to change. The ability to change, known as *changeability* (Ross et al., 2008), is an important quality of a system that allows stakeholders to reduce the impact of uncertainty of future contexts. Changeability addresses the issue of being able to change, but it does not specify anything about the change itself. What type of change occurs? Who performs the change? How quickly does the system change? Researchers have defined and used other change-related "ilities", such as "agility", "flexibility" and "adaptability" in the literature to address specific change-related properties that answer some of these questions. Unfortunately, these terms are not well defined in the literature (McManus & Hastings, 2006). This makes it very difficult for stakeholders to communicate the properties that they desire, and for architects to know that they have met those requirements (Ross, Beesemyer, & Rhodes, 2011). For the purposes of this dissertation, the definitions in Table 8-1 are used.

TABLE 8-1: DEFINITION OF VARIOUS "-ILITIES"

"-ility"	Description	Source
Value Robustness	The ability of a system to maintain value delivery in spite of changes in contexts or needs	(Ross & Rhodes, 2008a)
Robustness	The ability of a system to maintain its level and set of specification parameters in the context of changing system external and internal forces	(de Weck, Ross, & Rhodes, 2012; Ross, et al., 2008)
Changeability	The ability of a system to alter its form, and consequently possibly its function, or operations, at an acceptable level of resource expenditure	(de Weck, Ross, et al., 2012; Ross, et al., 2008)
Flexibility	The ability of a system to be changed by a system-external change agent with intent	(de Weck, Ross, et al., 2012; Ross, et al., 2008)
Adaptability	The ability of a system to be changed by a system-internal change agent with intent	(de Weck, Ross, et al., 2012; Ross, et al., 2008)
Evolvability	The ability of a architecture to be inherited and changed across generations (over time)	(Beesemyer, Fulcoly, Ross, & Rhodes, 2011; de Weck, Ross, et al., 2012)
Survivability	The ability of a system to minimize the impact of a finite duration disturbance on value delivery	(de Weck, Ross, et al., 2012; Richards, 2009)
Versatility	The ability of a system to satisfy diverse needs for the system without having to change form (measure of latent value)	(de Weck, Ross, et al., 2012; Ross, et al., 2011)
Scalability	The ability of a system to change the current level of a system specification parameter	(de Weck, Ross, et al., 2012; Ross, et al., 2008)
Modifiability	The ability of a system to change the current set of system specification parameters	(de Weck, Ross, et al., 2012; Ross, et al., 2008)
Interoperability	The ability of a system to effectively interact with other systems	(de Weck, Ross, et al., 2012; Ross, et al., 2011)
Reconfigurability	The ability of a system to change its configuration (component arrangement and links)	(de Weck, Ross, et al., 2012; Ross & Rhodes, 2011)
Agility	The ability of a system to change in a timely fashion	(de Weck, Ross, et al., 2012; Ross, et al., 2011)
Extensibility	The ability of a system to accommodate new features after design	(de Weck, Ross, et al., 2012)

Change-based "-ilities" are increasingly realized as desirable properties in systems (McManus, Richards, Ross, & Hastings, 2007). However, not all changes are positive and implementing the strategy of least privilege (Section 7.4.3) requires that restrictions are placed on what can change, and who can perform those changes. Most of the definitions in Table 8-1 seem to encourage change in various forms and none specifically address the issue of controlling endogenous change (although value robustness and survivability indirectly do by discussing value impact in general).

There are many types of change to consider, but they can be broadly categorized into intentional changes and unintentional changes.

Intentional Changes. Intentional changes are those that decision makers choose to execute, often in response to a shift in context or stakeholder needs. An example of an intentional change would be if airline engineers added AC power ports to all the seats in an existing aircraft to keep up with market demand. Intentional changes are typically the types of changes described by researchers when they discuss changeability, flexibility, agility and other types of change-related system properties. However, the concept of "intentional" changes should be clarified further than has traditionally been the case. Sometimes, decision makers may want to implement a change to improve value delivery, but end up reducing it instead. This is because as the system complexity grows, and as more changes are made to the system, it becomes increasingly difficult to verify and validate the effects that changes will have. Sometimes, the original architects and engineers who designed the system and really understood the CONOPs may no longer be available to evaluate impacts of planned changes. Subtle assumptions that were not made explicit, may be violated with any new changes and could prove to be not only disastrous, but difficult to find until it is too late. This is particularly a problem with systems of systems that have autonomous constituent systems with emergent behavior that is notoriously difficult to model. The larger the system, or the more expensive, or the more stakeholders involved, the likelihood of someone wanting to accept responsibility for any one particular change decreases. Thus, there may be a substantial bureaucratic process involved for any complex system that makes it extremely time-consuming and costly for all but trivial changes to be made. This "red tape" can seriously impair the ability of the system to respond quickly to changes in context, especially if it takes years to approve any significant changes.

Unintentional Changes. Unintentional changes are those that the system is "forced" to undergo, that is, changes that occur whether the decision makers want them to or not. Some unintentional changes just happen without any intervention or "approval" from the stakeholders. A collision with a bird may cause an engine to malfunction, or a power outage may cause a monitoring system to fail. Other unintentional changes happen after decision

makers authorize them, but only because they have to, in response to some other event beyond their control. A labor strike may force a city's engineers to shut down the subway, or a new environmental regulation may cause a chemical plant to replace certain older equipment with newer, more environmentally-friendly models. Since unintended changes are likely to be problematic, system architects typically try to minimize their causes and effects as much as possible. Unintentional changes that are the result of endogenous forces, such as random component failure, are the focus of reliability engineering (Leveson, 1995), whereas system survivability and robustness strive to prevent, mitigate and recover from unintended changes caused by exogenous forces such as lightning strikes and resource shortages.

8.1.1 Change Causing Large-scale Systems of Systems to Fail

Three recent examples of endogenous changes causing large-scale systems of systems to fail are briefly discussed below.

8.1.1.1 Hurricane Katrina and the New Orleans Emergency Response Service

On August 29, 2005, Hurricane Katrina made landfall in southeast Louisiana, causing one of the deadliest and most expensive disasters in North American history. Normally, local emergency services for a particular area belong to a system of systems that is coordinated through a central dispatch (i.e. "911"). However, when Hurricane Katrina struck, the efforts of all of these independent services, with their own personnel, vehicles, communication systems and supporting equipment, were to be coordinated and supervised by the Federal Emergency Management Agency (FEMA) (Figure 8-1). Although FEMA was intended to improve emergency response, the change in management of the independent services had the opposite effect. The poor emergency response that the police, fire and ambulance provided in the aftermath of Katrina was widely condemned by the media and led to the resignation of FEMA director Michael Brown (Stevenson, 2005).

8.1.1.2 GPS Dispatch Upgrade to Taiwan Taxicab Services

In 2001, Comfort Cabs of Singapore updated their taxicab dispatch system from radio-paging to satellite-based GPS with great success, improving both accuracy and productivity of its taxicab network (Liao, 2003). Since Taipei is similar in size and traffic to Singapore, Taiwan Taxi Company assumed that switching to a GPS-based dispatch would be equally successful. However, when the upgrade was made, a large number of the independent Taiwanese taxicab operators rejected the new dispatch system. Although Taiwan Taxi expected to attract more than 20,000 operators to sustain the company by the fourth year, they were only able to attract 7,000 members during an 8 year span (2001 to 2009) (Hsiao, Compeau, & Hou, 2010).

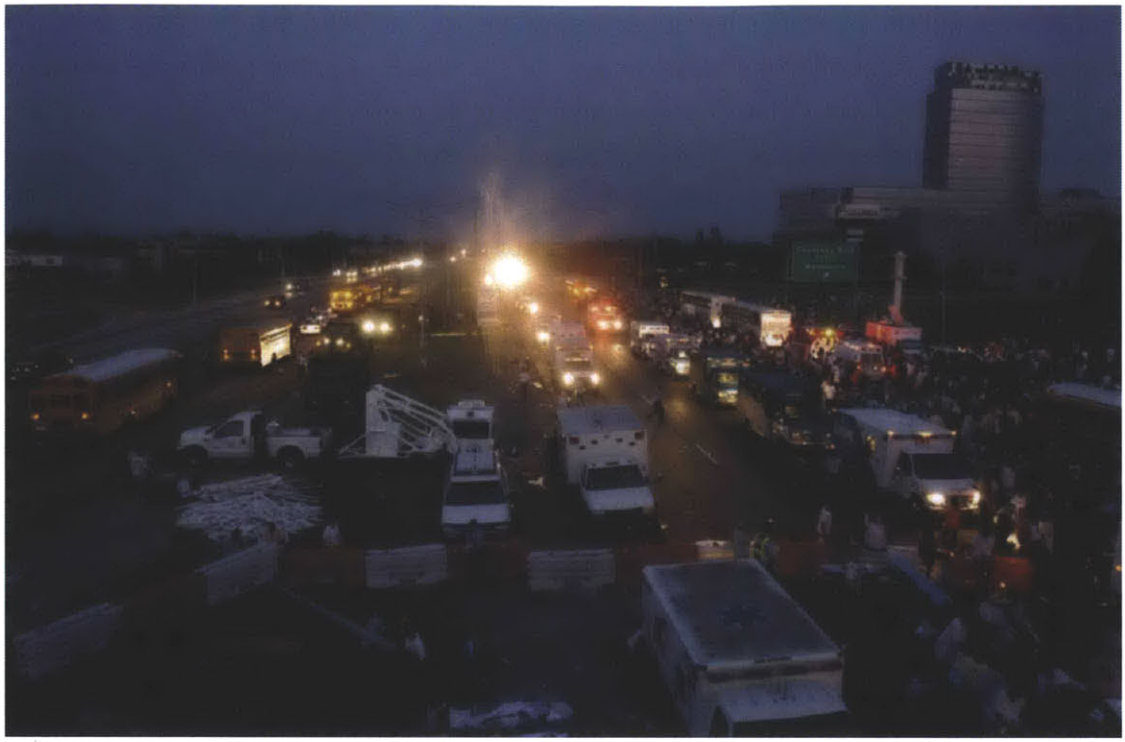

FIGURE 8-1: HURRICANE KATRINA RESCUE OPERATIONS, NEW ORLEANS, 2005

8.1.1.3 Computer Aided Dispatch and the London Ambulance Service

The London Ambulance Service is the largest free to patient ambulance services in the UK, responding to over 1.5 million calls a year in the London Area. On October 29, 1992, a new Computer Aided Dispatch (CAD) was introduced to optimize ambulance allocation and reduce wait times. Overloaded with calls and working with imperfect information, the CAD system failed to dispatch ambulances in a timely manner and was blamed by the media for as many as 30 deaths (Long, 2009). It was removed several days later, in what became one of the most well-known examples of large-scale IT project failures in history (Beynon-Davies, 1999).

The Taiwan Taxi Company, London Ambulance, and Hurricane Katrina emergency are examples of system of systems that failed to provide acceptable value to stakeholders after changes were made. These systems of systems were composed of geographically disperse, socio-technical, independent constituent systems, that collaborative together to achieve some overall mission objectives and/or mutual benefit. In the Hurricane Katrina example, the SoS failed when it was forced to change management CONOPs in response to context changes brought about by the hurricane. In the Taiwan Taxicab Company and London Ambulance situations, the systems of systems failed when the stakeholders voluntarily decided to make endogenous changes to take advantage of some new technology.

8.2 Endogenous Change within the MarSec

The MarSec is complex system that can change in many ways, through form and CONOPs, at the system and SoS level. In terms of form, the number of vehicles that make up the MarSec can vary between some minimum necessary to achieve the necessary task throughput, to some maximum defined by the space in the hangers and docks. This is an example of a SoS-level change. At the system level, the actual vehicles themselves can change from manned to unmanned, and from one payload type to another. The CONOPs can change as well, such as whether the vehicles operate in a mechanistic (i.e. specialize in one task) or organic (i.e. perform multiple tasks) manner. However, as with any complex system, care must be taken before any change is attempted, for there are many details and interactions that may be critical, but not salient.

An example of the need for restricting change in the MarSec would be the following. Suppose that the MarSec consisted of several large, medium-range UAVs that had a communication range that spanned the entire AOI. A command center in the middle of the AOI assigns tasks to the UAVs via a wired connection to the ground control stations. Suppose at some time after the MarSec has been designed and implemented, a political opportunity arises to exchange UAVs with another government agency. Instead of replacing them with the same UAVs, the new owners of the MarSec decide to "upgrade" the SoS by changing the UAVs into smaller, swarms of tactical UAVs. The tactical UAVs are faster, cheaper and have just as good a payload, and do not require the use of a command center to coordinate task assignment. The disadvantage is that their line-of-sight (LOS) range is only half of what the old, larger UAVs had. To get around this issue, the owners decided to split the AOI into two zones (Figure 8-2) and have half of the UAVs operate in one zone and half in the other.

The CONOPs makes sense on paper, so the owners decide to move ahead and make the "upgrade". Unfortunately, without realizing all of the details surrounding the original design and context, the new system fails. The reason it fails is because the process of detection, identification and interception is a linear process (i.e. first detection occurs, then identification, then interception if necessary) and the target ships are moving through the AOI as it occurs. If the ship makes it to the midway point of the AOI, they then cross into the other zone. Although there in intra-swarm communications between the UAVs, there is no inter-swarm communication since they do not have the range to traverse the entire AOI. If there is no communication sharing between the two swarms of UAVs, then any information about a target that crosses the zone change line will be lost and the new swarm will have to start the process at the first step of detection. This means that for certain fast ships, and busy contexts, an

unacceptable number of ships will be able to make it through each sub-zone, and hence the entire AOI, before the entire process can be completed by either swarm of UAVs.

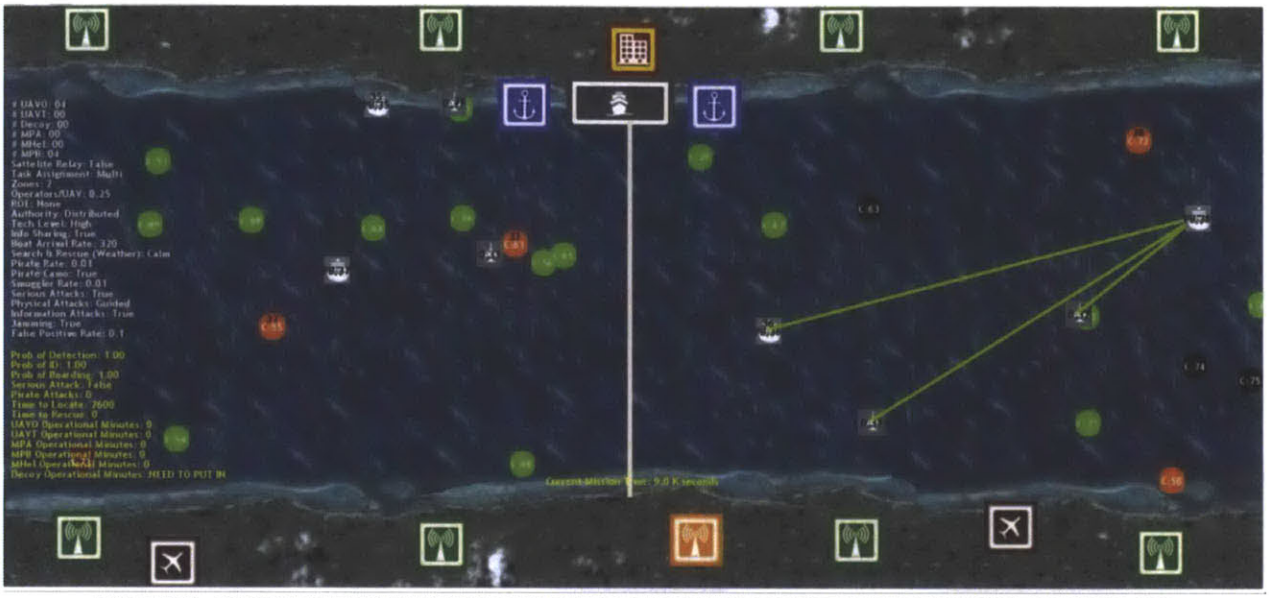

FIGURE 8-2: SCREENSHOT OF MARSEC SOS DES SHOWING TWO DISTINCT ZONES SEPARATED BY A WHITE DIVIDING LINE

This change, as well as those described in the case studies mentioned in this chapter, should not have been allowed by the architects who designed the systems in the first place. The stakeholders did not understand the complexity of the systems, and what should have been positive upgrades, ended up making the systems unviable. In the next chapter, a new "ility" is defined that sets limits on endogenous change and actually increases viability as a result.

8.3 SUMMARY

This chapter showed that while changeability is often a desirable and even necessary quality for a system to have, it is difficult for decision makers to know whether their changes will have unintended consequences, perhaps many years after the original system has been built. Several historical examples of spectacular failures are presented in this chapter to highlight that systems of systems are particularly vulnerable to this problem. In the next chapter, the concept of pliability will be introduced, which is an "ility" that both facilitates and restricts a system's ability to change.

9 SYSTEM ARCHITECTURE AND PLIABILITY

In the previous chapter, several examples of harmful endogenous changes were shown, where the stakeholders had good intent, but ended up harming the system instead of improving it. In this chapter, a new "ility" is introduced that sets limits on the amount of change that systems will be allowed to undergo. To help define this "ility", and clarify what is meant by "allowed", the idea of a system architecture is necessary.

9.1 SYSTEM ARCHITECTURE

The Department of Defense defines system architecture as a framework or structure that portrays relationships among all the elements of the system (Department of Defense, 2008a). Crawley (2004) defines system architecture as the description of the components of a system and the relationships between them. In this sense, the system architecture includes both what the system is composed of, and how these components work together. At a minimum, the system architecture should describe the functional behavior of the system, i.e. which tasks the components perform, as well as when and how the tasks are executed, similar to the Capability and Operational Views found in the Department of Defense Architectural Framework 2.0 (Department of Defense, 2010). For simple systems, the emergent behavior of the system is often predictable from the system architecture, through aggregating the functional behaviors of the components. For more complex systems, emergent behavior is more difficult to predict and the primary technical role and responsibility of a system architect is to ensure that the interconnections of the components achieve the necessary "system functions" to meet the overall purpose (Maier & Rechtin, 2000).

Although there are several definitions already in the literature, for the purposes of this dissertation, *system architecture* is defined as a collection of components and an associated concept of operations (CONOPs), whose instances provide some value, within a particular context. The terms *components*, *CONOPs* and *instances* are described below.

9.1.1 COMPONENTS

The *components* of a system architecture describe what the system architecture is composed of. The components include a description of all the Operational Elements (OEs) that perform, or could perform some function, as well as any additional entities that belong to the system. OEs are defined by their properties and capabilities. Properties are the physical characteristics of the components, such as length, mass and color. The capabilities describe the functional behavior that the OEs can perform, such as fly, transmit data, or store energy. The capabilities

describe what the OEs can do, but not necessarily what they actually do. Unused capabilities are latent capabilities, representing potential value in the system architecture. For example, a UAV may be able to relay communications to other vehicles, however in cases where it is performing solo missions, this capability is not used.

9.1.2 Concept of Operations

The *Concept of Operations (CONOPs)* describes what the system does. Specifically, the CONOPs describes the tasks that the OEs perform, as well as the logic for how the OEs perform these tasks, within a particular context. While some definitions of CONOPs assume that operators are external to the system, this definition includes operators as part of the system (in fact, they are considered OEs). Thus, the actions the operators perform within the system are part of the CONOPs, as well as the actions of other OEs.

Tasks. All systems do something, and that something can be referred to as tasks. Tasks can be broadly defined, such as "perform maritime surveillance" or they can be specific, such as "accelerate vehicle to 93.7 mph". Typically, broadly defined tasks are broken up into a set of smaller, specific sub-tasks. Each of these sub-tasks can then be further broken down into more specific sub-tasks. Typically, broader tasks have multiple OEs involved in performing those tasks, while lower-level sub-tasks are often performed by a single OE.

Logic: Merely specifying a set of tasks will not be sufficient to describe the manner by which the system achieves its purpose. Additional information, in the form of logic, must be specified as well.

- *Order.* The tasks must be given an order in which they are performed. Order does not have to be strict, as some tasks may be performed concurrently or even randomly.
- *Task Assignment.* The logic of the CONOPs specifies which OEs perform which task. Sometimes the output from one OE becomes the input to another OE. Specifying which OE performs a task is especially important when multiple OEs share the same capabilities, such as one might expect in a complex SoS.
- *Conditions.* Many tasks have conditions associated with their execution. Conditions may exist to determine whether a task gets performed at all. Conditions may also specify how a task gets executed. For example, if a hostile, anti-aircraft gun is present in the area, a UAV will fly at a higher altitude than it normally would. Other common conditions include starting and stopping conditions for tasks that are not being performed continuously.

9.1.3 INSTANCES

In many system architectures, there will be fixed and variable parameters. In a certain maritime security SoS architecture, as an example, the system architects may deem four patrol boats, six manned helicopters and two ground radar towers as fixed parameters. This means that any marine security SoS that adheres to this system architecture must have these components. Similarly, the architects may specify a certain CONOPs as fixed as well, such as the fact that the radar towers perform detection, the helicopters perform identification, and the patrol boats perform boarding. However, the architects may include some variability in the components and/or CONOPs specified in a system architecture, to satisfy a diverse set of environments and/or stakeholder preferences. For example, in the same maritime SoS, the system architecture may also require UAVs. The UAVs may be either short-range or long-range and there may be a command center or not. The CONOPs may also vary as well. For example, the tasks may be shared amongst the UAVs in a distributed authority manner, or they may be assigned via a central authority. Another CONOPs parameter is whether to separate the AOI into one or two zones and split the SoS accordingly.

Due to the variability in some of the parameters, most system architectures can allow systems to have different sets of components and CONOPs. A *design* is a specific set of components (known as its *form*) and a specific CONOPs (known as its *mode of operation*) that belongs to a system architecture. Thus,

$$D_{ij} = [F_i, MO_j] \tag{1}$$

where D_{ij} is the design consisting of the *ith* form specified in the set of components and the *jth* mode of operations specified in the CONOPs of some particular system architecture. In this maritime security SoS example, there are 16 possible designs (4 different variables to be specified, at 2 levels each).

A design is still conceptual and it only exists as an idea. However, a *system* is an actual physical realization of a design that provides value to stakeholders. To clarify the difference between system, design and system architecture, three types of system architectures will be examined; the Apple iPhone, the Nimitz class aircraft carrier and the International Space Station.

9.1.3.1 APPLE IPHONE AND SYSTEM ARCHITECTURES OF MASS PRODUCED SYSTEMS

The Apple iPhone is a system architecture consisting of core components (touchscreen, RAM, communication technology, etc.) and a CONOPs (what happens when a button is pressed, how the phone doubles as a MP3 player, etc.). The iPhone 4 and 4S are similar, but they are different architectures. The 4S has an upgraded communications technology, as well as better

camera, and it supports voice-activated commands via Siri. It is also important to point out that there is no official (i.e. Apple approved) way of turning a 4 into a 4S or vice versa. As of 2012 in the United States, iPhone 4S comes in three different forms on three different cell phone carriers. The forms are 16GB/32GB/64GB RAM and the carriers are AT&T/Sprint/Verizon. Although the hardware is the same, an iPhone 4S behaves differently depending on the network technology that the carriers use. Currently, AT&T uses an advanced HSPA network, while Sprint and Verizon use an older EVDO network. Due to the differences in the networks, a user can make a phone call and surf the Internet simultaneously on an AT&T iPhone 4S, but cannot with a Sprint or Verizon iPhone 4S. The iPhones 4S are "locked" to a particular carrier, meaning that a Sprint iPhone 4S cannot be used on an AT&T network, unless it is "unlocked" by Sprint – a process that is possible, but not always available to the consumers.

There are millions of iPhone 4/4S systems in the US, each with their own unique serial number. However, each one of the iPhone 4/4S cell phones in the US is an instance of one of the designs shown in Table 9-1. Even though many iPhones are identical in design, they are not the same system, just as two identical twins are not the same person. However, two systems with the exact same design can be considered equivalent.

TABLE 9-1: THE 18 UNIQUE DESIGNS OF THE APPLE IPHONE 4S IN THE US, 2012

Design	RAM (Form)	Network (Mode of Operation)
1	16GB	AT&T HSPA
2	32GB	AT&T HSPA
3	64GB	AT&T HSPA
4	16GB	Sprint EVDO
5	32GB	Sprint EVDO
6	64GB	Sprint EVDO
7	16GB	Verizon EVDO
8	32GB	Verizon EVDO
9	64GB	Verizon EVDO

9.1.3.2 System Architectures of Large, Complex Systems

For mass produced systems such as an iPhone, it is unusual and impractical to have only a single instance for a given design. However, for larger and more complex systems, it is far more likely that systems are unique instances of a particular design. For example, a Nimitz-class aircraft carrier is the system architecture of which the USS Ronald Reagan, USS Theodore Roosevelt and USS Carl Vinson ships belong. While these carriers adhere to the same system architecture,

there are some differences that make them unique. While the USS Ronald Reagan carries 2 RIM-7 Sea Sparrow and 2 RIM-116 missile systems, the USS Theodore Roosevelt carries 3 RIM-7 Sea Sparrow missile systems and 3 Phalanx anti-ship, Close-In Weapon (CIWS) systems. In the extreme case, there may only be one instance of a particular system architecture in existence, so the system, design and architecture are all the same. Such is the situation with the International Space Station (Table 9-2) or Facebook.

One of the primary differences between smaller, mass-produced systems and larger, unique systems of systems is the viability expectations in regards to the systems lifespan. A mass produced smartphone, for example, is probably only expected to be viable for several years before it is completely replaced by a new model. A larger SoS like Facebook, however, may be expected to last indefinitely, even if its individual constitute systems (users, servers, source code) gets replaced several times over. The longer a system is expected to exist, the harder it will be for its original system architecture to remain viable since it will be very difficult to architects to predict so far into the future during the design stage. For this reason, longer lasting systems should probably be changeable and/or versatile to be able to meet new demands. However, care must be taken to ensure that those changes do not violate the existing concept of operations and make the system unviable overall.

TABLE 9-2: EXAMPLES OF SYSTEMS, DESIGNS AND SYSTEM ARCHITECTURES. (ADAPTED FROM BEESEMYER, 2012)

<table>
<tr><td>CONCEPT</td><td>iPhone</td><td>Aircraft Carrier</td><td>Space Station</td></tr>
<tr><td>SYSTEM ARCHITECTURE</td><td>[iPhone 4, iPhone 4S]</td><td>[Nimitz, Enterprise, Ford]</td><td rowspan="3">International Space Station</td></tr>
<tr><td>DESIGN</td><td>[Black/White, 16GB/32GB/64GB, AT&T, Sprint, Verizon]</td><td rowspan="2">[USS Ronald Reagan, USS Theodore Roosevelt, USS Carl Vinson]</td></tr>
<tr><td>SYSTEM</td><td>Brian's iPhone.
(Black, 64GB, Sprint version)</td></tr>
</table>

9.2 PLIABILITY

The English word *pliable* is defined as "capable of being bent or flexed or twisted without breaking" (WordNet, 2011). For systems engineering, *pliability* can be defined as "the ability of a system to change, without breaking its system architecture". The *pliable set*, is the set of all validated designs that can be transitioned to, within a system architecture. Thus, we can write that for system architecture *X:*

$$Pliable\ set\ of\ X = \{D_{11}, D_{31}.., D_{nm}\} \quad (3)$$

where n is the number of sets of components and m is the number of modes of operation specified in the form and CONOPs respectively of that system architecture.

9.2.1 Architecture and Instance Transitions

Pliability is the property of a system to be able to switch to other viable instances, specified by the architecture's pliable set. If a system architecture has multiple allowable designs, then a system is always an instance of one of these designs at any time t, and can transition to the other instances defined in the pliable set of its system architecture, while remaining the same system. If a system transitions to an instance outside of its system architecture, then it becomes an unapproved system. Whether this new system will be viable is unknown (at best), since it does not belong to the set of viable instances, defined by the architects responsible for its value delivery. Thus, it is usually in the best interest of the architects and decision makers not to let systems change into instances outside of their system architecture.

9.2.1.1 Viable Designs and Allowable Transitions

Any design that is part of the pliable set for a system architecture is viable, meaning that the system architects have approved that particular instance. A viable design is one whose instances provide acceptable value to the stakeholders. In the maritime security SoS example described earlier, there were four parameters in the system architecture (UAV Range, Presence of Command Center, Number of Zones, Authority) with each parameter being two levels for a total of 16 possible designs. However, not all 16 designs are viable. Obviously, the central authority CONOPs requires a command center component to which the detection and identification information is sent, and from which the orders are given. Therefore, all designs that specify a central authority CONOPs, must also include a command center in the form. In addition, since the AOI is too large, short range UAVs cannot effectively communicate with each other if they are located at opposite ends of the AOI. Therefore, a distributed authority with two zones is not viable. Removing these unviable designs, leaves 10 possible designs (Table 9-3).

An allowable transition is one where the switch from one design to another is allowed by the system architects. There are a number of reasons why a system transition may not be allowed by the system architects. A system's ability to transition to other systems is referred to as system changeability (Ross, 2006). Changeability is affected by a number of factors including stakeholder support and cost. A laptop computer can be turned into a fighter jet, step-by-step, given infinite time, resources and stakeholder support. The question of whether system

architects would allow such a transition would likely depend on the answer to the following questions: Is there enough time to transition from a laptop to a fighter jet? Is the cost of transition (probably at least the cost of building a fighter jet from scratch) worth it? Does it matter if none of the original laptop components are used in the final fighter jet design (i.e. they all get removed or replaced at some stage in the transition)? Does it matter whether or not the intermediate stages between the original laptop and the final fighter jet are still viable systems themselves?

TABLE 9-3: VIABLE DESIGNS FOR THE MARITIME SECURITY SOS EXAMPLE

DESIGN	I	FORM	J	MODE OF OPERATION
D_{11}	1	4 Long-Range UAVs, No Command Center	1	1 Zone, Distributed Authority
D_{12}	1	4 Long-Range UAVs, No Command Center	2	2 Zones, Distributed Authority
D_{21}	2	4 Long-Range UAVs, Command Center	1	1 Zone, Distributed Authority
D_{22}	2	4 Long-Range UAVs, Command Center	2	2 Zones, Distributed Authority
D_{23}	2	4 Long-Range UAVs, Command Center	3	1 Zones, Central Authority
D_{24}	2	4 Long-Range UAVs, Command Center	4	2 Zones, Central Authority
D_{32}	3	8 Short-Range UAVs, No Command Center	2	2 Zones, Distributed Authority
D_{42}	4	8 Short-Range UAVs, Command Center	2	2 Zones, Distributed Authority
D_{43}	4	8 Short-Range UAVs, Command Center	3	1 Zone, Central Authority
D_{44}	4	8 Short-Range UAVs, Command Center	4	2 Zones, Central Authority

Although there are 10 viable designs in the maritime security SoS example above, not all designs may be reachable from all other designs. Determining which designs are reachable from other designs, is how the system architects define the pliable sets and the system architectures. Going from a command center to no command center is trivial, however going from no command center to a command center is not, if the command center is not already available. Similarly, transitioning from long-range to short-range UAVs and vice versa, can be costly if acquiring and preparing the vehicles require a significant cost. The CONOPs transitions are typically much easier, although there may be a non-trivial cost associated with re-training. Due to the inability to transition from the no-command center designs to the command center

designs, as well as the inability to transition between the different types of UAVs, there are actually four pliable sets here, as shown in Figure 9-1.

Each pliable set in Figure 9-1 defines a different architecture. Note than system architecture X is a subset of system architecture A, while system architecture Y is a subset of system architecture B. The reason why X is distinct from A, and Y is distinct from B, is simply that systems which start as designs in X or Y will never be able to transition out of X or Y, and thus there is no point validating any other instances in A or B.

9.2.2 Non-pliable Transitions

If a system changes in ways allowed by its system architecture, then this is a pliable change and something that is allowed, if the context change requires it. This is one of the main attractions of having a pliable system. However, if the new instance is not allowed by the system architecture, then this change can be considered a "hack". Sometimes hacks work, such as when users "jailbreak" their Apple iPhone 4S to run on unauthorized networks. Sometimes, however, the hacks fail to work, primarily due to the complexity of the system and the fact that the changes were never considered or approved by those who were responsible for building it in the first place. In some cases, non-pliable changes can result in a system that becomes a design of a different system architecture, if such an architecture is already defined, or requires a new system architecture to be defined. This is a more serious change that often requires stakeholder approval and input from system architects. If the new system architecture is validated by the system architects, then, like pliability, this type of transition is a special case of changeability known as *evolvability* (Beesemyer, Ross, & Rhodes, 2012).

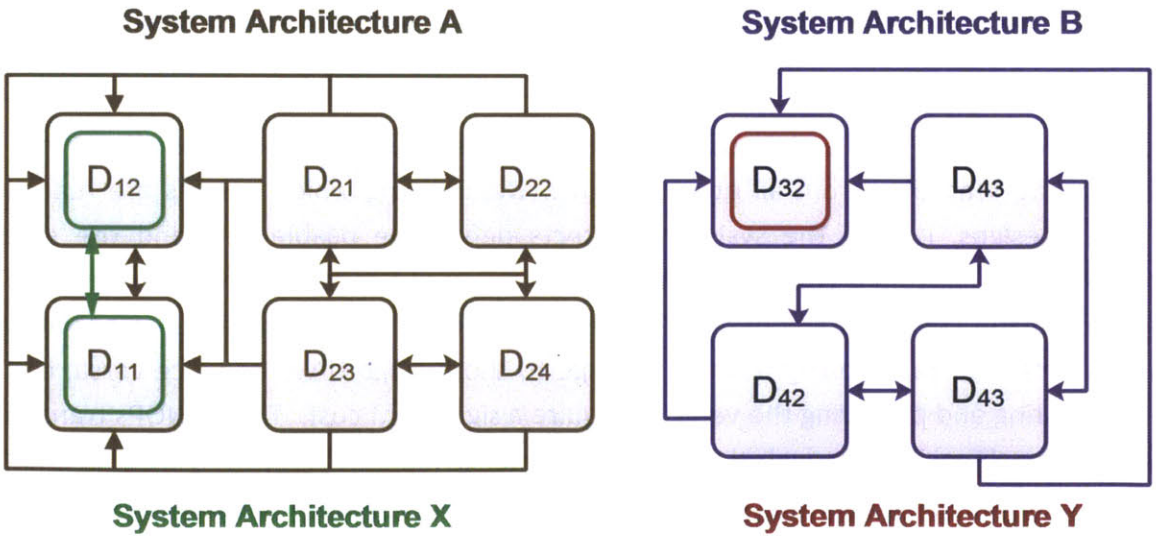

FIGURE 9-1: THE PLIABLE SETS OF THE MARITIME SECURITY SOS EXAMPLE

Another type of system transition is degradation. This type of transition is when the system changes in a way that deviates from any design within the system architecture and fails to provide value accordingly. For example, losing one of the four Long-Range UAVs in system architecture A would result in a degraded system, since there were no designs that have only three Long-Range UAVs. For the system to return to viability, it would need to replace the UAV and restore the system to its original system architecture, or transition to some other design that is viable through adaptation. Figure 9-2 shows the types of transitions discussed in this section.

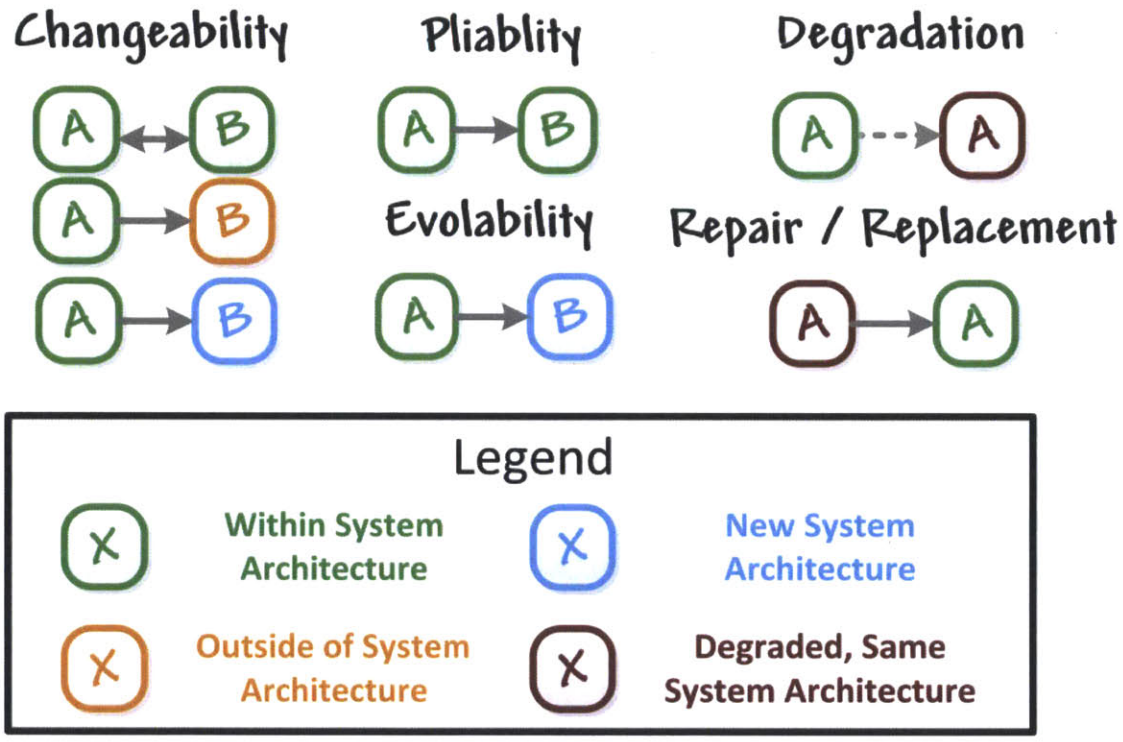

FIGURE 9-2: SYSTEM TRANSITIONS

9.2.3 Architecture Transitions and Viability

Survivability requires the effective handling of change within systems, either by preventing, mitigating or recovering from unwanted change caused by perturbations, or intentionally changing in response to new contexts. By requiring that systems be pliable, viability can be increased in three different ways; (1) By requiring system architects to go through a design cycle that explores the limits of their systems, (2) by increasing the number of viable instances that system can transition to, and (3) pre-validating change options to reduce the time and effort required for stakeholders to approve changes, allowing them to be implemented quicker and easier. These benefits are discussed below:

9.2.3.1 Disturbance Discovery and the Pliability Design Cycle.

In defining the pliability of systems, the architects must specify what is allowed to be changeable within the system, which means they must provide some guarantee that such changes will not adversely harm its value delivery. To do this, they need to examine the parameters within the system architecture, both in components and in CONOPs, to see what can vary, what can't, and what the limits should be. This exercise forces architects to think about the causes and effects of change within their system, perhaps at a level they normally would not have. In the maritime security example, system architects would have to determine how many UAVs the SoS would support. This is because it is possible that some disturbance may cause UAVs to be unavailable, or perhaps an increase in traffic would necessitate adding additional vehicles to meet the demand. In this example, they might realize that due to bandwidth constraints, the maximum number of UAVs would be 12. At this point, they can analyze whether these constraints will satisfy their value robustness requirements. What conditions would have to exist for the system to need more than 12 UAVs? Similarly, what is the likelihood, given labor shortages, bad weather, random component failures, and other disturbances, that the SoS may find itself with less than 4 Long-Range UAVs or 8 Short-Range UAVs? What is the minimum number of UAVs required to continue to provide acceptable value? In an iterative fashion, as more disturbances and change agents are considered, changes in a candidate architecture are made, which lead to an expansion or reduction in the pliable set, from which feasible systems are evaluated. Thus, the process of defining an architecture and its pliable set becomes a cycle (Figure 9-3), where system architectures and systems are analyzed in the presence of disturbances and change agents, through the concept of pliability.

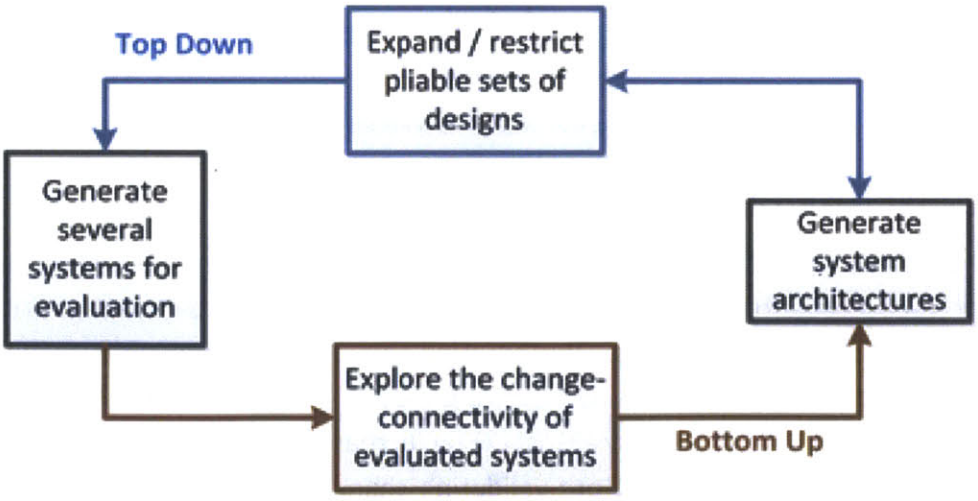

FIGURE 9-3: SYSTEM CYCLE FOR PLIABILITY

9.2.3.2 Increasing Available Options

If a system is pliable, this means it can change (to some extent) and still maintain acceptable performance under the original contexts considered. Survivability is increased automatically as the pliable set of a system architecture expands, simply because the outcome system state of an unintentional change is more likely to be contained within the pliability of the system it affects. Returning to the maritime security example, design D_{23} in system architecture A is more survivable than design D_{11} in system architecture X, simply because of the availability of the command center and the central authority CONOPs. If anything happens to the command center, or if the central authority CONOPs becomes unviable, then an instance of D_{23} can just transition into D_{21} or D_{11}. However, if a perturbation suddenly makes the distributed CONOPs unviable (for instance, if there is a security breach and a hostile entity gets the ability to spoof UAV-to-UAV communications), then an instance of D_{11} cannot transition to any other viable designs and thus a system that is part of that system architecture will be unviable for that particular perturbation.

Disturbances may break the architecture of a particular system by altering one of the fixed parameters, or by forcing a parameter to go outside its pliable range. Since the system was not designed to operate outside of its architecture, there is a risk that the disturbance will not be survivable if the new, architecture non-conforming, instance is not viable. In this situation, the system needs to either return to its original architecture through repair or replacement, or adapt to a new, viable system architecture. In other cases, disturbances may not break a system architecture, but instead change the context in such a way that the system will not be survivable unless it adapts to a new, viable system architecture (Table 9-4).

TABLE 9-4: RECOMMENDED ACTIONS FOR DEALING WITH DISTURBANCES

Effect of Disturbance	Recommended Action
Changes a system's parameters that violate the pliability of SA	Repair system to original SA, validate new SA, or change to an alternative, viable SA
Changes context so current SA not producing value	Change SA
Change context so current system not producing value	Change system instance

9.2.3.3 Increasing Agility

By pre-validating reachable instances during architecting, the amount of approval necessary for changes after design should decrease. This reduces the "red tape" and allows systems leverage their pliability, allowing them to respond quicker to context shifts, increasing the ability for a system to change quickly (i.e. increasing agility (McGaughey, 1999) in response to changes in context).

9.3 Pliability Strategies for Enabling Viability

Using the concept of pliability, several new strategies for viability can be developed. These strategies attempt to reduce the risk associated with making a large transition., by being able to transition into viable instances that are not necessarily the instances that the system is intending to transition to. Using the legend shown in Figure 9-4, the following pliability strategies are defined:

Strategy of Reversion: Being able to revert back to an instance within the pliable set, in case new system does not provide acceptable value (Figure 9-5).

Strategy of Stable Intermediate Instances: Transitioning to a less-risky instance within the pliable set, before attempting to transition to a higher-risk instance, in case the system never makes the full transition (Figure 9-6).

Strategy of Contingency. Transitioning to an instance within the pliable set that is intermediate between what the system was, and what it was trying to be, in case the new system does not provide acceptable value (Figure 9-7).

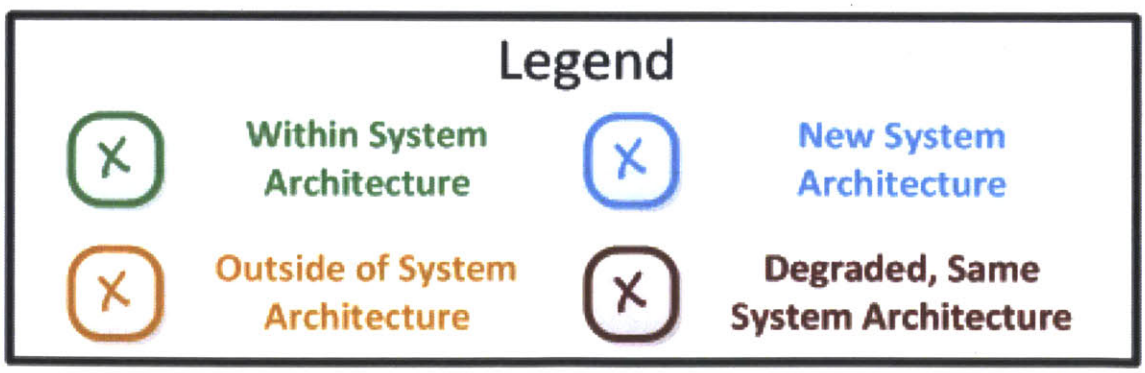

FIGURE 9-4: LEGEND USED FOR FIGURES 9-5, 9-6 AND 9-7

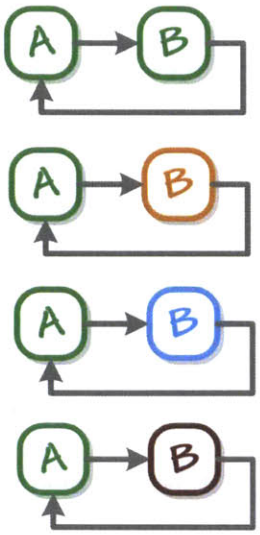

FIGURE 9-5: STRATEGY OF REVERSION

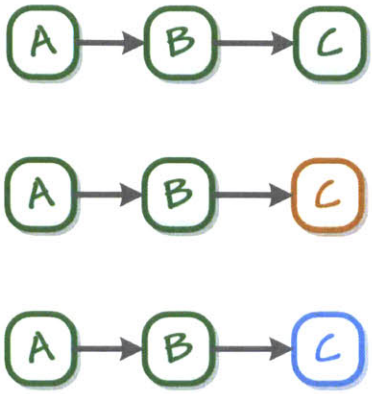

FIGURE 9-6: STABLE INTERMEDIATE INSTANCES

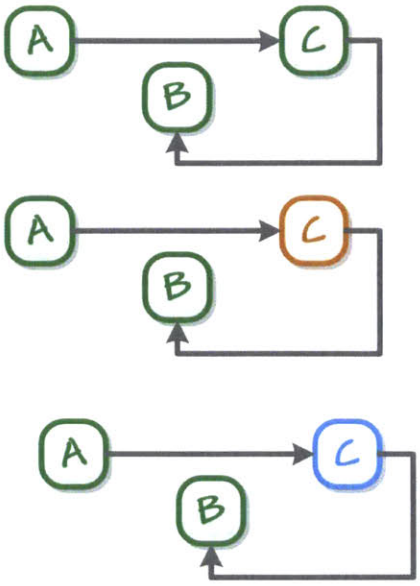

FIGURE 9-7: STRATEGY OF CONTINGENCY

9.4 PLIABILITY AND OTHER CHANGE-RELATED "ILITIES"

At first glance, pliability may seem to be the same as some other "ilities" already used such as "flexibility", "modifiability" or "adaptability". The following is a description of how pliability is related to, but distinct from, the other "ilities" as defined in Table 8-1.

Value robustness. Pliability is only concerned with changes in the system, and whether such changes are allowed under the SA. Implicit in the specification of "allowable under the SA" is the concept of value robustness. Pliable systems that undergo an allowable transition will retain value robustness for contexts that were considered in the design of the SA. Pliability does not guarantee value robustness in contexts that were not considered in the design of the SA, regardless of whether the system transition was allowable or not.

Modifiability. Pliability requires either modifiability or scalability, i.e. the parameter can change in some way. However, a system may be modifiable in a parameter beyond what is included in the pliable range.

Changeability. A system must be changeable to be pliable, but does not have to be pliable to be changeable. Pliability requires that the transition to a new system is possible (i.e. the system is changeable), but also that the new system is part of the same system architecture (not a requirement for changeability). In this way, changeability is more general than pliability.

Adaptability. Pliable systems need to be changeable, but they do not need to be changeable by an internal change agent. A system that is changeable only by an external agent can also be pliable.

Flexibility. Pliable systems need to be changeable, but they do not need to be changeable by an external change agent (an internal change agent will do), nor does the agent need intent. Pliable systems can transition to other systems by accident (e.g. as the result of a disturbance). In fact, one of the main goals of having a pliable system is that if a disturbance does cause an involuntary transition, the transition will be more likely to result in a system still within the system architecture.

Evolvability. A system that transitions to another instance within its SA is demonstrating pliability not evolvability. Evolvability requires changes in a system with inheritance during "redesign." This may correspond to changes in an inherited system architecture.

Agility. Pliability does not explicitly require timely changes, just as long as the transition "is possible". If stakeholders are not concerned with timeliness, then timeliness is not a factor in the pliability of the system. However, pliability may enable agility by reducing "red tape" for change.

Scalability. Pliability requires either modifiability or scalability, i.e. the parameter can change in some way. However, a system may be scalable in a parameter beyond what is included in the pliable range.

Versatility. Pliability allows change in Form and/or CONOPs. A system may be versatile and pliable if it changes CONOPs (only), within the pliable range, to satisfy diverse needs.

Reconfigurability. Reconfigurability is a specific change in CONOPS (one that changes the relationships between the Operational Elements). This change may or may not be in the pliable range, hence a system may be pliable and reconfigurable, pliable only (if the allowable changes do not include CONOPs changes concerning OE relationships) or reconfigurable only (if the CONOPs changes are not part of the pliable range).

Extensibility. Assuming that "new features" were features not included in the original design (i.e. SA), then extensibility and pliability are independent properties. For example, a computer can be both extensible and pliable. Performing an authorized upgrade from 2GB to 4GB RAM is pliability, if 4GB is part of the system architecture for that computer. Adding third-party peripherals or software is extensibility. Like any other change, adding features to a system may violate the SA, if it causes a parameter to change outside its pliable range (e.g. "power supply

needed"). Architects should try to safeguard the parameters of their SA when designing for extensibility.

Robustness. A pliable system may be robust, if it does not need to transition to a new instance to provide value under a new context. Pliability provides the system with the option to change, but does not require it to.

Modularity. Pliability is not concerned with the ability of a system to be composed of modules, although, modularity will likely make a system more changeable, thus facilitating pliability should the system architects wish to make those parameters pliable.

Interoperability. Not directly related to pliability, but a SA that supports interoperability will likely require a larger pliable set than a similar SA that doesn't, particularly if the system is taking a defensive posture (proposed Type I survivability design principle). This is because the range of what the system may need to accommodate will likely be larger than what would be needed if the interfaces were custom.

Survivability. Systems that are more pliable will be more survivable, due to the fact that there are validated system changes available should stakeholders need to change in response to a disturbance (Type II survivability). Systems may also be more survivable due to an increase in the likelihood that an involuntary change brought about by a disturbance, will still transition into one of the validated instances of the SA (Type II survivability).

9.5 SUMMARY

This chapter introduced the concept of pliability and defined it relative to a system architecture than includes both a form and a CONOPs. Not all change is valuable, and successful systems will be able to avoid, mitigate and recover from harmful changes, and implement beneficial ones in a timely and cost-efficient manner. Incorporating change into engineering design is something that many researchers and practitioners realize is necessary; however, pliability distinguishes systems from their architectures, and introduces the concept that the architecture should be validated so that it has multiple valid instances to which a system may transition, should the need arise. In this way, pliability increases survivability, value robustness, and possibly agility, by requiring architects to consider disturbances, context changes and changes that might not have been considered, increases the safe options available to a system in the event that a change occurs (intentional or not), and decreases the red tape involved in implementing intended changes.

10 CONCLUSIONS

Engineered systems are designed to deliver some benefit to stakeholders. When cost and other factors are considered, the net benefit, or value, that engineered systems deliver to stakeholders has to be acceptable, otherwise the system will be unsuccessful. System architects and stakeholders do not want their systems to fail, so they need to ensure that their engineered systems have the quality (or qualities) necessary to ensure that they will continue to provide acceptable value to their stakeholders. If nothing changes, then an engineered system that was designed to provide value for its stakeholders under a particular context or set of contexts, will continue to do so, and does not need to have any special qualities, other than being feasible. However, we live in a world where change is inevitable, and engineered systems need to be able to manage that change effectively, so that they can continue to provide value to their stakeholders for long periods and possibly numerous contexts. As technology improves, particularly communication technology, systems are being interconnected into larger, more complex, systems of systems with long, sometimes infinite lifespans. These systems of systems often operate in diverse contexts, in which both temporary and permanent perturbations exist that threaten the value delivery of the SoS. Furthermore, the systems of systems themselves are often dynamic, and can change their form voluntarily or involuntarily in response to contextual variability or stakeholder whims. Therefore, more often than not, system architects need to design systems that are more than just feasible, but able to provide value for their stakeholders over their expected life era, which may or may not involve some degree of change.

10.1 VIABILITY DEFINED FOR ENGINEERED SYSTEMS

Researchers have defined viability for biological systems and organizations, but not for systems that are engineered to provide value to stakeholders. To recognize that engineered systems may need to be more than just feasible, the concept of viability for engineered systems was defined in Chapter 4 as *the likelihood that the engineered system will provide acceptable value to its stakeholders, over its life era.* This is distinct from other previously defined concepts of value sustainment since it accepts that there is some uncertainty as to whether the system actually possesses the desired property or not. Although viability is a likelihood and certain systems may be more viable than others, a system itself is *viable* if it is expected to provide acceptable value to its stakeholders, over its life era, and not viable otherwise. This definition gives viability some important properties that system architects and stakeholders need to understand in order for them to apply the concept properly:

Viability is a Prediction. Viability is a *likelihood* of a system being able to provide acceptable value during its life era, but it is not a guarantee. If the life era of the system is different from the expected life era at the time viability was assessed, then viable systems may fail, and unviable systems may succeed. Since viability is a prediction, it involves risk. Assessing viability involves predicting what the system will encounter during its life era, what the system will be expected to do, and in the case of complex systems (particularly systems of systems), what the system will actually do.

Viability is Subjective. A system may be viable to one stakeholder and not to another, based on differences in the stakeholder preferences, and/or differences in the stakeholder's risk tolerance. This was shown in Section 4.3, where Designs II was viable to one stakeholder but not to others, based on lower preference thresholds.

Viability is not Invulnerability. A system may be viable, even if perturbations exist that may cause it to fail, as long as the chance of encountering those perturbations and/or the chance of not surviving them is low enough that the stakeholders do not expect them to be problematic.

Viability is Dynamic. Any engineered system should be viable throughout its life era, however it may become unviable at any time, as the system, context of expectations of the stakeholders change.

Viability is Relative. Some systems will be more likely to provide acceptable value to stakeholders than other systems, and thus some systems will be more viable as well. For example, consider a bomber that will fly missions over hostile territory. Although it is supposed to be accompanied by fighter jets, the bomber may find itself alone once in a while and needs to survive an encounter with hostile enemy aircraft. The stakeholders of the bomber do not expect that the bomber will be able to survive solo encounters 100% of the time, but they do expect it to be able to survive at least 90% of the time to be viable. Bomber design A is considered viable, because according to simulation, it can survive 94% of encounters with hostile enemy aircraft. However, bomber design B has additional margin on the wings, and as such, can survive 99% of encounters with hostile enemy aircraft. Both designs are viable because they exceed the thresholds set by the stakeholder, but design B is more viable since it is more likely to provide value to the stakeholders than design A.

Viability is not Existence. As discussed in Chapter 4, the ability to exist is mentioned in definitions for viability in both the life sciences and cybernetic management literature. A system needs to exist in order to be viable, but it does not need to be viable in order to exist. A system that cannot last the coming winter, for example, may be expected to die, even though it exists and perhaps is even providing acceptable value to stakeholders currently. Similarly, a

viable system can suddenly become unviable due to a change in stakeholder preference, but that does not mean it suddenly stops existing. It may continue to exist for a while, either until a change is made so that the system can become viable again, or until the system is terminated.

Viability does not Require Autonomy. According to the viable systems model found in the cybernetics management literature (Beer, 1984), a viable organization needs to be able to be autonomous and retain its independence. This may be true for systems like organizations where the distinction between stakeholders and the system itself is not clear, but for engineered systems that have separate and distinct stakeholders, it is not necessary for the system itself to be autonomous to provide acceptable value to its stakeholders, and thus be viable. If fact, systems that are designed to be constituent systems in systems of systems should be viable, even if that viability is dependent on other systems.

10.1.1 Viability and Other "-ilities"

From this broad definition, it seems intuitive that all engineered systems should be viable, and thus if a system may experience endogenous or exogenous change, viability will require other qualities other than just feasibility, to maintain its value delivery to stakeholders. Researchers have identified several system qualities, known as "ilities", which help a system maintain value in spite of change. In Chapter 3, several system qualities such as "survivability", "robustness", "resilience", and "reliability" were discussed and it was shown that there is lack of consensus in the literature as to their exact definitions. This makes applying the concepts difficult, but each of these terms relative to viability was explored in Chapter 4.

Viability and Survivability/Robustness/Resilience. There is a lack of consensus in the literature as to the exact definitions of survivability, robustness and resilience. For example, resilience alone has at least 35 different definitions (Sheard & Mostashari, 2008). In general, however, all of these terms have been used to describe a systems ability to continue to provide value in spite of perturbations, i.e. be value sustainable (Beesemyer, 2012).

This research demonstrated that more often than not, perturbations do not occur in isolation and that each perturbation may have multiple causes and multiple effects, which are also perturbations themselves. In some cases, effects of a certain perturbation may become the causes (directly or indirectly) of the same perturbations, in a reinforcing loop (Figure 10-1). These loops are particularly dangerous and can cause cascading failures, like those seen in large SoS failures like the Northeast Blackout of 2003.

Certain perturbations are not survivable, and have been called terminal events. Any system may encounter a terminal event; in fact, it is those terminal events that define when a system stops delivering acceptable value to its stakeholders and ceases to exist. Therefore,

survivability to a particular perturbation, is really about avoiding the terminal events eventually caused by the perturbation, by mitigation and/or resilience. Therefore, viability and survivability are not the same, but rather it can be stated that for a system to continue to exist, it must avoid the perturbations that it cannot survive, and survive the perturbations that cannot avoid.

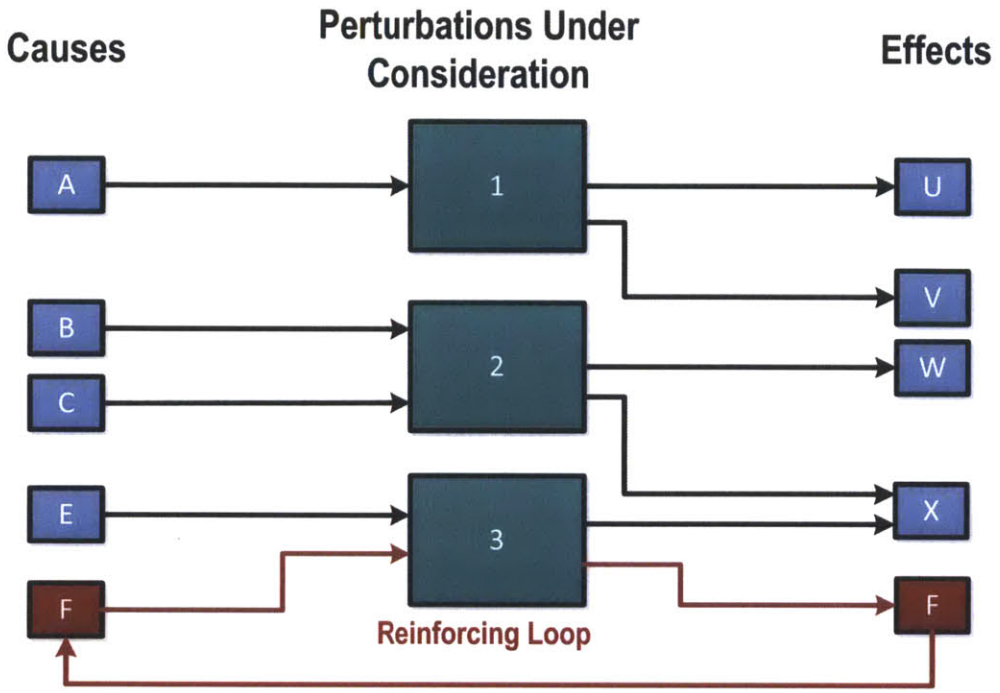

FIGURE 10-1: MULTIPLE CAUSES AND EFFECTS OF PERTURBATIONS, INCLUDING FEEDBACK LOOPS

Viability and Reliability. Reliability is ability of a system to continue to provide the outputs it was designed to provide. Thus, systems have to be reliable in order to be viable, unless they are resilient to involuntary changes.

Viability and Changeability. The viable system model also requires that systems have the ability to change, known as changeability, but it is not necessary for some engineered systems to have such an ability if the context and stakeholder needs remain fixed. A system that can only be viable if the context remains fixed is defined as fragile and fragile systems do not need to be changeable, accordingly. Of course fragile systems can be changeable, but it is unnecessary.

10.2 Strategies for Viability

A system that is expected to encounter perturbations will have to survive them in order for that system to be viable. Thus, existing strategies for survivability will also be strategies for viability,

in the cases where those perturbations can be expected in the system's life era. The technical literature has identified dozens of design principles for survivability, which include strategies for avoiding, mitigating and recovering from perturbations. Unfortunately, researchers have developed their own lists of design principles for survivability that have very little overlap with each other. It was shown that these design principles are not invariant, and are application dependent in many circumstances. Worse, it was also shown that implementing some of these principles may actually decrease survivability to other perturbations. Care must be taken when applying some of these existing design "principles", but in those situations, the "principles" are really just guidelines to consider and not rules that should always be followed. For these reasons, the existing survivability design principles were re-classified into two types of strategies for enabling system viability: patterns and heuristics.

10.2.1 Design Patterns for Enabling Viability in Systems

The first type of survivability strategies identified in the literature are solutions to certain types of perturbations. In Chapter 6, these solutions were expanded into design patterns similar to those introduced by Alexander (1977), in the field of architecture, but used in software engineering and recently in systems engineering. Design patterns specify a general solution to a typical problem, in such a way that it can be applied to many different systems in diverse contexts. Design patterns also cross reference other design patterns, and denote drawbacks associated with the solution, to help system architects avoid conflicts that may arise when multiple perturbations need to be addressed. Design patterns were divided into structural patterns, which deal with improvement to the system's form to enable viability, and operational design patterns, which attempt to improve the operation of the system to enable viability. Dividing the design patterns into these two types helps specific decision makers apply the concepts. For example, operators do not design the system, so they often not have much influence over its form. However, they do have some say in how the system operates, and thus would likely focus on operational design patterns as opposed to structural ones.

10.2.2 Design Heuristics

The second type of survivability design principles found in the technical literature, were not solutions to specific perturbations, but guidelines that may enable survivability in general. These survivability design principles were adapted into design heuristics for viability. Implementations of design heuristics often do not result in verifiable differences that can be quantifiably demonstrated like the implementations of many design patterns can. For example, a system analyst could show that implementing the design pattern of redundancy for a particular system may reduce the chance of complete system failure by a certain amount. This type of analysis makes the value of implementing certain design patterns easier to assess. Most

design heuristics, however, are not as easy to assess because they attempt to increase survivability by following good design and operational practices, as opposed to implementing solutions to known problems. An example of a heuristic is complexity avoidance, where the design is simplified as much as possible, (such as using the Rule of Simplicity in the Unix Operating System) to avoid hidden interactions which may either cause perturbations, or make responding to perturbations more difficult. The value of design heuristics may intrinsic and difficult to quantify in the design stage, but helps systems deal with unknown unknowns.

A summary of the design patterns and heuristics for viability is given in Table 10-1.

TABLE 10-1: DESIGN PATTERNS AND HEURISTICS FOR VIABILITY

Viability Strategy	Type		Survivability Design Principle(s)
Evasion	Design Pattern	Operational	Mobility (Richards, 2009)
Camouflage	Design Pattern	Structural	Concealment (Richards, 2009)
Skulking	Design Pattern	Operational	Concealment (Richards, 2009)
Miniaturization	Design Pattern	Structural	Concealment (Richards, 2009)
Containment	Design Pattern	Structural	Containment (Richards, 2009), Loose Coupling (Jackson, 2012)
Separation	Design Pattern	Structural	Containment (Richards, 2009), Loose Coupling, Localized Capacity (Jackson, 2012), Distribution (Richards, 2009)
Patrols	Design Pattern	Operational	Deterrence (Richards, 2009)
Maintenance	Design Pattern	Structural	Drift Correction (Jackson, 2012), Repair, Replace (Richards, 2009)
Fail-Safe	Design Pattern	Operational	Fail-safe (Richards, 2009)
Armor	Design Pattern	Operational	Hardness (Jackson, 2012; Richards, 2009)
Physical Redundancy	Design Pattern	Operational	Redundancy (Richards, 2009), Physical Redundancy (Jackson, 2012)
Heterogeneity	Design Pattern	Structural	Heterogeneity (Richards, 2009), Functional Redundancy (Jackson, 2012)
Margin	Design Pattern	Operational	Margin (Jackson, 2012; Richards, 2009)

Viability Strategy	Type		Survivability Design Principle(s)
Human-in-the-Loop	Design Pattern	Operational	Human-in-the-Loop (Jackson, 2012)
Failure Mode Reduction	Design Heuristic		Failure Mode Reduction (Richards, et al., 2009)
Layered Defense	Design Heuristic		Layered Defense (Jackson, 2012)
Avoid Human Error	Design Heuristic		Avoid Human Error (Jackson, 2012)
Complexity Avoidance	Design Heuristic		Complexity Avoidance (Madni & Jackson, 2009)
Independent Review	Design Heuristic		Independent Review (Jackson, 2012)

10.3 Applying Viability Strategies at the SoS-Level

Since systems of systems are also a class of systems, then existing strategies for viability can also be applied to systems as systems as well. However, there are several important caveats which this dissertation highlighted, that need to be taken into consideration when considering viability at the SoS-level.

Viability to the System vs. Viability to the SoS. Viability is defined relative to needs of a particular set of stakeholders. This means that in a SoS, a constituent system may be viable to its own stakeholders, but not to the SoS stakeholders, or vice versa. As discussed in Chapter 6, this has important consequences for decision makers, since changes to a system may affect the viability of its constituent systems, or to the SoS to which it belongs, in unexpected ways. This means that a change to increase the viability of a SoS, for example, may in fact backfire in the long run, because that change unexpectedly decreased the viability of a particular constituent system, which caused it leave the SoS, resulting in an unexpected SoS perturbation.

Axiom of Component Viability: In order for a system to be viable, a necessary and sufficient set of its components must be viable with respect to the system and contexts under consideration.

Strategies at the System vs. SoS level. Strategies for enabling viability can be applied at the constituent system or SoS-level. Some strategies are more effective at constituent system-level, whereas other strategies are more effective at the SoS-level. The example of margin being applied at both levels of a maritime security system of systems, was shown to be more effective at the system level, mostly because the changes at the SoS-level did not take into consideration bottlenecks in the overall concept of operation, which nullified the benefits of the strategy.

Strategies as Points of Intervention. Since strategies have costs associated with them, and in some cases drawbacks as well, it is not possible to implement every type of viability strategy, and system architects and stakeholders need to make tradeoffs. The principle of viability is used to identify points of intervention in cause-effect chains, where viability strategies would have the most impact. Specifically, events that have many causes (i.e. are likely to be encountered) should be mitigated or recovered from, while events that have many effects (likely not to be survivable) are good candidates for strategies that focus on avoidance or prevention.

10.4 Pliability To Restrict and Facilitate Endogenous Change

Researchers have been promoting various system qualities that allow for, or facilitate endogenous change, such as flexibility, adaptability, extensibility, and modifiability. Through historical cases such as the London Ambulance and the emergency response systems assembled for Hurricane Katrina, as well as an example from the MarSec DES, Chapter 8 gave examples of endogenous changes that were made to make systems more viable, but instead made them not viable at all. While there has been a considerable amount of empirical research to show that systems that have the ability to change are more value sustainable than systems that do not, there has not been much discussion in the literature about the need for system qualities that deliberately restrict change that could have prevented such failures from occurring.

To restrict change, one must first define exactly what a system is, and what change explicitly means. In Chapter 9, a concept of system architecture is explained that describes how actual, implemented systems can be thought of as instances of system architectures. System architectures were defined in terms of form and a concept of operations, and instances of these architectures were referred to as designs. A new "-ility", pliability, was defined to be the ability of a system to change, without breaking its system architecture. The *pliable set*, was defined to be the set of all validated designs that can be transitioned to, within a system architecture.

Pliability both restricts and facilitates endogenous change, and enables viability as a result. It restricts change because decision makers will be less likely to approve transitions to unverified, and possibly unviable designs, when viable design transitions exist. It facilitates change, however, because in large complex systems, particularly systems of systems, where there are diverse stakeholders and many possible unintended interactions between the components, getting major changes approved can be a very lengthy process that may hinder a systems' ability to rapidly respond to changes in contexts or needs. By pre-validating transitions, decision makers can have confidence in the changes to be made and approve them quickly, thereby allowing systems to remain viable when rapid transitions are necessary. Large pliable sets also enable viability by increasing the chance that a perturbation that causes an involuntary change in the system, such as a maritime security SoS losing a UAV, will be within the pliability of the system.

10.5 New Strategies for Viability

Using historical case studies, such as the Northeast Blackout of 2003, and experimenting with the MarSec DES, some strategies for avoiding and/or surviving perturbations, particularly in systems of systems, were discovered when the systems.

Intensity Regulation. Components within a system should be able to regulate their output so as not to disturb the local contexts of other components within the overall system.

Diversion. The local context of systems should be protected by diverting perturbations away from critical components. Deflection is diversion of exogenous perturbations (such as lighting rods deflecting lighting strikes) and redirection is diversion of endogenous perturbations (such as flow control between routers inside a network).

Stockpiling. The local context is protected by perturbations of resource availability by storing reserves of critical resources and using them when necessary.

Commitment. The local context of systems are protected against perturbations caused by stakeholders not providing the level of service or resources expected according to the concept of operations. Commitment can typically be implemented through service level agreements, as an example.

Authentication. The identify of constituent systems is verified before access to critical system resources and functions are granted.

Vigilance. Entities within the system of system boundaries are monitored for violation of the concept of operations, even if they have been authenticated previously.

Least Privilege. Once the identity of a constituent system has been identified, it is granted the minimum authority necessary to perform it tasks within the concept of operations.

Pliability. The strategy of pliability to design for change and pre-validate viable instances at the design stage to a) facilitate stakeholder approval for voluntary transitions and b) increase the changes that involuntary transitions will be viable.

Reversion. Allow the possibility of reverting back to the original, viable design of a system, in case the new design is not viable.

Stable Intermediate Instances. If a significant transition is to be performed that requires the system to be unviable for a significant period of time, attempt to transition into several, intermediate viable designs along the way to the final design instead. That way, if the final design cannot be reached, the system will still be viable.

Contingency. If reversion is not possible, then ensure that the system can transition into another viable state, in case the intended becomes unviable.

10.6 Summary of Responses to Research Questions

Direct answers to the research questions posed in Section 1.2 are presented below:

RQ1: What does it mean for an engineered system to be viable? How does viability relate to existing "-ilities" already defined?

An engineered system is viable if it is likely to provide value for its stakeholders, during its life era. A system needs to be reliable to be viable, and survivable if perturbations are to be expected. A system does not have to be survivable to contextual perturbations, however, if its context remains static. Keeping the local context of a constituent system static, is the key to some of the new viability strategies introduced in this dissertation.

RQ2: How does viability of constituent systems relate to the viability of systems of systems?

Viability is relative to stakeholder expectations and risk tolerances. Since constituent systems can have different stakeholders than the system of systems it belongs to, then the viabilities can be different as well. This can have important consequences since actions and strategies that affect the viability of a system, can have the opposite effect on the viability of the SoS, and vice versa.

RQ3: Do existing system-level strategies for enabling systems to resist the harmful effects of contextual change apply to systems of systems as well? Are there additional strategies that enable viability in systems, particularly systems of systems?

A SoS is still a system, so existing viability strategies still apply to systems of systems as well. However, the same strategy applied at the constituent system level, can have different results than if it were applied at the SoS level, so care must be taken. Additional viability strategies other than the ones currently evident in the systems literature, were discovered, including some that were SoS-specific (such as redirection). Many of these strategies centered on the need to keep local contexts of constituent systems static, if they are fragile to certain contextual changes.

RQ4: Is changeability always a good thing? What can system architects do to enable positive change in complex systems, while restricting change that can be harmful?

The ability to change is usually a desirable trait for systems that may experience dynamic contexts and/or stakeholder expectations, but as systems become more complex, intentional changes can have undesirable side effects. System architects should design pliable systems, where pre-verified, allowable changes are defined so that decision makers can quickly approve such changes to their systems without worrying about making them unviable.

10.7 Contributions of the Research

The main contributions of this research relate to the responses to the four research questions posed in Section 1.2.

1. The concept of viability is defined for engineered systems and its importance is shown. When comparing viability to other similar "-ilities" such as survivability and robustness, a lack of consensus in the literature emerges. In Section 3.4, the literature in regards to these "-ilities" is clarified, and viability is uniquely placed with the other "-ilities" already defined.

2. The viability of constituent systems and viability of the systems of systems that they belong to are shown to be related, but not the same. Furthermore, a SoS may be viable even if its constituents systems are not, and vice versa. This has important consequence for stakeholders since decisions concerning the viability of systems need to be taken carefully, as they may affect other stakeholders in unexpected ways.

3. Existing strategies for viability are transformed into design patterns and heuristics, to make them more useable by researchers and practitioners. Application of existing viability strategies are applied to the SoS level and shown to be still applicable. Perturbations are shown to occur in causal chains and not in isolation, and as such using cause-effect mapping, several new viability strategies are proposed and validated empirically with historical examples.

4. Several historical examples highlight the danger of making well-intended changes to complex systems without really understanding the impact, As such, "pliability" is proposed as a new "-ility" and strategy which is shown to improve viability by both facilitating and restricting endogenous changes.

10.8 Applicability of the Contributions and Limitations

The contributions of this dissertation were developed and/or validated using historical cases and a simulation of a maritime security system of systems, as described in the research approach (Chapter 2). Although the system that was modeled quantitatively was a directed SoS, the historical cases used to develop and validate the concepts and strategies in this dissertation ranged from simple, traditional systems (e.g. CPUS in laptops throttling their output to reduce heat), to virtual systems of systems (e.g. the TCP protocol used to regulate flow between routers in the Internet). This suggests that while the viability strategies and concepts introduced in this dissertation are applicable to traditional systems and directed systems of systems, they may be generalizable and applicable other systems and systems of systems, however there are some limitations.

First, viability can only be determined if the system has clearly defined stakeholders, stakeholder preferences and a means of assessing value delivery. For simple engineered systems, the stakeholders and value delivery of the system are often clearly defined, so the concept of viability and its associated strategies can be used. For engineered systems of systems that are directed or acknowledged (see Section 3.2.1), the SoS typically has clearly defined objectives and a central management where value delivery (and thus viability) for the entire SoS can be assessed. However, for natural systems, or systems of systems that are collaborative or virtual in nature and that lack SoS objectives and management, then the concept of viability, as it is defined in this dissertation, may not apply.

Even for systems where the concept of viability does apply, certain viability strategies may be ineffective due to the particular characteristics of the system and/or its context. As already demonstrated in Section 6.3, viability strategies for a SoS can have different effects if they are

implemented at the system or SoS-level, and sometimes it is not possible or effective to implement them at all. For example, increasing the technology of the UAVs may be a system-level implementation of the principle of margin, but the ability to do so may not be possible due to trade restrictions, for example. Thus, the stakeholders may be restricted to increasing the number of UAVs (a SoS-level implementation) instead, which may or may not be as effective. Even if the system-level implementation of margin could be implemented, it may not make the system viable if it is not effective against a potential disturbance. For example, if an earthquake damages the command center that directs all the tasks to the UAVs, then the SoS may become unviable regardless of the number of UAVs. For this reason, care must be taken when choosing viability strategies to ensure that they are applicable to the system and context of interest, and effective against the disturbances that the system is expected to encounter over its lifetime.

10.9 Future Work

As with any research, the contributions of this research were constrained by the time and resources available. However, this research made several enhancements to the body of knowledge, and laid the foundation for future contributions in the area of system of systems viability. Over the course of the research, limitations and areas for further exploration and research were identified and are presented below:

10.9.1 Expanding the List of Strategies for Enabling Viability

This dissertation introduced viability, and several strategies for viability were derived from existing survivability strategies, and generated from historical cases and empirical research done with the MarSec DES. However, the viability design patterns and heuristics presented in this dissertation can be expanded through additional research into related strategies in other disciplines that have similar problems to system engineering. For example, the area of computer science has similar problems of hardware and software entities interacting with each other to provide some benefit to stakeholders, and the area of operating systems and computer security may provide additional strategies that can be generalized for engineered systems. Additionally, other systems of systems besides maritime security should be simulated and used as experimental research test-beds where existing strategies can be refined (such as whether they are more applicable at the system or SoS-level), and new strategies can be discovered.

10.9.2 Quantitative Tools and Metrics For Assessing Viability

Viability was qualitatively defined in Chapter 4, however it may be possible to define precise, quantitative tools and metrics that will help stakeholders and system architects assess viability,

so that they can compare different designs with tradespace exploration methods such as Multi-Attribute Tradespace Evaluation (MATE) (Ross, Hastings, Warmkessel, & Diller, 2004). Since viability has been defined as a likelihood, a simple start could be to say that the viability of a system can be represented by a real number between 0 and 1, where 0 is a system that absolutely will not provide value to its stakeholders for the life era considered, and 1 absolutely will. However, it may be misleading to assign a specific value to a concept that is not trivial to measure. Assessing the viability for a particular system may involve a considerable amount of uncertainty, including

- Assessing the probability of occurrence and impact of rare events.
- Quantifying subjective qualitative preferences, such as "fun to use".
- Aggregating (or not) a viability into a single metric that covers a number of different stakeholders preferences, as well as a number of different possible life eras.
- Assessing viability for systems that are difficult to model, such as those that have considerable amounts of human decision-making.
- Assessing viability for contexts that have unknown perturbations.

These and other issues need to be addressed for a viability metric to be useful across many complex engineered system, but it was beyond the scope of this research.

10.9.3 Exploring the Issue of Real vs. Perceived Context

One of the key properties that distinguishes a system of systems from a traditional system, is the managerial and operational independence of the constituent systems. This means that in many cases, the constituent systems have decision-making ability within the SoS. Even if the SoS is directed, i.e. constituent systems are under the orders of a central command, and the constituent systems have to follow a strict concept of operations, the constituent systems are still responsible for sensing and interpreting the context, and taking appropriate action. Ideally, the actions (and thus the attributes) the system takes depend solely on the system context, φ_X. In reality, the system takes action based on what the system perceives as context, φ'_X. However, the results of those actions depend upon the (real) system context φ_X, not the perceived context φ'_X. For example, suppose a MarSec (system A), falsely identifies a civilian ship in its AOI as a pirate. The MarSec decides to attack the ship with a combat UAV, resulting in the ship being sunk. The real context (φ_A) before the attack was that there was a civilian ship in the AOI which was misidentified as a pirate. The context as perceived by the MarSec (φ'_A), was that there was a ship identified as a pirate in the AOI. The action the MarSec took based on φ'_A, was to sink the pirate ship. However, the results of that action, was that a civilian ship was sunk. The difference between perceived and real context is a source of vulnerability to many systems, but also a source of survivability, and should be explored.

10.9.4 Pliability in Tradespace Studies

Tradespace studies are often used to help decision makers assess and compare various system designs in the conceptual stage of a system's lifecycle (Stump et al., 2004). Typically, designs are modeled and simulated and their overall utility (as determined by stakeholder/decision maker preferences) is plotted against cost. For a system that does not change, the utility and cost is constant for a particular context (Figure 10-2). However, if the system is pliable, such as that for a given context, there can be a range of utility/cost that a system can achieve or change to . Several interesting questions arise. Suppose in Figure 10-2, that system B differs from system A only in that an extra component is added (providing additional value at additional cost). Should the changeability of systems A and B be represented somehow in the tradespace (Ross and Hastings, 2006) and if so, how should their utilities/costs be aggregated so that they can be compared against one another?

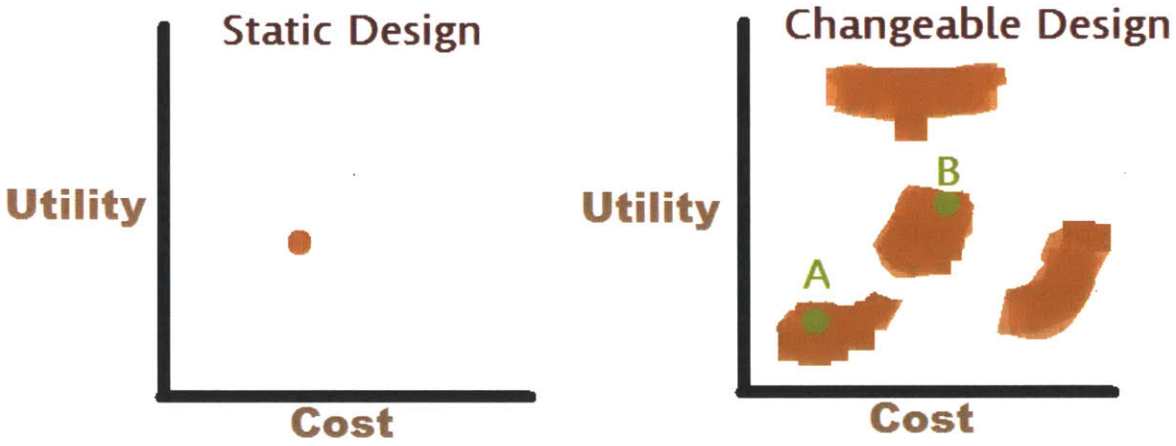

FIGURE 10-2: SAMPLE UTILITY-COST PLOTS OF A STATIC (L) SYSTEM AND A PLIABLE (R) SYSTEM

Changes are not always as discrete and obvious as adding components. Sometimes small changes in the way a system operates could have large cost and performance consequences. As an example, if a 911call center prioritizes calls based on location, it may have vastly different performance results that one whose mode of operation is to respond to calls in a first-in, first-out manner. Explicitly trading the modes of operation may result in multiple points in the tradespace for the same set of components. If the modes of operation are not traded, then system architects must recognize that if operators use the same system in a different way, it may appear to "move" in the tradespace. It is for this reason that system architects should explicitly consider an operations "envelope" around points to account for various modes of

operation, and future work should examine how to assess and utilize such an "envelope" when attempting to compare pliable systems in tradespace studies.

10.9.5 Using Systemic Models To Derive Viability Strategies

The cause-effect mapping used in Section 7.1 was useful for determining intervention points where decision makers can implement strategies that prevent terminal events from occurring by avoiding, mitigating and recovering from the perturbations that cause them. Like causal diagrams used in system dynamics (Sterman, 2001), the cause-effect mapping in Section 7.1 included non-linear relationships that include reinforcing loops. However, causal diagrams such as those used in this dissertation and cause-effect relationships that form the basis of many of the hazard analysis techniques such as Fault Tree Analysis (FTA) and Failure Mode and Event Chain Analysis (FMECA), tend to rely on knowledge gained from existing designs to determine what could go wrong and why (Ishimatsu et al., 2010). For system architects designing cutting-edge systems, this type of experience is not to be taken for granted. Newer approaches in system safety have looked at developing systemic models, such as the System-Theoretic Accident Model and Processes (STAMP) that look at accidents (system failures) as a dynamic control problem instead of a perturbation problem. According to STAMP, perturbations are the result of inadequate control that result from the lack of constraints in system design and operations (Leveson, 2004). Future work in viability should look at applying techniques similar to STAMP to determine how control strategies affect viability of complex systems in avoiding and surviving perturbations that have unknown causes and unknown effects.

10.10 Concluding Thoughts

As John F. Kennedy and many others have pointed out, change is inevitable. As existing systems are being interconnected into systems of systems, they are becoming more complex, they operate in multiple contexts, have diverse stakeholders, and exist for long periods of time. As such, system architects must consider how exogenous and endogenous change will affect their system and design accordingly.

"Things alter for the worse spontaneously, if they be not altered for the better designedly.

-Francis Bacon

GLOSSARY

TERM	DEFINITION
ADAPTABILITY	The ability of a system to be changed by a system-internal change agent with intent.
AGILITY	The ability of a system to change in a timely fashion.
CRITICAL COMPONENT	Any component, whose presence or operation is required for adequate value delivery to the stakeholders.
COMMON CAUSE	A perturbation that has many effects.
COMMON EFFECT	A perturbation that has many causes.
CONSTITUENT SYSTEM	A system that is a component of a larger system of systems.
CHANGEABILITY	The ability of a system to alter its form or operations, and consequently possibly its function, at an acceptable level of resources.
DESIGN DECISION MAKER	Someone who is able to make decisions about what the system is, or what the system does, at the design level.
DESIGN PATTERN	A solution to a design problem.
DECISION MAKER	Someone who is able to make decisions about what the system is, or what the system does, at the design, strategic or operational level.
DISRUPTION	An Instant (or near-Instant) perturbation.
DISTURBANCE	A perturbation of finite length, i.e. after some change has occurred, the original context resumes.
EPOCH SHIFT	A permanent change in context.
EXTENSIBILITY	The ability of a system to accommodate new features after design.
EVOLVABILITY	The ability of a design to be inherited and changed across generations (over time).
FLEXIBILITY	The ability of a system to be changed by a system-external change agent with intent.

Term	Definition
Interoperability	The ability of a system to effectively interact with other systems.
Modifiability	The ability of a system to change the current set of specified system parameters.
Modularity	The degree to which a system is composed of modules.
Operational Decision Maker	Someone who is able to make decisions about what the system is, or what the system does, at the operational level.
Perturbation	An imposed state change in a system's form, operations, or context which could jeopardize value delivery
Pliability	The ability of a system to change, without breaking its system architecture.
Reconfigurability	The ability of a system to change its configuration (component arrangement and links).
Robustness	The ability of a system to maintain its level and/or set of specified parameters in the context of changing system external and internal forces.
Scalability	The ability of a system to change the current level of a specified system parameter.
Spontaneous Event	An event that has no cause of relevance to the system.
Strategic Decision Maker	Someone who is able to make decisions about what the system is, or what the system does, at the strategic level.
Strategy	A general approach to a problerm.
Survivability	The ability of a system to minimize the impact of a finite duration disturbance on value.
System Architecture	A collection of components and an associated concept of operations (CONOPs), whose instances provide some value, within a particular context.
System of Systems	A system whose components are systems themselves, i.e. they can provide value on their own outside the system.
Traditional System	A system whose components are not able to provide value to stakeholders on their own.

Term	Definition
Utility	A dimensionless measure of usefulness to a stakeholder. Typically, utility is measured between 0 and 1, with 0 being useless and 1 being the most useful.
Value	The net benefit or utility (i.e. when cost is taken into consideration) that magee an engineered system provides to its stakeholders.
Value robustness	The ability of a system to maintain value delivery in spite of changes in needs or context.
Versatility	The ability of a system to satisfy diverse needs for the system without having to change form (measure of latent value).
Viability	The likelihood that an engineered system will provide acceptable value to its stakeholders, over its life era.

APPENDIX A: VIABILITY STRATEGIES

The strategies for avoiding/mitigating certain perturbations and enabling viability, as discussed in Chapter 6 and 7, are summarized below for reference.

PERTURBATIONS	STRATEGY	STRATEGY DESCRIPTION	HISTORICAL CASE	METHOD OF GENERATION
Cost Threshold Exceeded UAV Runs Out of Fuel	Stockpile	Store excess consumable resources to prevent disruptions of supply.	Strategic Petroleum Reserve (SPR) by the United States.	Derived from cause/effect mapping of MarSec SoS. Validated with hisotrical case study.
Rocket Attack	Deflection	Divert perturbations away from the system. Deflection is diversion of exogenous perturbations away from critical components.	Chaff used during WWII to deceive radar systems.	Derived from cause/effect mapping of MarSec SoS. Validated with hisotrical case study.
Stakeholder Withdrawal	Commitment	Ensure that critical entities do not withdraw their support to the system.	Service Level Agreement (SLA) such as Google API SLA.	Derived from cause/effect mapping of MarSec SoS. Validated with hisotrical case study.
Violation of Assignment Rules	Authenticaion	Verify entities before allowing them within system boundary.	Username and password on major commercial websites, ID badges at nuclear power plants	Derived from cause/effect mapping of MarSec SoS. Validated with hisotrical case study.
	Strategy Of Least Privilege	Components should only be limited in their influence of other components to only what is necessary.	User account privileges on computer networks such as the Sony PlayStation Network (PSN).	Derived from cause/effect mapping of MarSec SoS. Validated with hisotrical case study.

Perturbations	Strategy	Strategy Description	Historical Case	Method of Generation
UAV Not In Range	Vigilance	Monitor for both internal and external changes.	Multiple security checkpoints at an airport.	Derived from historical case study. Validated through MarSec SoS DES (Section 7.5.1.1)
Operator Overworked	Redirection	Divert perturbations away from critical components. Redirection is diversion of endogenous perturbations away from critical components.	Transfer Control Protocol (TCP) used in the Internet.	Derived from historical case study. Validated through MarSec SoS DES (Section 7.6.1.1).
Large Increase in Target Arrivals	Throttling	Reduce intensity of outputs to prevent perturbations to nearby contexts.	CPU downclocking to avoid excessive heat spreading to nearby components.	Derived from cause/effect mapping of MarSec SoS. Validated with hisotrical case study.

Appendix B: Viability Design Patterns

The following are a re-classification of existing survivability/resilience design principles into viability design patterns, which are re-usable solutions to common perturbations.

Name	Evasion
Type	Operational
Design Principle	Mobility (Richards, 2009)
Problem	System is within the physical sphere of influence of a hostile entity
Context	Hostile entity with ability to physically harm system if system is within sphere of influence
Solution	The system moves away from the hostile entity, reducing the impact of its influence.
Unintended Consequences	The system may move outside of the sphere of influence of friendly entities, making communication, coordination, resource sharing, etc. more difficult. May violate the concept of operations for a SoS
Example	Navy TACAMO E-6 strategic communications aircraft, which keeps moving to avoid detection.
Related Patterns	Separation

Name	Camouflage
Type	Structural
Design Principle	Concealment (Richards, 2009)
Problem	System can be detected by hostile entity
Context	Hostile entity with ability to visually detect entity, and ability to harm entity
Solution	The system reduces its visual signature by using colors that are not easily distinguished from the environment
Unintended Consequences	The system cannot be easily seen by friendly entities. This may delay or hinder collaboration and/or resource sharing, as well as possibly cause collisions
Example	Painting Abrams tanks colors that match terrain (i.e. tan, green, etc.)
Related Patterns	Skulking, Miniaturization

Name	Containment
Type	Structural
Design Principle	Containment (Richards, 2009), Loose Coupling (Jackson, 2012)
Problem	Perturbation spreads from component to component due to the physical influence the components have on each other
Context	Components that have physical influence on each other, and perturbations that are physically based
Solution	Physical barrier is placed between components to disrupt the physical influence they have on each other
Unintended Consequences	Reduced communication, coordination and cooperation between components (may conflict with inter-node impediment strategy)
Examples	The steel or reinforced concrete structures that houses most nuclear reactors
Related Patterns	Separation, Fail-safe

Name	Separation
Type	Structural
Design Principle	Containment (Richards, 2009), Loose Coupling, Localized Capacity (Jackson, 2012), Distribution (Richards, 2009)
Problem	Perturbation spreads from component to component due to the physical influence the components have on each other
Context	Components that have physical influence on each other Perturbation that is physically based
Solution	Components are located beyond their physical spheres of influence
Unintended Consequences	Reduced communication, coordination and cooperation between components (may conflict with inter-node impediment strategy)
Example	Blackhawk helicopter using quick disconnects and leak isolation valves in the flight control system
Related Patterns	Evasion, Containment, Fail-safe

Name	Patrols
Type	Operational
Related Design Principle	Deterrence (Richards, 2009)
Problem	Hostile entity wishes to perform an unpermitted act against or within system
Context	Hostile entity with opportunity within visual proximity of the system
Solution	Constituent systems with monitoring and possible effectors (e.g. weapons, etc.) purposely at boundary of system and environment, in view of potential hostile entities
Unintended Consequences	Conflicts with concealment strategy
Example	Security patrols around an airport
Related Patterns	Camouflage

Name	Maintenance
Type	Operational
Related Design Principle	Drift Correction (Jackson, 2012), Repair, Replace (Richards, 2009)
Problem	Components naturally degrade over time, resources are consumed and must be replaced
Context	Any complex system where resources are consumed and must be replaced and components degrade over time
Solution	Replace consumables to adequate levels, ensure that components are operating as expecting and repair/replace those that are not
Unintended Consequences	May need to make system inoperable while checks are being performed, even if nothing needs to be done
Example	Aircraft maintenance checks mandated by the FAA for commercial aircraft
Related Patterns	

Name	Fail-Safe
Type	Structural
Related Design Principle	Fail-safe (Richards, 2009)
Problem	The system may not have the time or resources necessary to implement an active solution to a perturbation and prevent its effects from propagating
Context	A perturbation whose effects can be interrupted through a passive solution
Solution	Leverage the physics of the failure to implement a passive solution that does not require resources or time to engage
Unintended Consequences	Automatic response may be unnecessary or even dangerous if context is not what the designers had in mind
Example	Dead-man's switch on trains
Related Patterns	Containment, Separation

Name	Armor
Type	Structural
Related Design Principle	Hardness (Jackson, 2012; Richards, 2009)
Problem	A physical attack on the system
Context	The system is within the physical influence of a physical perturbation (hostile attack, environment, etc)
Solution	Encase delicate materials within an enclosure that can withstand the physical perturbation
Unintended Consequences	Armor may interfere with communications, may add unnecessary size (reducing the strategy of miniaturization), may add unnecessary weight (making the strategy to mobility difficult to achieve)
Example	Composite Armor on m1 Abrams Battle Tanks
Related Patterns	Containment, Deflection

Name	Physical Redundancy
Type	Structural
Related Design Principle	Redundancy (Richards, 2009), Physical Redundancy (Jackson, 2012)
Problem	Failure of a component leaves system without capability
Context	A component provides a unique capability and failure of the component is possible
Solution	Duplicate component and allow for spare to be used in the event that the original fails or is otherwise unable to provide the capability
Unintended Consequences	May add unnecessary size (reducing the strategy of miniaturization), may add unnecessary weight (making the strategy to mobility difficult to achieve). The probability of at least one component failing actually increases, if failure is completely independent. This may cause additional maintenance actions, reducing the overall value delivery of the system
Example	Space Shuttle avionics systems (five identical general-purpose computers)
Related Patterns	Heterogeneity, Maintenance, Margin

Name	Heterogeneity
Type	Operational
Related Design Principle	Heterogeneity (Richards, 2009), Functional Redundancy (Jackson, 2012)
Problem	Capability is not being provided adequately.
Context	A necessary operation is not being completed, because there is perturbation that interferes with the standard operations of the system
Solution	The same capability can be achieved through a different operation
Unintended Consequences	May require additional components, meaning unnecessary size (reducing the strategy of miniaturization), and/or unnecessary weight (making the strategy to mobility difficult to achieve)
Example	Nuclear "triad" of bombers, ICBMs and nuclear submarines.
Related Patterns	Physical Redundancy, Margin

Name	Margin
Type	Operational
Related Design Principle	Margin (Jackson, 2012; Richards, 2009)
Problem	Capability is not being provided adequately
Context	Perturbation causes the output of system to be inadequate
Solution	Have system be capable of producing more output than necessary.
Unintended Consequences	May require additional components, meaning unnecessary size (reducing the strategy of miniaturization), and/or unnecessary weight (making the strategy to mobility difficult to achieve)
Example	Extra-long wings on an A-10 aircraft, which generates enough lift in the event that part of the wings are destroyed or damaged
Related Patterns	Heterogeneity, Physical Redundancy

Name	Human-in-the-Loop
Type	Operational
Related Design Principle	Human-in-the-Loop (Jackson, 2012)
Problem	Automation performs inappropriate action, or fails to perform proper action.
Context	Automation capable of performing an action independently
Solution	Require human input before critical actions are taken or ignored
Unintended Consequences	May delay critical actions
Example	Predator UAV requires human authorization for missile launch at target
Related Patterns	Heterogeneity, Physical Redundancy

Name	Storage
Type	Structural
Related Viability Strategy	Stockpiling
Problem	Supply of external resources diminishes, or cost is raised
Context	System cannot get enough external resources at a reasonable cost
Solution	Store excess resources when supply is plentiful/cost is low, use when needed
Unintended Consequences	May add require additional components, meaning unnecessary size (reducing the strategy of miniaturization), and/or unnecessary weight (making the strategy to mobility difficult to achieve)
Example	United States Strategic Petreleoum Reserve (SPR)
Related Patterns	

Name	Deflection
Type	Structural
Related Viability Strategy	Deflection
Problem	External perturbation disurpts critical component
Context	Critical component exposed to external perturbation
Solution	Divert external perturbation away from critical component
Unintended Consequences	May add require additional components, meaning unnecessary size (reducing the strategy of miniaturization), and/or unnecessary weight (making the strategy to mobility difficult to achieve)
Example	Lightning rod diverting lightning away from rest of building
Related Patterns	Armor, Redirection

Name	Redirection
Type	Operational
Related Viability Strategy	Diversion
Problem	Internal consitutuent system has exceded capacity
Context	Constitunet systems send tasks / information / materials to each othe (flow)
Solution	Constinetunet systsems send flow to other consituent sytems that have note xceedec capcity
Unintended Consequences	Delays in flow being sent, addiiontal cost to handle, re-route flow
Example	Flow control on TCP/IP protocol
Related Patterns	Deflection, Throttling

Name	Commitment
Type	Operational
Related Viability Strategy	Commitment
Problem	Staekholders not providing critical services
Context	SoS relies on constientu system providing critical services
Solution	Make consituten systesm commit to providing services buy making it unattractive not to
Unintended Consequences	Commitment may be two-way street and may lock up rest of SoS in an unviable state, if context, needs or rest of system changes
Example	Google Service Level Agreement (SLA)
Related Patterns	

Name	Vigilance
Type	Operational
Related Viability Strategy	Vigilance
Problem	Entity inside of system boundary performs unauthorized action
Context	Entity has already entered system and has ability to influence the system from within
Solution	Continually monitor the actions of entities interla to the system to ensure they do not violated CONOPs
Unintended Consequences	Additional cost, delays in oeprations if additional monitoring requires addiitonal authenticion, disurption in CONOPs if friendly entities fails authentication (for forgetting ID, etc)
Example	Requiring user names and passwords to be re-entered before crtiical actions, even after initial authenticiation
Related Patterns	Authentication

Name	Authentication
Type	Operational
Related Viability Strategy	Strategy of Least Privledge
Problem	Entities must be verified as being part of system before privelgse are granted
Context	SoS allows entities to interact with system and access / control critical information / services
Solution	Verify the identity of the entity wishing to gain access to the system
Unintended Consequences	Additional cost, delays in operation, disruption in CONOPs if friendly entities fails authentication (for forgetting ID, etc)
Example	Username and password being required before logging into online bank services
Related Patterns	Vigilance

Name	Throttling
Type	Operational
Related Viability Strategy	Intensity Regulation
Problem	Unintended interations between consiteunt systesms
Context	Quantiy of output from one consituen system may disrupt the context of other consiteunt systesms
Solution	Reduce the quantity ofoutput from consituent system (throttle)
Unintended Consequences	Reduced output of system
Example	CPUs downclocking to reduce overall heat in system, routers delaying sending packets to other overworked routers
Related Patterns	Redirection

Name	Authentication
Type	Operational
Related Viability Strategy	Strategy of Least Privledge
Problem	Entities must be verified as being part of system before privelgse are granted
Context	SoS allows entities to interact with system and access / control critical information / services
Solution	Verify the identity of the entity wishing to gain access to the system
Unintended Consequences	Additional cost, delays in operation, disruption in CONOPs if friendly entities fails authentication (for forgetting ID, etc)
Example	Username and password being required before logging into online bank services.
Related Patterns	Vigilance

Appendix C: Viability Design Heuristics

The following are a re-classification of existing survivability/resilience strategies into viability heuristics or "rules-of-thumb" that enable viability in systems (or systems of systems).

Heuristic name	Details
Failure Mode Reduction (Richards, et al., 2009)	**Description:** Elimination of system hazards through intrinsic design. **Example:** Replacing Teflon insulation with stainless steel for the Apollo 13 mission.
Layered Defense (Jackson, 2012)	**Description:** Use more than one independent viability strategy to address a single perturbation. **Example:** Most servers use a firewall as well as usernames and passwords to protect user accounts.
Avoid Human Error (Jackson, 2012)	**Description:** Strategies should be employed to reduce human error, particularly for humans involved in critical decision making. **Example:** Confirmation dialog on most graphical user interfaces.
Complexity Avoidance (Madni & Jackson, 2009)	**Description:** The system should not be more complex than necessary. **Example:** The Rule of Simplicity used in the Unix Operating System (Raymond, 2004) .
Independent Review (Jackson, 2012)	**Description:** Have experts who are not stakeholders or the system architects to review the design for viability. **Example:** The independent Standing Review Board assessment that is part of NASA's engineering design life cycle for space flight projects (NASA, 2011).
Strategy of Least Privileges	**Description:** Grant entities the minimum privileges necessary to satisfy the system's CONOPs. **Example:** User accounts that do not have administrator access in Windows systems.

Heuristic name	Details
Pliability	**Description:** Ensure that there are multiple, viable instances that a system may transition to in case in needs to. **Example:** Convertible in automobile can put top on in case of rain.
Reversion	**Description:** Before a transition, ensure that revision to the original instance is possible. **Example:** If upgrading the UAVs in a MarSec, do not get rid of the old UAVs as they may be needed if new UAVs do not work as intended.
Stable Intermediate Instances	**Description:** Before making a large transition, make smaller transitions to viable intermediate instances first, in case larger transition becomes impossible to implement. **Example:** If upgrading an entire network, make small transitions first, such as upgrading each client computer one at a time, and verify before proceeding.
Contingency	**Description:** When reversion is not possible, make sure that there is an alternate design availabel to transition to, incase intended design becomes unviable or impossible to implement. **Example:** When replacing radar towers with UAVs in a MarSec, make sure that manned patrol aircraft can perform same job in case UAVs cannot be implemented, or are not as viable as thought.

APPENDIX D: SUMMARY OF SIMULATION DATA

A summary of the quantitative data generated by the MarSec SoS DES and used in this dissertation is presented here.

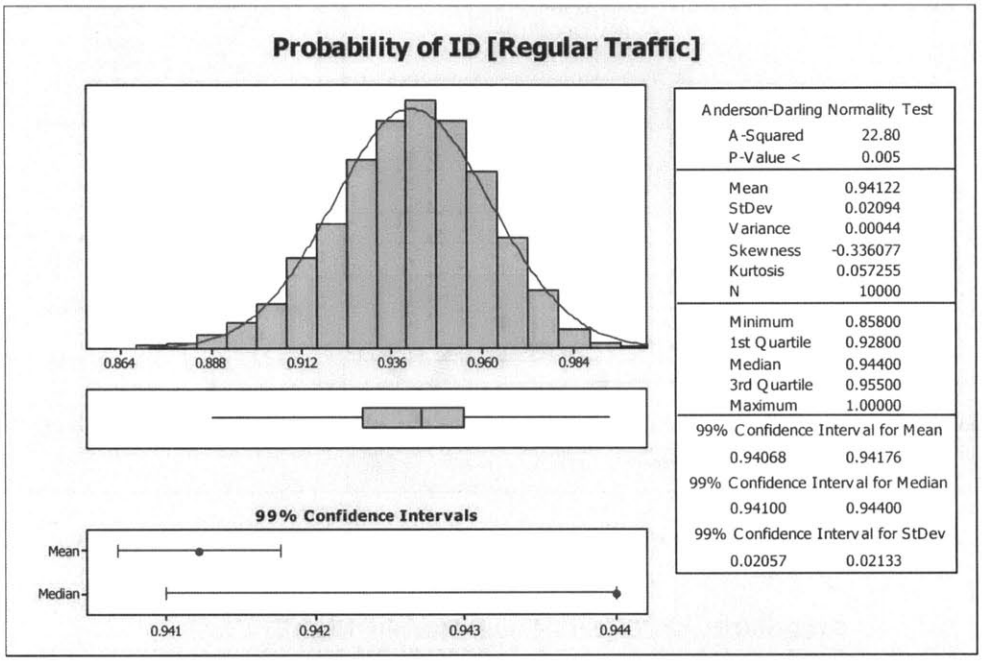

FIGURE D-1: PROBABILITY OF ID WITH BASELINE SOS EXPERIENCING REGULAR TRAFFIC

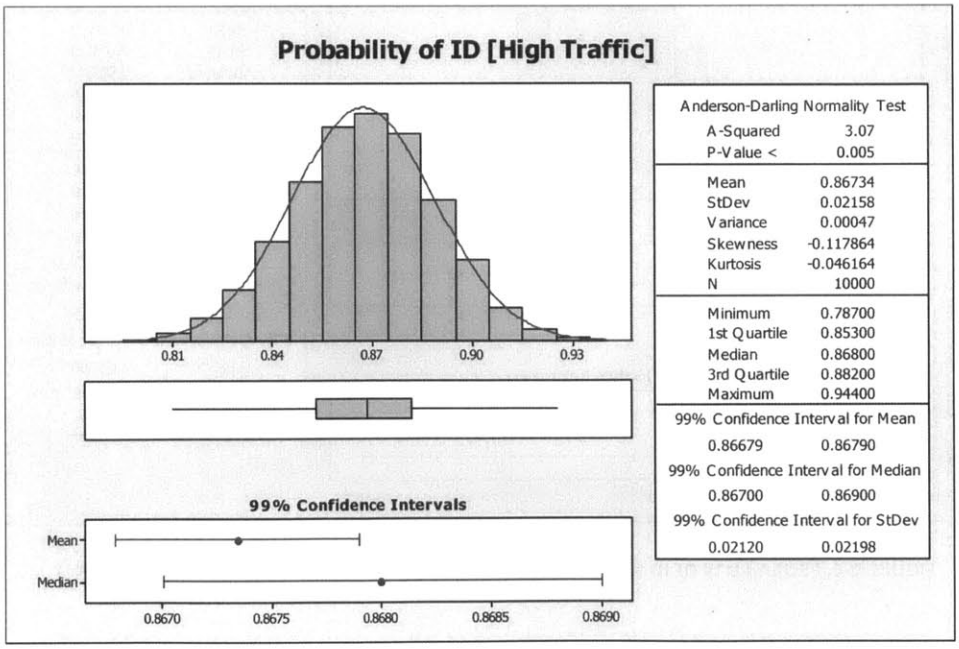

FIGURE D-2: PROBABILITY OF ID WITH BASELINE SOS EXPERIENCING HEAVY TRAFFIC

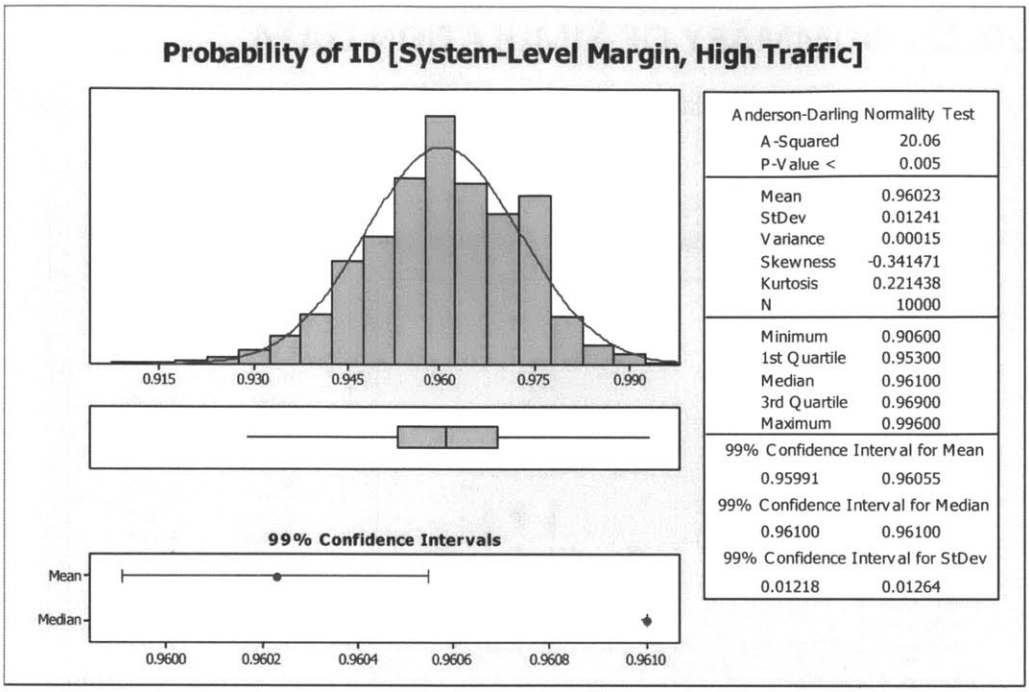

FIGURE D-3: PROBABILITY OF ID WITH SYSTEM-LEVEL MARGIN STRATEGY (UNDER REGULAR TRAFFIC)

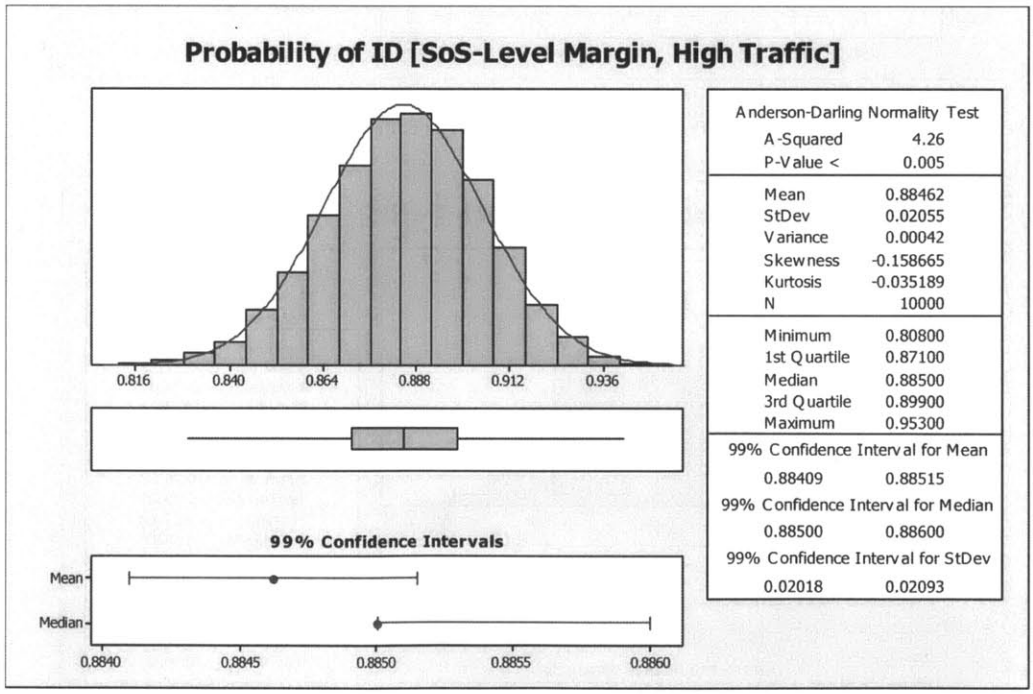

FIGURE D-4: PROBABILITY OF ID WITH SYSTEM-LEVEL MARGIN STRATEGY (UNDER HEAVY TRAFFIC)

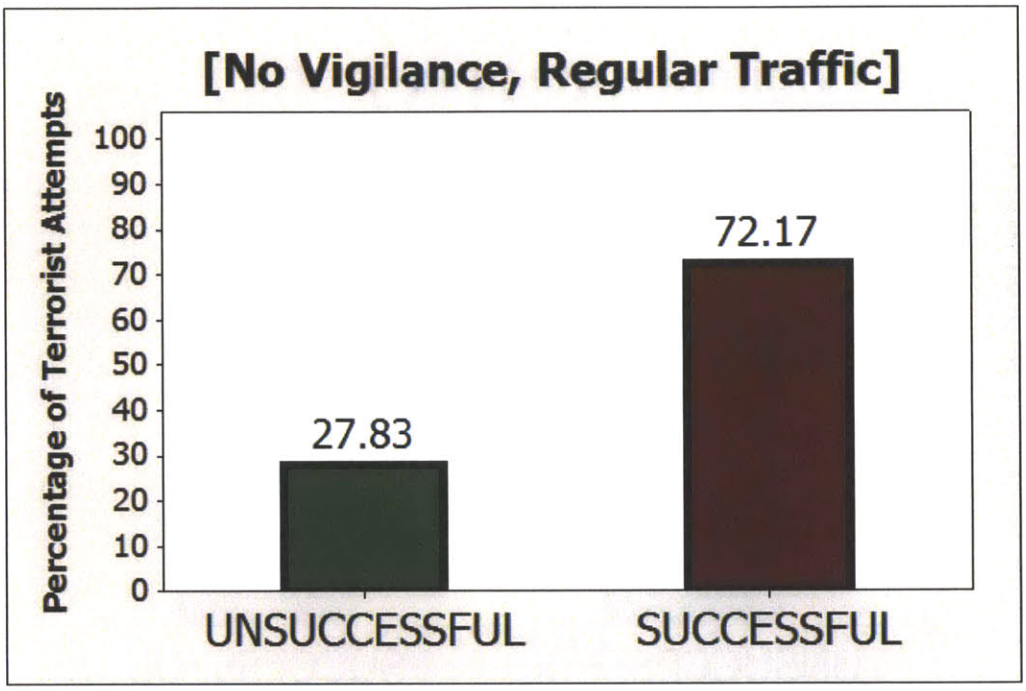

FIGURE D-5: CHANCE OF SUCCESSFUL TERRORIST ATTACK ON BASELINE SOS (GVIEN ATTEMP UNDER REGULAR TRAFFIC CONDITIONS)

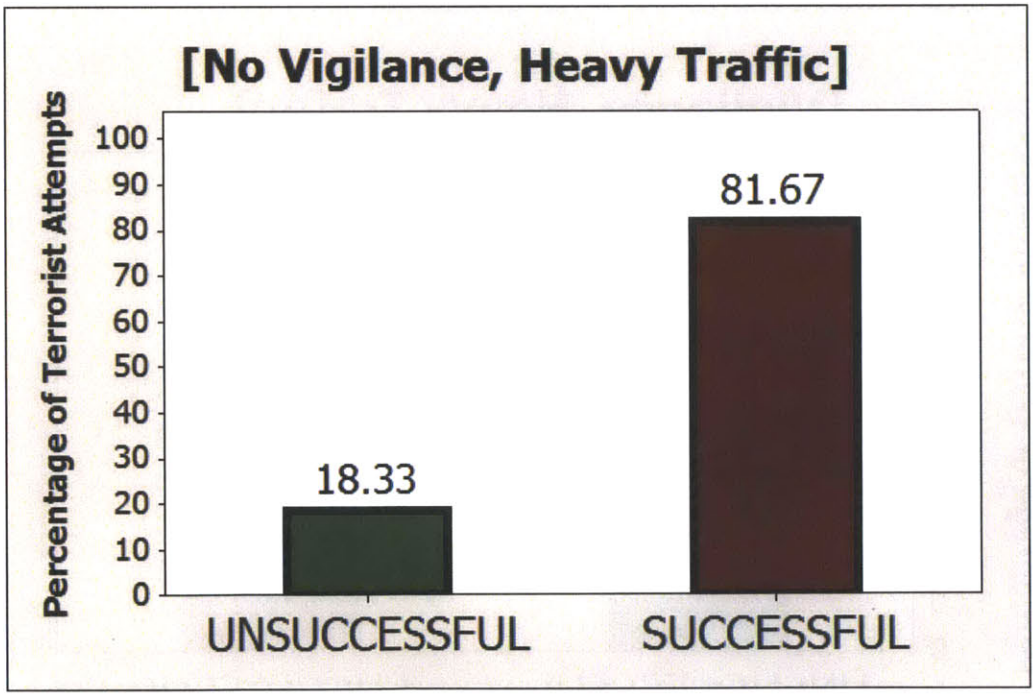

FIGURE D-6: CHANCE OF SUCCESSFUL TERRORIST ATTACK ON BASELINE SOS (GIVEN ATTEMPT UNDER HEAVY TRAFFIC CONDITIONS)

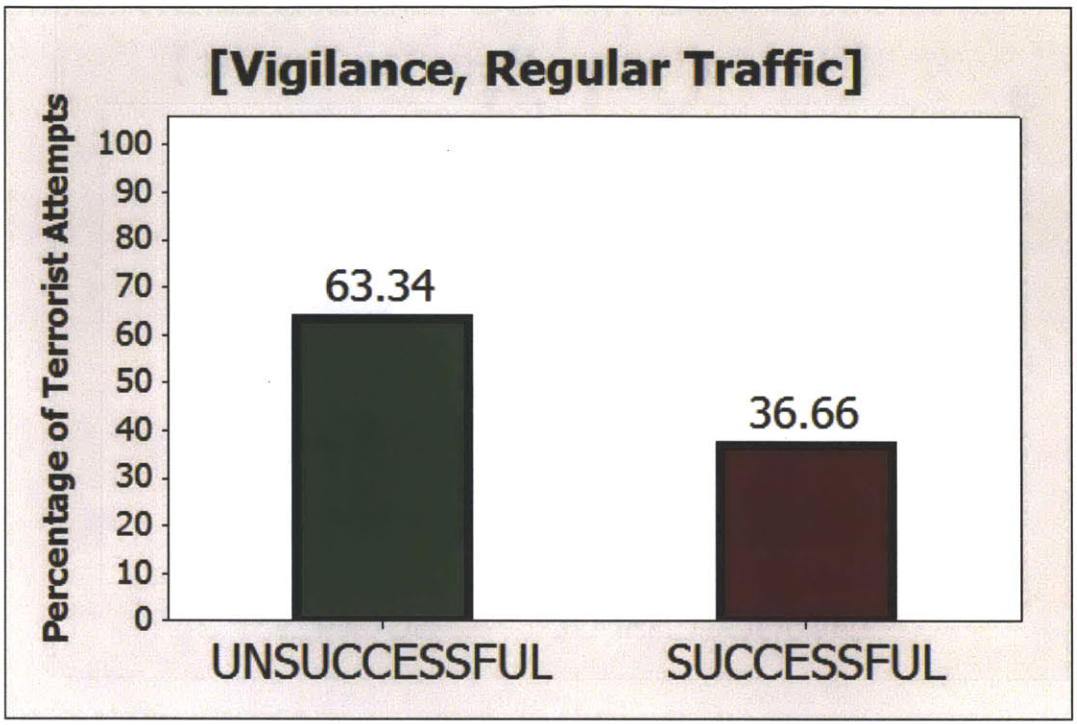

FIGURE D-7: CHANCE OF SUCCESSFUL TERRORIST ATTACK ON VIGILANT SOS (GIVEN ATTEMPT UNDER REGULAR TRAFFIC CONDITIONS)

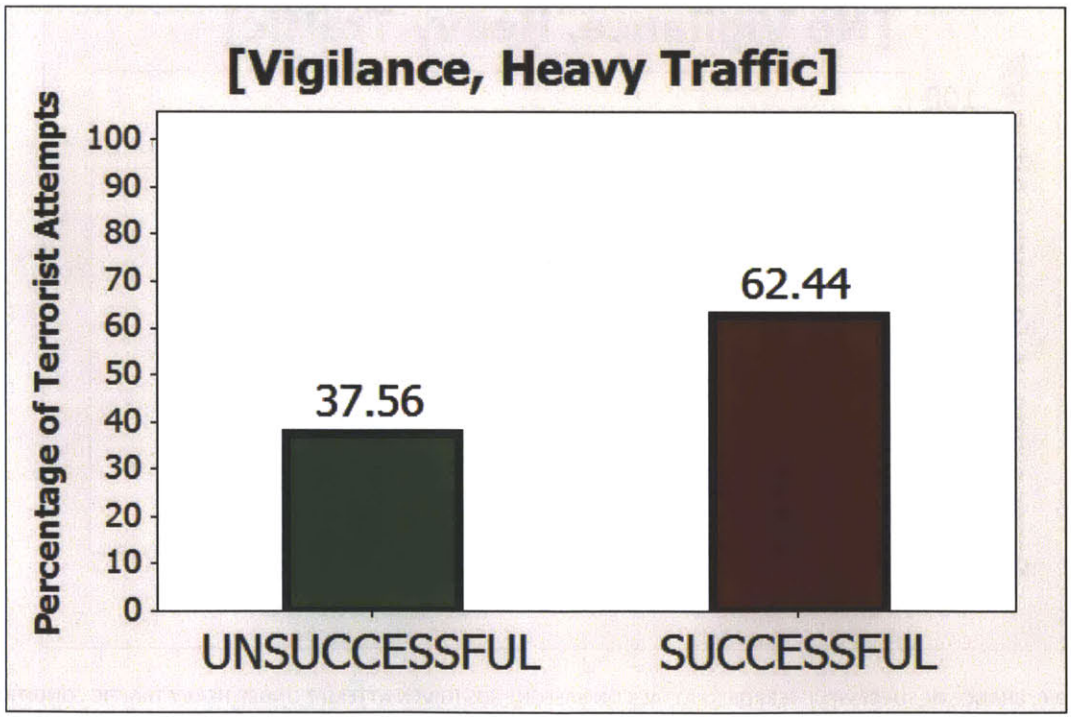

FIGURE D-8: CHANCE OF SUCCESSFUL TERRORIST ATTACK ON VIGILANT SOS (GIVEN ATTEMPT UNDER HEAVY TRAFFIC CONDITIONS)

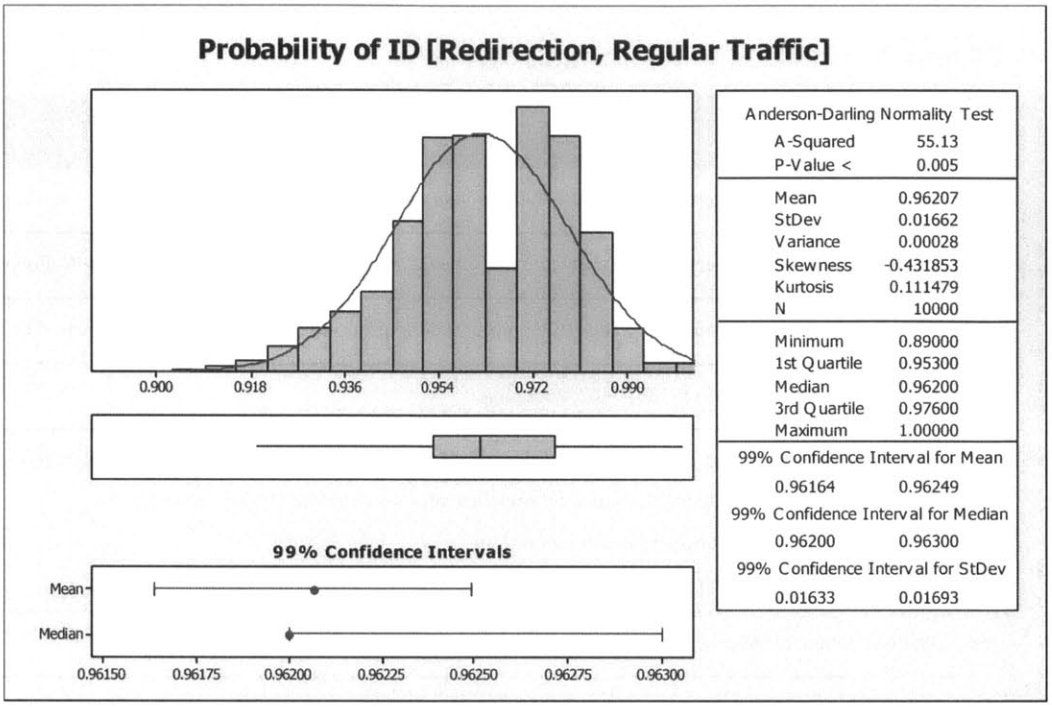

FIGURE D-9: PROBABILITY OF ID WITH REDIRECTION STRATEGY (UNDER REGULAR TRAFFIC)

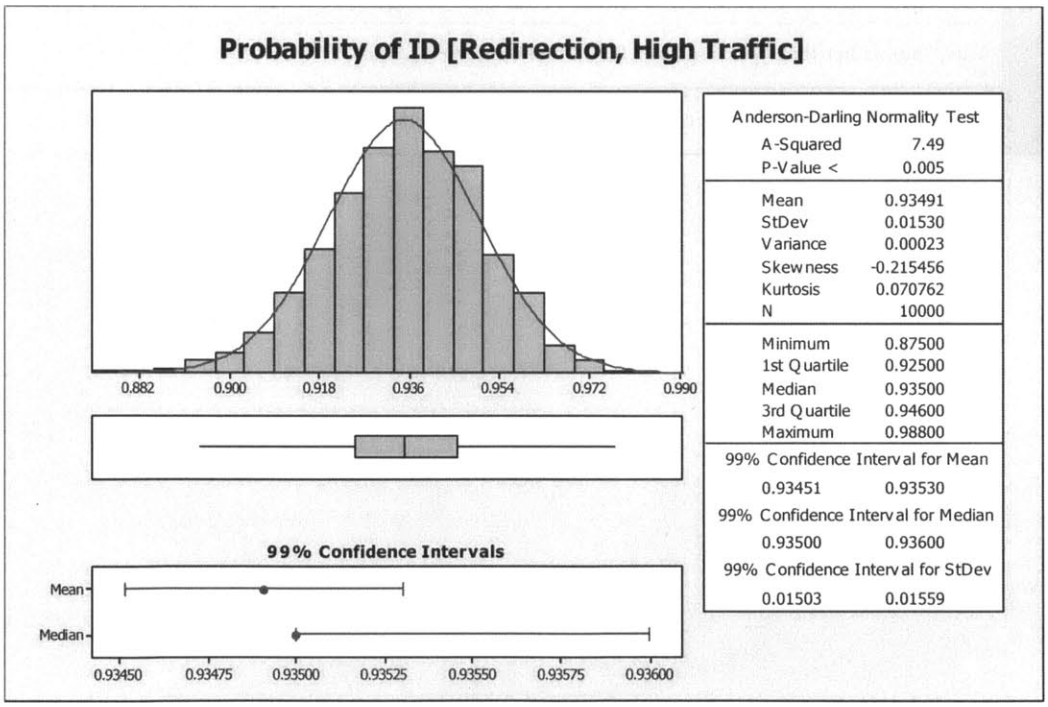

FIGURE D-10: PROBABILITY OF ID WITH REDIRECTION STRATEGY (UNDER HEAVY TRAFFIC)

IMAGE CREDITS

FIGURE	SOURCE
1-1	http://ecosocialismcanada.blogspot.com/2012/03/one-year-after-fukushima-nuclear.html
2-2	http://www.newyorkwallpapershd.com/user-content/uploads/wall/o/69/New-York-Yellow-Cab-Wallpaper.jpg
2-3	http://www.webmarketinggroup.co.uk/Blog/the-webmarketing-group-the-fourth-emergency-service-1801.aspx
2-4	http://img123.imageshack.us/img123/9602/z6mq.jpg
3-2	http://www.fareastgizmos.com/media_devices/sony_nasd5hd_stereo_system_with_40_gb_hard_drive.php http://omgtoptens.com/misc/vehicles/top-10-diesel-cars-under-rs-10-lakhs-in-india-2012/ http://www.howstuffworks.com/wireless-network.htm http://www.zazzle.com/combined_arms_poster-228781205990224197
6-4	http://globalchange.mit.edu/news-events/news/news_id/141
6-6	http://en.wikipedia.org/wiki/Northeast_blackout_of_2003
7-9	http://www.environmentalgraffiti.com/nature/news-7-most-iconic-skyscrapers-earth-being-struck-lightning
7-10	http://nonsei2gm.blogspot.com/2011_01_01_archive.html
8-1	http://hemodynamics.blogspot.com/2007/06/found-down-hmshsdm-commencement-speech.html

REFERENCES

Albert, R., Albert, I., & Nakarado, G. (2004). Structural vulnerability of the North American power grid. *Physical Review E, 69*(2), 25103.

Alexander, C., Ishikawa, S., & Silverstein, M. (1977). *A pattern language: towns, buildings, construction* (Vol. 2): Oxford University Press, USA.

Allenby, B., & Fink, J. (2005). Toward inherently secure and resilient societies. *Science, 309*(5737), 1034-1036.

Anderson, B., & Mutch, J. (2011). *Preventing Good People From Doing Bad Things: Implementing Least Privilege*. New York, NY: Apress.

Ashby, W. R. (1962). Principles of the self-organizing system. *Principles of Self-organization*, 255-278.

Babcock, C. (2010). Amazon Says Wikileaks Plug Pulled Over SLA Violation. *InformationWeek.* Retrieved December 4, 2012 from http://www.informationweek.com/cloud-computing/software/amazon-says-wikileaks-plug-pulled-over-s/228500303.

Backlund, A. (2000). The definition of system. *Kybernetes, 29*(4), 444-451.

Banks, J., & Carson, J. S. (1984). Discrete event system simulation.

Barbier, M. (2007). *D-Day Deception: Operation Fortitude and the Normandy Invasion*. Westport, CT: Greenwood Publishing Group,.

Beck, K., Crocker, R., Meszaros, G., Vlissides, J., Coplien, J. O., Dominick, L., et al. (1996). *Industrial experience with design patterns*. the 8th international conference on Software engineering.

Beer, S. (1984). The viable system model: its provenance, development, methodology and pathology. *Journal of the operational research society*, 7-25.

Beesemyer, J. C. (2012). *Empirically Characterizing Evolvability and Changeability in Engineering Systems.* S.M. Thesis, MIT, Cambridge, MA.

Beesemyer, J. C., Fulcoly, D. O., Ross, A., & Rhodes, D. H. (2011). *Developing Methods to Design for Evolvability: Research Approach and Preliminary Design Principles*.

Beesemyer, J. C., Ross, A. M., & Rhodes, D. H. (2012). An Empirical Investigation of System Changes to Frame Links between Design Decisions and Ilities. *Procedia Computer Science, 8*.

Benkler, Y. (2011). WikiLeaks and the protect-ip Act: A New Public-Private Threat to the Internet Commons. *Daedalus, 140*(4), 154-164.

Bernanke, B. S. (2010). Preserving a Central Role for Community Banking. *2010 Speeches.* Retrieved March 20, 2010 from http://www.federalreserve.gov/newsevents.

Beynon-Davies, P. (1999). Human error and information systems failure: the case of the London ambulance service computer-aided despatch system project. *Interacting with Computers, 11*(6), 699-720.

Biltgen, P. T. (2007). *A Methodology for Capability-Based Technology Evaluation for Systems-of-Systems.* PhD Thesis, Georgia Institute of Technology, Atlanta, GA.

Boardman, J., & Sauser, B. (2006, 2006). *System of Systems-the meaning of of.* the IEEE / SMC International Conference on System of Systems Engineering, Los Angeles,CA.

Brill, J. H. (1998). Systems engineering— A retrospective view. *Systems Engineering, 1*(4), 258-266.

Bruzzone, A. G., Tremori, A., & Merkuryev, Y. (2011). Assymmetric Marine Warfare: PANOPEA A Piracy Simulator for Investigating New C2 Solutions. *SCM MEMTS 2011*, 32.

Buschmann, J., Crider, T., Guillermo Ferraris, E. G., Gungor, H., Hoffmann, S., Kelley, M., et al. (2005). *Maritime Domain Protection in the PACOM AOR*. Monterey, CA: Naval Postgraduate School.

Buurman, J., Zhang, S., & Babovic, V. (2009). Reducing risk through real options in systems design: the case of architecting a maritime domain protection system. *Risk Analysis, 29*(3), 366-379.

Carter, J. (1979). Primary Resources: Crisis of Confidence. Retrieved January 13, 2013 from http://www.pbs.org/wgbh/americanexperience/features/primary-resources/carter-crisis/.

Central Intelligence Agency. (2012a). Country Compairison - Oil - Exports. *CIA World Factbook.* Retrieved October 30, 2012 from https://www.cia.gov/library/publications/the-world-factbook/.

Central Intelligence Agency. (2012b). Oil - Consumption By Country. *CIA World Factbook.* Retrieved October 30, 2012 from https://www.cia.gov/library/publications/the-world-factbook/.

Chattopadhyay, D. (2008). *A Framework for Tradespace Exploration of Systems of Systems.* S.M. Thesis, MIT, Cambridge, MA.

Cloutier, R. J., & Verma, D. (2007). Applying the concept of patterns to systems architecture. *Systems Engineering, 10*(2), 138-154.

Crawley, E., de Weck, O., Eppinger, S., Magee, C., Moses, J., Seering, W., et al. (2004). *The Influence of Architecture in Engineering Systems*. the MIT Engineering Systems Symposium.

Dahmann, J., Rebovich, G., Lowry, R., Lane, J., & Baldwin, K. (2011). *An implementers' view of systems engineering for systems of systems.* the 2011 Annual IEEE International Systems Conference.

Dahmann, J. S., & Baldwin, K. J. (2008). *Understanding the current state of US defense systems of systems and the implications for systems engineering.* the 2nd Annual IEEE Systems Conference, Montreal, Canada.

de Neufville, R., & Scholtes, S. (2011). *Flexibility in Engineering Design*. Cambridge, MA: MIT Press.

de Weck, O. L., Roos, D., & Magee, C. L. (2012). *Engineering Systems: Meeting Human Needs in a Complex Technological World*. Cambridge, MA: MIT Press.

de Weck, O. L., Ross, A. M., & Rhodes, D. H. (2012). *Investigating Relationships and Semantic Sets amongst System Lifecycle Properties (Ilities)*. 3rd International Engineering Systems Symposium CESUN 2012.

Department of Defense. (2008a). Dictionary of Military and Associated Terms. Retrieved January 12, 2013 from http://www.dtic.mil/doctrine/new_pubs/jp1_02.pdf.

Department of Defense. (2008b). Systems Engineering Guide for Systems of Systems. Retrieved http://www.acq.osd.mil/se/docs/SE-Guide-for-SoS.pdf from

Department of Defense. (2010). DoD Architecture Framework 2.02. from http://dodcio.defense.gov/Portals/0/Documents/DODAF/DoDAF_v2-02_web.pdf.
Department of Homeland Security. (2012). Threat and Hazard Identification and Risk Assessment Guide. Retrieved Dec 11, 2013 from http://www.state.nj.us/njhomelandsecurity/grants/grants-main/06-21-12-thira-guide.pdf.
Dickerson, C. (2009). Defense applications of SoS. In M. Jamshidi (Ed.), *Systems of Systems Engineering: Principles and Applications*. Boca Raton, FL: CRC Press.
Dijkstra, A. (2007). *Cybernetics and Resilience Engineering: Can Cybernetics and the Viable System Model Advance Resilience Engineering?* the Resilience Engineering Workshop.
DiMario, M. J. (2006). *System of systems interoperability types and characteristics in joint command and control*. the 2006 IEEE/SMC International Conference on System of Systems Engineering.
Downer, J. (2009). *When failure is an option: redundancy, reliability and regulation in complex technical systems*. London, UK.: London School of Economics and Political Science.
Eisner, H., Marciniak, J., & McMillan, R. (1991). *Computer-aided system of systems engineering.* the IEEE Conference on Systems, Man, and Cybernetics, Charlottesville, VA.
Elkins, D. A., Huang, N., & Alden, J. M. (2004). Agile manufacturing systems in the automotive industry. *International Journal of Production Economics, 91*(3), 201-214.
Ellison, R., Fisher, D., Linger, R., Lipson, H., & Longstaff, T. (1997). *Survivable network systems: An emerging discipline*. Pittsburgh: Carnegie Mellon.
Ellison, R. J., & Woody, C. (2007). Survivability Challenges for Systems of Systems. *News at SEI.* Retrieved January 23, 2013 from www.sei.cmu.edu/library/abstracts/news-at-sei/securitymatters200706.cfm.
Fulcoly, D. O. (2012). *A Normative Approach to Designing for Evolvability: Methods and Metrics for Considering Evolvability in Systems Engineering.* S.M. Thesis, MIT, Cambridge, MA.
Furlani, C. M. (2009). *Minimum Security Requirements for Federal Information and Information Systems*: DIANE Publishing.
Gonzalez, A., Piel, E., Gross, H. G., & Glandrup, M. (2008). *Testing Challenges of Maritime Safety and Security Systems-of-Systems*. Delft, Netherlands: Software Engineering Research Group, Delft University of Technology.
Gonzalez, R. A. (2009). *Crisis response simulation combining discrete-event and agent-based modeling*. the 6th International ISCRAM Conference.
Gunderson, L. H. (2000). Ecological resilience--in theory and application. *Annual review of ecology and systematics*, 425-439.
Gupta, Y. P., & Goyal, S. (1989). Flexibility of manufacturing systems: concepts and measurements. *European Journal of Operational Research, 43*(2), 119-135.
Hall, A. D. (1962). *A Methodology for Systems Engineering*. Princeton, NJ: D. Van Nostrand Company.

Henderson, S., & Mason, A. (2005). Ambulance Service Planning: Simulation and Data Visualisation. In M. L. Brandeau, F. Sainfort & W. P. Pierskalla (Eds.), *Operations Research and Health Care* (Vol. 70, pp. 77-102): Springer US.

Hildbrand, S., & Bodhanya, S. (2011). *The Viable System Model (VSM) and Qualitative Studies: A Research Perspective to Manage in a World of Complexity.* the 10th European Conference on Research Methodology for Business and Management Studies.

Hogan, L. J. (2001). Terrorism: Defensive Strategies for Individuals, Companies and Governments. *No.: ISBN 0-9659174-5-2*, 452.

Hollnagel, E., Woods, D. D., & Leveson, N. (2006). *Resilience Engineering: Concepts and Precepts*. Aldershot, UK: Ashgate.

Hsiao, R., Compeau, D., & Hou, S. T. (2010). *Taiwan Taxi's ICall System: Realizing the Value of GPS Dispatch Systems*. London, ON: Richard Ivey School of Business.

Huynh, T. V., & Osmundson, J. S. (2006). *A Systems Engineering Methodology for Analyzing Systems of Systems Using the Systems Modeling Language (SysML).* the 2nd Annual System of Systems Conference, Ft. Belvoir, VA.

Institute of Electrical and Electronics Engineers. (1990). *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*. New York: Institute of Electrical and Electronics Engineers

International Standards Organization. (2011). Information technology - Security techniques - Information security risk management. Geneva, Switzerland.

Ishimatsu, T., Leveson, N., Thomas, J., Katahira, M., Miyamoto, Y., & Nakao, H. (2010). *Modeling and hazard analysis using STPA.* the Conference of the International Association for the Advancement of Space Safety, Huntsville, Alabama.

Jackson, S. (2010). *Architecting Resilient Systems: Accident Avoidance and Survival and Recovery from Disruptions*. Hoboken, NJ: John Wiley & Sons.

Jackson, S. (2012). Resilience Principles for Human-Made Systems. *Systems Engineering,* (Draft submitted to the INCOSE Resilience Working Group).

Kilgore, R., Harper, K., Nehme, C., & Cummings, M. (2007, May 2007). *Mission planning and monitoring for heterogeneous unmanned vehicle teams: A human-centered perspective.* the AIAA Infotech@Aerospace Conference, Sonoma, CA.

Lankford, J. (2003, March 8-15, 2003). *Measuring system and software architecture complexity.* the 2003 IEEE Aerospace Conference.

Law, A. M., & Kelton, D. W. (2000). *Simulation Modeling and Analysis*: McGraw-Hill.

Leathrum, J. F., Mathew, R., & Mastaglio, T. W. (2011). Modeling the impact of security and disaster response on cargo operations. *Simulation, 87*(8), 696-710.

Leduc, P. A., Rash, C. E., & Manning, S. D. (2006). Human Factors in UAV Accidents from http://intellibriefs.blogspot.com/2006/01/human-factors-in-uav-accidents.html.

Lee, M. T. (1998). The Ford Pinto case and the development of auto safety regulations, 1893-1978. *Business and Economic History, 27*(2), 390-401.

Lerner, E. J. (2003). What's wrong with the electric grid? *The Industrial Physicist.* Retrieved June 28, 2012 from www.aip.org/tip/INPHFA/vol-9/iss-5/p8.html.

Leveson, N. (1995). *Safeware: system safety and computers*: ACM New York, NY, USA.

Leveson, N. (2004). A new accident model for engineering safer systems. *Safety Science, 42*(4), 237-270.
Leveson, N. G. (2002). System Safety Engineering: Back to the Future. Retrieved Dec 11, 2012 from www.sunnyday.mit.edu/book2.pdf.
LG Electronics. (2012). LG Smart Appliances for 2012 Deliver Connectivity, Efficiency Though Smart Thinq Technologies. *Home Appliances.* Retrieved Nov 11, 2012 from www.lgnewsroom.com/ces2012/view.php?product_code=95&product_type=95&post_index=1828.
Liao, Z. (2003). Real-Time Taxi Dispatching Using Global Positioning Systems. *Communications of the ACM, 46*(5), 81-83.
Lonergan, B. (1992). *Insight: A Study of Human Understanding* (Vol. 3). Toronto: University of Toronto Press.
Long, J. F. (2012). Interactive Video Game Technologies. *Communication Technology Update, 10/e, 10*, 184.
Long, T. (2009). This Day In Tech: Software Glitch Cripples Ambulance Service. *This Day In Tech.* Retrieved December 6, 2012 from http://www.wired.com/thisdayintech/tag/london-ambulance-failure/.
MacCollum, D. V. (2007). *Construction safety engineering principles: designing and managing safer job sites*: McGraw-Hill Professional.
Madni, A. M., & Jackson, S. (2009). Towards a conceptual framework for resilience engineering. *Systems Journal, IEEE, 3*(2), 181-191.
Maier, M. W. (1998). Architecting Principles for System of Systems. *Systems Engineering, 1*(4), 267-284.
Maier, M. W., & Rechtin, E. (2000). *The Art of Systems Architecting*: CRC Press.
McAllister, T., & Corley, G. (2002). *World Trade Center Building performance study: Data collection, preliminary observations, and recommendations*: Federal Emergency Management Agency, Federal Insurance and Mitigation Administration.
McFee, M. (2011). The Principle of Least Privilege – A Failure in MA. *my blog*. Retrieved from http://maggiemcfee.com/2011/05/18/the-principle-of-least-privilege-a-failure-in-ma/
McGaughey, R. E. (1999). Internet technology: contributing to agility in the twenty-first century. *International Journal of Agile Management Systems, 1*(1), 7-13.
McHugh, J. (2012). Networking the Soldier. Retrieved November 11, 2012 from http://www.bctmod.army.mil/index.html.
McManus, H., & Hastings, D. (2006). A framework for understanding uncertainty and its mitigation and exploitation in complex systems. *IEEE Engineering Management Review, 34*(3), 81-94.
McManus, H., Richards, M. G., Ross, A., & Hastings, D. (2007). *A Framework for Incorporating" ilities" in Tradespace Studies.".* the AIAA Space 2007, Long Beach, CA.
McNeill, D. (2011). Why the Fukushima disaster is worse than Chernobyl. Retrieved December 13, 2012 from http://www.independent.co.uk/news/world/asia/why-the-fukushima-disaster-is-worse-than-chernobyl-2345542.html.

Mekdeci, B., Ross, A. M., Rhodes, D. H., & Hastings, D. (2011a). *System Architecture Pliability and Trading Operations in Tradespace Exploration*. the IEEE International Systems Conference 2011.

Mekdeci, B., Ross, A. M., Rhodes, D. H., & Hastings, D. E. (2011b). *Examining Survivability of Systems of Systems*. the INCOSE International Symposium 2011.

Mekdeci, B., Ross, A. M., Rhodes, D. H., & Hastings, D. E. (2012). *A taxonomy of perturbations: Determining the ways that systems lose value.* the 2012 IEEE International Systems Conference, Vancouver, Canada.

Mittal, S., Zeigler, B. P., Martín, J. L. R., & Sahin, F. (Eds.). (2008). *Modeling and simulation for systems of systems engineering*. Hoboken, NJ: John Wiley & Sons, .

Monperrus, M., Long, B., Champeau, J., Hoeltzener, B., Marchalot, G., & Jézéquel, J. M. (2010). Model-driven architecture of a maritime surveillance system simulator. *Systems Engineering, 13*(3), 290-297.

NASA. (2011). *NASA Space Flight Program and Project Management Requirements*: NASA.

Networking the Soldier. (2012). Retrieved November 11, 2012 from http://www.bctmod.army.mil/index.html.

Ng, C. W. (2007). *Discrete-Event Simulation with Agents for Modeling of Dynamic Asymmetric Threats in Maritime Security*: Master's Thesis, Naval Postgraduate School, Monterey, CA.

Oxford English Dictionary (2012a), *Principle*, Oxford University Press: Oxford, UK.

Oxford English Dictionary (2012b), *Resilience*, Oxford University Press: Oxford, UK.

Oxford English Dictionary (2012c), *Robustness*, Oxford University Press: Oxford, UK.

Oxford English Dictionary (2012d), *Survivable*, Oxford University Press: Oxford, UK.

Parasuraman, R., Sheridan, T. B., & Wickens, C. D. (2000). A model for types and levels of human interaction with automation. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, 30*(3), 286-297.

Passikoff, R. (2012). Hanging Up on Landlines. *Forbes.* Retrieved November 12, 2012 from http://www.forbes.com/sites/marketshare/2012/05/17/hanging-up-on-landlines/.

Ramsey, W. G. (1995). *D-Day then and now* (Vol. 1): Battle of Britain Prints International.

Raymond, E. S. (2004). *The art of Unix programming*: Addison-Wesley Professional.

Rechtin, E. (1992). The art of systems architecting. *Spectrum, IEEE, 29*(10), 66-69.

Renfoe, N. A. (2011). Threat/Vulnerability Assessments and Risk Analysis. *Whole Building Design Guide.* Retrieved December 4, 2012 from http://www.wbdg.org/resources/riskanalysis.php.

Richards, M. G. (2009). *Multi-Attribute Tradespace Exploration for Survivability.* PhD Thesis, MIT, Cambridge, MA.

Richards, M. G., Hastings, D., Rhodes, D., & Weigel, A. (2007, March 2007). *Defining Survivability for Engineering Systems.* the 5th Conference on Systems Engineering Research, Hoboken, NJ.

Richards, M. G., Ross, A., Hastings, D., & Rhodes, D. (2008). *Two Empirical Tests of Design Principles for Survivable System Architecture.* the 18th INCOSE Symposium, Utrecht, The Netherlands.

Richards, M. G., Ross, A., Hastings, D., & Rhodes, D. (2009). *Survivability Design Principles for Enhanced Concept Generation and Evaluation.* the 19th Annual INCOSE International Symposium, Singapore.

Roiter, N. (2011). *Small Businesses, Big Losses: How SMBs Can Fight Cybercrime*. Manhasset, NY: InformationWeek.

Ross, A. M. (2006). *Managing unarticulated value: Changeability in multi-attribute tradespace exploration.* PhD Thesis, MIT, Cambridge, MA.

Ross, A. M., Beesemyer, J., & Rhodes, D. (2011). *A Prescriptive Semantic Basis for System Lifecycle Properties (SEAri Working Paper Series, WP-2011-2-1)*: SEAri Working Paper Series, WP-2011-2-1.

Ross, A. M., Hastings, D., Warmkessel, J., & Diller, N. (2004). Multi-attribute tradespace exploration as front end for effective space system design. *Journal of Spacecraft and Rockets, 41*(1), 20-28.

Ross, A. M., O'Neill, M. G., Hastings, D. E., & Rhodes, D. H. (2010). *Aligning Perspectives and Methods for Value-Driven Design.* the American Institute of Aeronautics and Astronautics: Space 2010 Conference and Exposition.

Ross, A. M., & Rhodes, D. H. (2008a). *Architecting Systems for Value Robustness: Research Motivations and Progress*. the 2008 2nd Annual IEEE Systems Conference.

Ross, A. M., & Rhodes, D. H. (2008b). *Using natural value-centric time scales for conceptualizing system timelines through epoch-era analysis.* the 2008 INCOSE International Symposium.

Ross, A. M., & Rhodes, D. H. (2011). Anatomy of a Change Mechanism (SEAri Working Paper WP-2011-1-2). MIT.

Ross, A. M., Rhodes, D. H., & Hastings, D. E. (2008). Defining changeability: Reconciling flexibility, adaptability, scalability, modifiability, and robustness for maintaining system lifecycle value. *Systems Engineering, 11*(3), 246-262.

Saltzer, J. H., & Schroeder, M. D. (1975). The protection of information in computer systems. *Proceedings of the IEEE, 63*(9), 1278-1308.

Sargent, R. G. (2005). *Verification and validation of simulation models.* the Winter Simulation Conference.

Sauser, B., Boardman, J., & Gorod, A. (2008). System of systems management. *System of Systems Engineering*, 191-217.

Schumacher, M., Fernandez-Buglioni, E., Hybertson, D., Buschmann, F., & Sommerlad, P. (2006). *Security Patterns: Integrating security and systems engineering* (Vol. 7): Wiley.

Shah, N. B. (2013). *Influence Strategies for Systems of Systems.* PhD Thesis, MIT, Cambridge, MA.

Shappell, S. A., & Wiegmann, D. A. (2001). Applying reason: The human factors analysis and classification system (HFACS). *Human Factors and Aerospace Safety*.

Shappell, S. A., & Wiegmann, D. A. (2002). The human factors analysis and classification system-HFACS. from http://www.slc.ca.gov/Division_Pages/MFD/Prevention_First/Documents/2002/Presentation%20by%20Douglas%20Wiegmann.pdf.

Sheard, S., & Mostashari, A. (2008). *A Framework for System Resilience Discussions.* the 18th Annual International INCOSE Symposium.
Smart, A. G., Amaral, L. A. N., & Ottino, J. M. (2008). Cascading failure and robustness in metabolic networks. *Proceedings of the National Academy of Sciences, 105*(36), 13223.
Snuder, M. (2011). Sony's profit forecast turns into a $3.2 billion loss. *USA Today.* Retrieved October 22, 2012 from http://www.usatoday.com/tech/news/2011-05-23-sony-earnings_n.htm.
Staff, J. (2010). Joint Publication 1-02: Department of Defense Dictionary of Military and Associated Terms. *Washington, DC.*
Stallings, W. (2003). *Cryptography and network security* (Vol. 2): Prentice Hall.
Sterman, J. D. (2001). System Dynamics Modeling. *California management review, 43*(4), 8.
Stevenson, R. W. (2005). After Days of Criticism, Emergency Director Resigns. *The New York Times.* Retrieved October 21, 2012 from http://www.nytimes.com/2005/09/13/national/nationalspecial/13brown.html.
Taguchi, G., & Cariapa, V. (1993). Taguchi on robust technology development. *Journal of pressure vessel technology, 115*, 336.
Thomas, B. K. (2008). *Improving Maritime Prepositioning Force (MPF) offloads using modeling and simulation*: DTIC Document.
U.S.-Canada Power System Outage Task Force. (2004). *Final Report on the August 14, 2003 Blackout in the United States and Canada: Cause and Recommendations*. Washington, DC.
Vego, M. N. (2009). *Joint Operational Warfare Theory and Practice and V. 2, Historical Companion*. Newport, RI: Naval War College Press.
Von Neumann, J. (2012). *The computer and the brain*: Yale University Press.
Wasson, C. S. (2006). *System analysis, design, and development: Concepts, principles, and practices*: John Wiley & Sons.
Westrum, R. (2006). *A Typology of Resilience Situations*. Aldershot, England: Ashgate.
Wiener, E. L., & Nagel, D. C. (1988). *Human factors in aviation*. San Diego, CA: Academic Press.
Wolfgang, P. (1994). *Design patterns for object-oriented software development*: Reading, Mass.: Addison-Wesley.
Woods, D. D., Johannesen, L. J., Cook, R. I., & Sarter, N. B. (1994). *Behind human error: Cognitive systems, computers and hindsight*: DTIC Document.
WordNet (2011), *Pliability*, Princeton University: Princeton, New Jersey.
Yee, K. P. (2003). *Secure interaction design and the principle of least authority.* the CHI Workshop on Human-Computer Interaction and Security Systems.
Yolles, M. (2000). From viable systems to surfing the organisation. *Journal of Applied Systems, 1*(1), 127-142.
Zhivich, M., & Cunningham, R. K. (2009). The real cost of software errors. *Security & Privacy, IEEE, 7*(2), 87-90.
Zhou, B., Drew, O., Arabo, A., Llewellyn-Jones, D., Kifayat, K., Merabti, M., et al. (2010). *System-of-systems boundary check in a public event scenario.* the th International Conference on System of Systems Engineering.