Fall 2020

An approach to system of systems resiliency using architecture and agent-based behavioral modeling

Paulette Bootz Acheson

Department: Engineering Management and Systems Engineering

AN APPROACH TO SYSTEM OF SYSTEMS RESILIENCY USING ARCHITECTURE AND AGENT-BASED BEHAVIORAL MODELING

by

PAULETTE BOOTZ ACHESON

A DISSERTATION

Presented to the Graduate Faculty of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

SYSTEMS ENGINEERING

2020

Approved by:

Cihan H. Dagli, Advisor
Steve M. Corns
David L. Enke
Nil Kilicay-Ergin
Ruwen Qin
Allyson D. Yarbrough

ABSTRACT

In today's world it is no longer a question of whether a system will be compromised but when the system will be compromised. Consider the recent compromise of the Democratic National Committee (DNC) and Hillary Clinton emails as well as the multiple Yahoo breaches and the break into the Target customer database. The list of exploited vulnerabilities and successful cyber-attacks goes on and on. Because of the amount and frequency of the cyber-attacks, resiliency has taken on a whole new meaning. There is a new perspective within defense to consider resiliency in terms of Mission Success.

This research develops a new approach of assessing resiliency from the Mission Engineering perspective. Mission Engineering is a new field of systems engineering where the Mission is the system of interest. The Mission requires a SoS with the goal of Mission Success. To the literature, this research contributes an approach to evaluate SoS resiliency based on Mission Success. An agent-based model (ABM) called the SoS architecture resiliency model (SARM) was developed and is a second contribution to the literature. The SARM includes a fuzzy architecture assessor (FAA) as well as SoS behavior represented using fuzzy decision analysis (FDA). The SARM uses DoD architecture framework (DoDAF) views and includes threats. Results show that resiliency can be measured using SARM given the systems, capabilities, and interfaces. Tests with a generic SoS and with a specific SoS provide a proof of concept for the method. To summarize, this research contributes to the literature a method and an executable model for evaluating architecture resiliency as well as the FAA and the FDA.

ACKNOWLEDGMENTS

I would like to thank my family for their support and encouragement. I appreciate all the support from the members of my committee who graciously gave of their time and arranged their schedules to attend my defense. Thank you to Dr. Dagli who met with me each week to review and provide guidance on my research.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

NOMENCLATURE

Symbol	Description
β	Angle of Attack
$\propto$	Proportional Operator

1. INTRODUCTION

1.1. PROBLEM

This Resiliency has become important topic considering the recent cyber-attacks on emails. Consider the recent hack of the Democratic National Committee (DNC) and Hillary Clinton emails [1][2]. These hacks demonstrated a lack of resilience in two different email systems which resulted in national consequences. Worse than that is the set of Yahoo breaches that first occurred in late 2016 and continued into early 2017 [3][4][5]. In less than 6 months, Yahoo accounts were hacked three times! These examples illustrate the lack of resilience in those email systems.

According to cyber experts, it is no longer a matter of whether a system will experience a cyber-attack but rather a question of when the cyber-attack will occur and how well the system will respond to the cyber-attack. [6][67]. The ability of the system to continue to operate and satisfy its mission goals is a characteristic of the system. This characteristic of the system is something that must be designed into the system and is called resiliency.

The literature review in Section 2 identifies research in system resilience. While some of the system resilience research might also apply to system of systems (SoS), the development of the SoS is quite different from the system development. The system development adds capabilities to a system constrained by the system priorities, funding, and schedule. But in SoS development there are multiple sets of priorities and schedules set by the systems themselves. The SoS manager must pull together the system level capabilities into higher level SoS capabilities and the SoS architecture. For this reason, the approach to SoS resiliency must be different from that used in the past to address system resiliency.

Within the literature, there seems to be a lack of SoS resilience. This research examines SoS resilience in terms of the SoS architecture and provides a methodology and model for investigating resilience in SoS. A new way of looking at resiliency is in terms of architecture and specifically SoS architecture. The SoS architecture really defines the SoS. Time to develop a new SoS architecture after a cyber incident (or cyber-attack) can be a measure of the SoS resiliency. A longer time for development would mean longer time to get capabilities to the warfighter or longer time to complete the mission. Thus, a SoS architecture that has a shorter time for re-development after a cyber-attack, would be a more resilient architecture.

This research developed a framework for modeling SoS architecture and evaluating the SoS resiliency. The SoS architecture resiliency model (SARM) is the result of this research.

1.2. DESCRIPTION

The SARM represents the SoS architecture using an agent-based model. The SARM receives the SoS architecture from the SoS architecture model and uses that as the SoS description. The Resiliency is defined as the degree to which the mission success can still be achieved after a cyber-attack. Resilience capacity is the measurement of resiliency.

The SARM was developed as part of this novel approach to SoS resiliency assessment. The SARM is an agent-based model with models to represent the SoS manager and individual systems' behavior. The system behavior is used to represent the systems' ability to support the SoS request for capabilities. The SARM includes a threat whose behavior is used to represent vulnerability to the SoS to determine the resilience of the SoS architecture. The SARM includes a fuzzy assessment model to qualitatively evaluate the resilience of the SoS architecture that results from varying the systems' behaviors. Some background and previous research are presented in the following section.

1.3. BACKGROUND AND PREVIOUS RESEARCH

Initial research on resilience was mostly in the area of material strength. Resilience in ecological systems followed and then resilience in urban planning. Today resilience is evident in many domains including psychology, enterprise, and engineering.

Early resilience models were not executable. More recently other models such as the ones by White [7] and Holt [8] do not provide an executable model. Long [9] talks about an executable model but just defines an approach to creating an executable SoS model from DoDAF views. The SARM, developed as part of this research, is executable and can be run with different scenarios to see the effect of different SoS architectures on resilience.

The SARM is an agent-based model (ABM) which more accurately reflects the real-world Enterprise or SoS development. Using an ABM preserves the independence and autonomy of the constituent systems which is essential characteristic of a SoS [10].

Other models such as Bowen [11] use discrete event models to represent the asynchronous behavior of the constituent systems in the SoS. A discrete event model can only consist of sequences of events, which limits its ability to represent real world systems.

The SARM is adaptable and generic for any domain. Adjusting the input parameters allows the SARM to be applied to different domains. Except for White [7], previous models were specific to certain domains such as missile defense [12], immune system [11], weather [13], and threat detection [14]. The model defined in Baldwin [10] appears at first to be a generic SoS model, but on closer examination the implementation is actually a collaborative SoS.

White [7] provides a static class diagram of a SoS that includes the elements of a SoS namely: systems, event, state, data, and function and can be considered a generic model. However, the White model is not executable as is the SARM.

In summary, the SARM has the following advantages over previous models:

- Agent-based
- Enterprise or SoS level
- Generic/adaptable to multiple domains
- Executable
- Flexible for different SoS and system behaviors

More background research is presented in Section 2. The SARM developed in this research is used to support the objectives of this research which are described in the next subsection.

1.4. RESEARCH OBJECTIVES AND CONTRIBUTIONS

1.4.1. Research Objectives. The objective of this research was to develop an approach to resiliency assessment based on mission success. Resiliency in the literature is focused on system resiliency and the approaches to resiliency are based on the resiliency of each component in the system. This research evaluates the resiliency of SoS with the understanding that the Mission is beyond the scope of a single system. This resiliency approach can be applied to the Enterprise level as well as the SoS level. So, within this research references to Enterprise are also appropriate.

1.4.2. Research Contributions. This research defines an approach to resiliency that is not found in the literature. Rather than approaching resiliency at the unit or component level, this research shows how SoS resiliency is evaluated based on the Mission. The Mission is the military, naval, or aerospace task that depends on the SoS for success. As mentioned in DoDAF v2.0 volume 1, Figure 2-2 [15], Missions drive the architectures, and the architectures align to the mission outcomes. If the desired mission outcome is success, then the architecture must align to the mission success.

The main contribution of this research is the approach to SoS architecture resiliency based on Mission Success. This research describes the role of the SoS architecture in the Mission and in Mission Success. An executable model, SARM, is created to qualitatively assess the SoS resiliency. An interface between the SoS architecture model and the SARM is designed so that the SoS resiliency is directly tied to the architecture. Finally, the SARM includes a threat agent that simulates the real-world obstacles to Mission Success. There are additional secondary contributions from this research, and they are mentioned below.

One of the secondary contributions from this research is the SARM. Evaluation of architecture resilience at the Enterprise or SoS level required some way to represent architecture degradation due to exploitation of a vulnerability. An executable model was used to fulfill this need. Since the model should represent the real-world scenario as much as possible, an agent-based type of model was selected. An ABM best represents the independence and autonomous behavior of the constituent systems. An executable model allows running the model several times with different scenarios. The SARM is one of the contributions from this research. The SARM is an executable ABM that allows for the use of different models to represent the SoS manager behavior as well as the systems' behaviors.

For this research it was necessary to create a method to assess the architecture resilience. As another secondary contribution, a fuzzy architecture assessment (FAA) was developed. The FAA uses fuzzy mathematics to qualitatively evaluate the architecture resilience. The FAA was developed in 2012 [16]. The FAA is described in more detail later in this dissertation.

In addition to the FAA, it was necessary to develop models that represent the SoS manager behavior and the systems' behavior. Fuzzy decision analysis (FDA) and a fuzzy negotiation model (FNM) became another contribution from this research and were used to represent the SoS manager behavior. The FNM was designed to use a fuzzy associative memory (FAM) to represent the fuzzy rules. The behavior of the individual systems was represented using FDA and FAM as well as a probability model. These are all described in detail later in this dissertation.

As mentioned earlier, the interface between the SoS architecture model and the SARM was created. This leads to another contribution of this research, namely the incorporation of the DoDAF architecture views as the SoS architecture representation used in the SARM. Within the DoD programs there is increasing emphasis on representing system and SoS architectures using DoDAF views. Many of the DoDAF views are essentially unified modeling language (UML) or system modeling language (SysML) views that can be executed. Since the DoDAF views are required for every program they already exist in a manner that is understandable to the SoS and system managers. In addition, many organizations within the DoD maintain repositories of SoS and system architectures that are DoDAF views. Using the DoDAF views in the SARM provides a means to validate the DoDAF views.

To summarize, the contributions from this research are:

- Method to evaluate Enterprise or SoS architecture resilience
- SARM
- FAA
- FDA and FNM representing the SoS behavior
- FAM that contains fuzzy rules which can be altered for different scenarios
- Integration of the SoS architecture model with the SARM

For this research it is necessary to define an approach to evaluating architecture resilience that is based on mission success. This approach is described in the next subsection.

1.5. RESEARCH APPROACH

One way to evaluate architecture resilience is the time it takes to restructure the architecture so that mission success can still be accomplished after architecture degradation caused by an exploited vulnerability. Malicious threats seek to exploit Enterprise and SoS vulnerabilities with the aim of preventing mission success. The degree to which the architecture can still accomplish mission success is the resilience capacity. The greater the resilience capacity the better the Mission Resiliency.

In the SARM an architecture vulnerability is exploited, and the architecture becomes degraded. The Enterprise or SoS manager must negotiate with the constituent systems and determine how best to update the architecture to mitigate the vulnerability and still achieve mission success. The time it takes to do this is an indication of Mission Resilience. The longer it takes to evolve the SoS architecture after a cyber-attack, the less resilient. The shorter the time the greater the resilience. Resilience capacity is the inverse of this time.

1.6. RESEARCH RESULTS

The results are described in detail in Section 5 but are briefly described here.

This research describes a method to evaluate Mission Resiliency. This method is implemented in the SARM. Figure 1.1 shows the model results. The results show that the resilience capacity is related to the number of systems involved in the Mission. The system is involved in the Mission when the system supports the SoS request to provide certain capabilities. The evaluation of Mission Resiliency is dependent on the Mission. A model was developed as part of this research and provides the proof of concept for this approach.

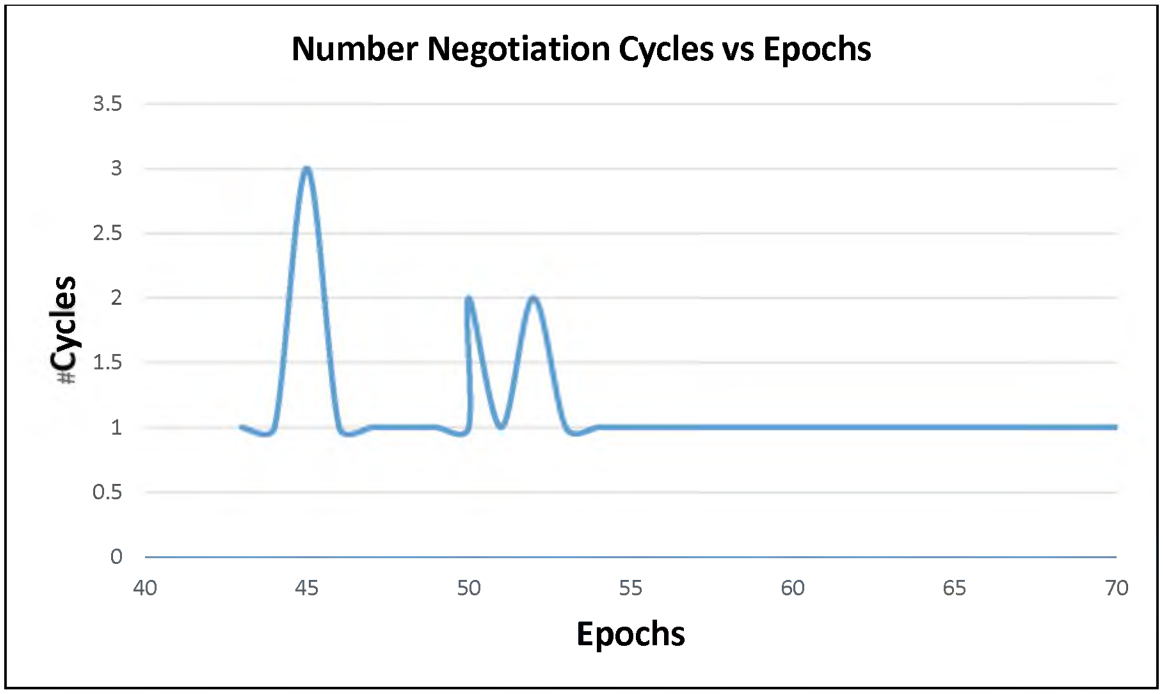


Figure 1.1. Resilience Capacity in Terms of Negotiation Cycles

1.7. DISSERTATION ORGANIZATION

This dissertation is organized into sections. Section 1introduces the motivation for this research. Section 2 is the literature review that discusses the development and resilience of a SoS. Section 3 defines Mission Resilience. Section 4 describes the SoS Architecture Resiliency Model (SARM). Section 5 shows the results of this research. Section 6 summarizes the conclusions. Section 7 describes future work. Section 8 provides the definitions and assumptions.

2. LITERATURE REVIEW

This section focuses on SoS development and resilience. Other topics related to this research are discussed in other sections within this dissertation.

2.1. BACKGROUND

Resilience research was first in the areas of the textile industry, urban planning, and disaster preparedness. As technology grew so did adversarial attack on the cyberspace which led to resilience research in information technology (IT) systems such as networks, computers, and other IT systems. Most of this resilience research has evaluated resilience in a bottom-ups approach using a resilience value for each configuration item in some combination to achieve an overall system resilience value. While this approach works for evaluating system resilience, this does not work for evaluating mission resilience.

Webster's dictionary defines resilience as "an ability to recover from or adjust easily to misfortune or change" [17]. Reggiani provides another definition of resilience as the amount of disturbance that a system can absorb before it moves away from an equilibrium state to another state [18]. According to this definition, a resilient system would be one that could absorb a catastrophic event and maintain the same capabilities.

The earliest work in resilience is in engineering sciences in the resilience of materials. The Charpy impact test (sometimes referred to as the Charpy specimen test) was devised in 1900 to test the resilience in metals [19] [20]. Later in the middle of the

20th century, resiliency was applied in textile industry [21][22] to determine the strength of materials. This resilience like the Charpy test was done as a physical test.

Resilience then appeared in evaluation of ecological systems [23][24]. Berkes and Folke were the first to integrate ecological and social resilience. Holling was the first to propose resilience and capacity found in those ecological systems that would be classified as unstable. The instability being the characteristic that caused the ecological system to survive. More on resilience capacity in the next section.

In the early part of the 21st century, resiliency research was applied to urban renewal and city planning. Cities and urban areas were evaluated for their resilience to changing population demographics as well as resilience to a natural catastrophe [25]. More recently resilience in Ecology was given a more precise definition as change in response to disturbance [26].

Resilience has been recently researched in many different areas including ecosystems, psychology, enterprise, and engineering [27]. Having briefly outlined the background for resiliency, the next sections describe resiliency research as applied in systems, architecture, and SoS.

2.1.1. Resilience in Systems. Recently, system resiliency has become a hot topic of research. This is due to the increase in cyber-attacks that exploit system vulnerabilities and result in degraded or failed system performance. Whether the system is a medical patient system or a satellite system, the probability of cyber-attacks is high. Cyber experts indicate that it is not a question of whether a cyber-attack will occur but when it will occur. For this reason, resiliency as a system attribute has become important. But the question remains: how to measure system resilience. This has prompted research into the area of system resilience. Some of this research is presented in the following paragraphs.

Argonne Lab was one of the first to define resilience in terms of other characteristics. This was to assess the resilience of a community or a region. Argonne Lab has put together the parts of resilience with the measures that enhance resiliency. Argonne defines resilience as composed of anticipate, resist, absorb, respond, adapt, and recover [28]. For each of the components, Argonne defines an action to enhance resilience. Actions are being prepared, mitigation, response, and recovery. According to Argonne, being prepared enhances anticipate; mitigation enhances resist and absorb; response enhances respond and adapt; and finally, recovery enhances recover. Three categories of approaches they call quantitative approaches: implement resilience strategies prior to unusual event; implement resilience strategies after the event; implement resilience strategies before, during, and after the event.

Resilience has been used in ecology and archaeology. A paper by Juan-Garcia and others [29] provides a background and history of resilience in ecology. The paper categorizes ecology papers that deal with resilience but provides no new information.

In addition to the Argonne Lab paper, there are papers in the literature that define resilience in terms of other characteristics. One of those papers describes resilience in the motor-fuel supply chain (FSC). The paper by Beheshtian and others applies network resilience concepts to the motor-fuel supply chain [30]. This paper provides a mathematical representation of resilience in FSC and evaluates the ability of the FSC to recover from a failure due to unusual events. In this paper, resilience is defined in terms of three characteristics: absorption, adaptation, and restoration. Beheshtian’s paper describes a mathematical model to maximize resilience in FSC. Resilience in motor-fuel supply chain depicted in a quantitative model evaluates a system's ability to recover from a failure due to unusual events. This paper defines resilience. But it does not really define exactly what is meant by absorption, adaptation, or restoration. The paper claims to provide a qualitative approach but the details only reveal a quantitative model and data. This contrasts with the Argonne Lab paper that describes resilience using three additional characteristics. Neither paper models nor provides case study using these characteristics. In practice more characteristics does not necessarily provide a better picture of resilience as the lines differentiating the characteristics becomes more indeterminate.

Another paper that uses the three characteristics for resilience is the research by Nan & Sansavini [31] in the electric power domain. Nan & Sansavini apply resilience to the electric power infrastructure and describe resilience in terms of absorption, adaptation, and recovery. This paper provides definitions for these characteristics. Absorptive capability defined as ability to reduce negative impacts in the system from a disruptive event. Adaptive capability defined as ability to self-organize and adapt the system to minimize consequences. Restorative capability is ability to repair the system.

One domain that is receiving a lot of attention is the energy domain. There are different approaches to resilience in the energy domain. One approach to resilience involves decoupling the subsystems [32]. This was described as a method for resilience in the electric power grid. By decoupling the subsystems each subsystem was able to continue to provide electricity if one subsystem was attacked. The approach required each subsystem to also be used as a testbed to test the ability of the grid to continue operation when one of the subsystems went down. Decoupling can support resilience but is not a solution for most systems. Another approach to resilience in the energy domain involves the energy storage system [33] which is necessary for photovoltaic (PV) generation. The paper by Confrey shows the resilience for different energy storage system architectures.

Resilience is described in the archaeological domain [34][35]. Bradtmöller includes a literature review of the resilience research and points out the multiple ways that resilience is defined and used in archaeology. Weiberg describes resilience in Aegean archaeology with respect to the adaptation cycle.

Wang proposes developing countermeasures for cyber-attacks as a method to resilience in the financial domain [36]. Command and control (C&C) of Botnets is usually a Block-chain architecture with various mechanisms. This paper describes the different mechanisms and how to disrupt them to protect the system. As new Block-chain mechanisms emerge, new counter-mechanisms should be developed to take down the Block-chain. This paper is specific to defending against Botnets that have a Block-chain architecture and does not work for other domains.

Resilience has also been applied to hardware architectures [37][38][39]. Leng and Kalra both address resilience in the Graphic Processor Unit (GPU) but approach it in different ways. Ascia describes resilience in the chip design for wireless communication. Asymmetric resilience is described for computer hardware where the responsibilities are split between the parts [37]. The GPU is responsible for detecting the cyber-attack and the CPU is responsible for recovery. This paper is focused on low level architecture of computer hardware so the approach to resilience only works for computer hardware. Kalra uses a static approach to resilience by means of a compiler to support resilience in the GPU [38]. ArmorAll is a compiler for software system that uses a GPU. ArmorAll optimizes duplicate instructions in order to enhance resilience. Ascia states that the architecture of the wireless network on a chip has a tradeoff between resilience and energy [39]. This paper suggests some methods for keeping the energy requirement low while maintaining resilience if the user can provide some details of the data in the communication. All three papers are specific to hardware and not applicable to other domains. On the other hand, the SARM, to be described later, can be applied to the hardware domain.

This concludes the look at resilience in systems. The next section provides a look at resilience in architecture.

2.1.2. Resilience in Architecture. Resilience in architecture is not well represented in the literature. But it is especially important for cyber physical systems. Some resilience in architecture has been in the construction domain [40] [41]. Campos deals with resilience in the architecture of school buildings [40] and argues that resilience means the school building should motivate learning. Hardiilla & Nugroho describe resilience in vernacular architecture as an integral part of Urban Planning [41] to accommodate cultural and social change. Both papers are specific to physical buildings and not applicable to other domains. The model described in the next paragraph could be used in this domain.

One paper describes how resilience in architecture can be modeled [42]. The paper by Acheson describes an agent-based model that determines architecture resilience. In this paper the SoS architecture resilience model (SARM) is described, and resilience is defined by how well the architecture handles a cyber-attack. More on this paper later.

Having discussed resilience in systems and in architecture, the next section describes resilience in a SoS.

2.1.3. Resilience in SoS. Within the literature there are examples of resilience in SoS. One of them is applied to a SoS that assists automobile drivers [43]. This driver assist SoS consists of a lead vehicle with a following vehicle that must maintain a safe distance between the vehicles. The communication between the vehicles is normally wireless but the research suggests using a speed estimator in the follower vehicle to estimate the lead vehicle's motion. So, the resilience approach is not use wireless communication which is susceptible to cyber-attack. But rather to estimate the distance to the lead vehicle. This is a specific approach to resilience that does not work in most cases. Most all modern vehicles have wireless communication and replacing the wireless communication with an estimator will not work in most cases.

One of the resilience papers seems appropriate for the current year as it involves voting and media influence [44]. This is an agent-based model where the voter is an agent. Media influence is an external influence on the agent. This paper simulates the effect on voting after a terrorist attack as happened in the Spanish election in March of 2004. The simulation reproduced the spread of the media coverage of the terrorist attacks using an agent population. The simulation also models the voting in the Spanish election after the terrorist attacks. The agent-based simulation best represents the voter actions and media coverage. This paper describes resilience in terms of a voter's action to not be influenced by the terrorist attack. External influence of the media coverage of terrorist attack on the voter choosing who to vote for. Resilience is the magnitude of the external influence needed to get the voter to change its vote.

One approach to resilience in SoS is to use contracts and negotiation to create a collaborative resilient SoS [45]. Kilian calls this engineering resilience into the system. The approach lacks any method of assessing the system resilience or determining where there might be vulnerabilities. It also does not consider the Mission in the evaluation of resilience. The modeling in this paper is just a paper representation and not implemented in a modeling tool where it could be validated across the systems or executed.

Resilience in SoS can be modeled [42]. The paper by Acheson describes resilience in an SoS using the SoS architecture. This paper uses an agent-based model to represent the SoS and the threat as agents and resilience is determined by how well the SoS can respond to the cyber-attack.

While this section does focus on resilience in SoS, it is important to note that this is related to Mission Resilience. Recall that Mission Engineering is where the Mission is the system of interest and the Mission requires an underlying SoS. Mission success is dependent on a SoS that can complete the mission. Mission resilience is defined as the ability to successfully complete the Mission despite cyber-attacks. The paper by Acheson and Dagli [42] describes the SoS Architecture Resiliency Model (SARM) which bases the resiliency evaluation on Mission Success. Further details of SARM are in Section 3.

The literature mentioned in this and the previous sections focused on research in resilience for systems, architecture, and SoS. But, as mentioned earlier, one of the questions is how to quantify resilience. Research in resilience quantification is provided in the next section.

2.1.4. Resiliency and Resilience Capacity. One of the first to consider a quantitative value for resilience is Mebarki [19]. Mebarki defined resilience index as the capacity for metal to absorb a disturbance. This is structural resilience and not system or SoS resilience. But it is a way to quantify resilience using a boundary for resilience and categorizing the resilience as supra-resilient or non-resilient depending on whether the resilience index is above or below an optimum or threshold value, respectively. A more recent view of resilience capacity is purported by Jackson as mentioned in the following paragraph.

Jackson defines resilience as a combination of avoidance, survival, and recovery and considers brittleness to be the opposite of resilient [46]. Jackson also defines capacity as the ability of a system to absorb a disruption without loss of capability [46]. It is easy to see the similarity of this "capacity" to the resilience index Mebarki [19] defined.

In social science the resilience capacity index (RCI) refers to the degree to which a metropolitan area can absorb a disruption without the loss of capability. One non-profit organization defines Resilience Capacity Index (RCI) in terms of twelve factors [47]. The ability of an urban area to jump back after change is hypothetically influenced by these twelve factors. The RCI is used to compare metropolitan regions. While this definition seems reasonable, there does not seem to be any details on how it is calculated.

Chen specifies RCI in terms of a disaster and coins the term DRC: "Disaster Resilience Capacity (DRC) is ability of community to survive following a disaster" [48]. Changes to a community can be from disasters or from the normal evolution of the area, such as the movement of people to/from the area for jobs.

Recently, General Hyden used the term "resilience capacity" as a measure of system resiliency [49]. Hyten explains that a physical number like the MTBF is not useful in describing resiliency. The resilience is determined by whether the Mission can be completed. Will Mission Success be achieved in a contested environment? Hyten again describes resiliency in terms of space architectures and capabilities at the NDIA Luncheon in 2015 [50]. Here Hyten talks about resilient space architectures and using other characteristics such as disaggregation to define resilience. General Hyten further expanded on this idea at the 2016 Space Symposium by defining a resilient enterprise as one whose vision is focused on the threats [51]. Hyten specifically mentioned the need for a resilient enterprise and resilient capabilities. Resilient capabilities are designed with the threats in mind. The degree to which the system supports mission success in a contested environment is resilience capacity.

Resilience specific to Mission Engineering, which involves a SoS, is somewhat lacking in the literature. In order to look at resilience specific to Mission Engineering, it is necessary to consider the SoS that enables the mission. As mentioned earlier, the paper on modeling SoS resiliency [42] describes a technique that can be used for evaluating Mission Resiliency. This technique involves evaluating SoS resiliency in terms of Mission Success. For this research, resilience capacity is defined in terms of Mission Success. The following section describes how resilience capacity is related to Mission Success.

2.2. LITERATURE SUMMARY

Initial resilience was specific to metals and later applied to urban planning. Current research into system resiliency mostly focuses on the system or parts of the system. As mentioned earlier, Mission Resilience is lacking in the literature. This dissertation fills this gap by addressing Mission Resilience.

The literature is also lacking in a model of resilience. Jackson developed a generic state transition diagram that can be used to evaluate system resiliency [66] but only if the system is described in terms of states. The state transition diagram is not executable. This research fills another gap and provides an executable model that can be used to evaluate Mission Resiliency as well as SoS and system resiliency. The model is an updated version of the SoS Architecture Resiliency Model (SARM) [42] which is discussed in a latter section. Before that some context for this research is needed. The context is Mission Resilience and is discussed in the following section.

3. MISSION RESILIENCE

In Mission Engineering, the purpose for the SoS is to fulfill some mission or task that requires capabilities at a higher level than system capabilities. SoS level capabilities depend on contributions from each of the constituent systems to provide a system level capability that supports the higher level SoS capability. Capabilities from each constituent system contribute to the overall SoS capabilities. Cyber-attacks remove system level capabilities which in turn affect the SoS level capabilities. If the SoS can still accomplish and fulfill the mission after a cyber-attack, then mission success is achieved and the SoS can be considered resilient. The SARM uses this method to assess resiliency of the SoS. But first some descriptions are in order.

Mission Engineering has a Mission and an underlying SoS to support the Mission. The SoS is defined by its architecture and the architecture determines the resilience of the SoS. Mission Resilience is defined as the ability of the SoS to complete the Mission despite cyber-attacks.

Cyber-attacks will happen and can result in losing a system in the SoS or an interface in the SoS. When a system or interface is lost, the SoS must adjust to replace the missing capability with another capability (or capabilities) that will still support Mission Success. This process of replacing the missing capability amounts to a re-architecture of the SoS with the goal of Mission Success.

The loss of a system or interface means that the SoS, and hence the Mission, can no longer depend on that system/interface. The SoS is in a degraded state when this happens. So, the question is whether Mission Success can still be achieved in this degraded state. In the degraded state a re-architecture is performed to account for the reduction in capability. It is possible that a specific capability is not needed for the Mission Thread and Mission Success can be achieved without that capability. For this research, it is assumed that the capability is necessary for the Mission Thread. Since the capability is necessary for the Mission Thread, a cyber-attack that takes out the capability can result in Mission Failure. Mission Resilience would mean that Mission Success would still occur despite the capability loss. For this to happen, some other system capability (or capabilities) must replace the lost system capability. The measure of Mission Resilience is Resilience Capacity.

Resilience capacity is the degree to which the SoS can still deliver the Mission despite a cyber-attack that results in the loss of a system or interface. After the loss of the system or interface the SoS must reconfigure the SoS architecture for a workaround to the loss. A system or interface loss really means losing the system level capability that feeds into the SoS higher level capability. To see how the system level capability relates to the SoS level capability consider the following description.

There are two aspects to resilience capacity. Mission Success depends on backfilling or replacing or finding a work around for the capability lost when the cyber-attack removed the system or the interface. But Mission Success also depends on getting the workaround capability within the required time frame. Workaround in this context refers to either a replacement or substitution or method of overcoming the system or interface loss. So, the first step in the process is finding the workaround which means re-architecting the SoS to support the Mission Thread.

According to the MEI Guidebook [52], a Mission Thread is a “series of integrated end-to-end tasks linked by interoperable interfaces required to successfully achieve a desired outcome within a given tactical situation”. The Mission Thread can be viewed as a sequence of tasks. Equation (1) show the Mission Thread:

$$Mission\,Thread = T_1, T_2, \dots, T_O \tag{1}$$

Each task requires one or more SoS capabilities. Each task can be represented as shown in equation (2):

$$T_i = C_1, C_2, \dots, C_L \tag{2}$$

Each SoS capability requires multiple system level capabilities. Each SoS capability can be represented by the sequence of system capabilities as shown in equation (3):

$$C_i = c_{j,k}, c_{j+1,k=1}, \dots, c_{j+R,k+R} \tag{3}$$

where $c_{j,k}$ is capability k of system j

Thus, each task in the Mission Thread can be represented by a set of system level capabilities from the constituent systems relevant to the Mission Thread. Equation (4) shows this representation:

$$T_i = c_{j,k}, c_{j+1,k=1}, \dots, c_{j+R,k+R} \tag{4}$$

The SoS capabilities are documented and decomposed on the DoDAF CV-2 diagram. Many projects use the DoDAF framework to describe the architecture. Each SoS level capability is a composition of lower level system capabilities. The lower level system capabilities are the decomposition of the SoS capabilities on the SoS CV-2 diagram. The SoS SV-5b shows the allocation of the capabilities to the individual systems. Figure 3.1 and Figure 3.2 illustrate the CV-2 and SV-5b, respectively.

The CV-2 and SV-5b feed into the SARM to describe the capabilities. The cyber-attack consists of removing one of the system capabilities at the lowest level of the CV-2. The system capability at the lowest level of the CV-2 is one of the capabilities needed to complete the Mission. It is required for the Mission Thread.

The CV-2 and SV-5b feed into the SARM to describe the capabilities. The cyber-attack consists of removing one of the system capabilities at the lowest level of the CV-2. The system capability at the lowest level of the CV-2 is one of the capabilities needed to complete the Mission. It is required for the Mission Thread.

A cyber-attack can cause the loss of a system or interface which results in loss of one or more of the system level capabilities (c_i) mentioned above. For this research, the loss of the system level capability affects Mission Success, and the system level capability is part of SoS capability that is part of the Mission Thread. The cyber-attack and subsequent loss of the system capability triggers a re-architecture of the SoS.

The re-architecture takes the form of the SoS requesting new capabilities from some or all the constituent systems. The SoS request includes the required capabilities,

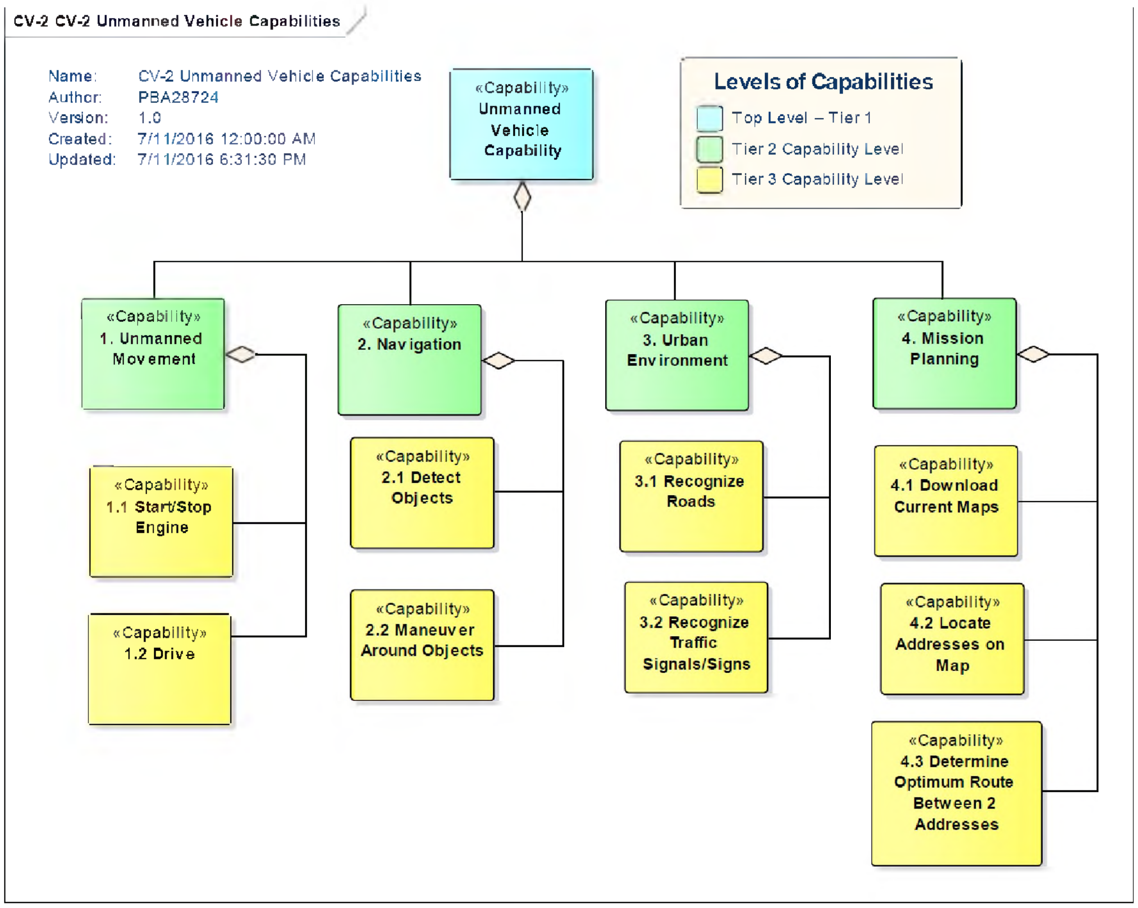

CV-2 CV-2 Unmanned Vehicle Capabilities
Name: CV-2 Unmanned Vehicle Capabilities
Author: PBA28724
Version: 1.0
Created: 7/11/2016 12:00:00 AM
Updated: 7/11/2016 6:31:30 PM
«Capability» Unmanned Vehicle Capability
Levels of Capabilities
Top Level – Tier 1
Tier 2 Capability Level
Tier 3 Capability Level
«Capability» 1. Unmanned Movement
«Capability» 1.1 Start/Stop Engine
«Capability» 1.2 Drive
«Capability» 2. Navigation
«Capability» 2.1 Detect Objects
«Capability» 2.2 Maneuver Around Objects
«Capability» 3. Urban Environment
«Capability» 3.1 Recognize Roads
«Capability» 3.2 Recognize Traffic Signals/Signs
«Capability» 4. Mission Planning
«Capability» 4.1 Download Current Maps
«Capability» 4.2 Locate Addresses on Map
«Capability» 4.3 Determine Optimum Route Between 2 Addresses

Figure 3.1. CV-2 Showing the Capabilities Taxonomy

required performance, and time to provide the capabilities. Each system responds to the SoS request with what it can provide and the SoS determines if that will support the Mission Thread and Mission Success. The cycle continues until Mission Success can be achieved. The time required to get to Mission Success is the resilience capacity. The longer it takes to evolve the SoS architecture after a cyber-attack, the less resilient. The shorter the time the greater the resilience. The SARM then evaluates the best

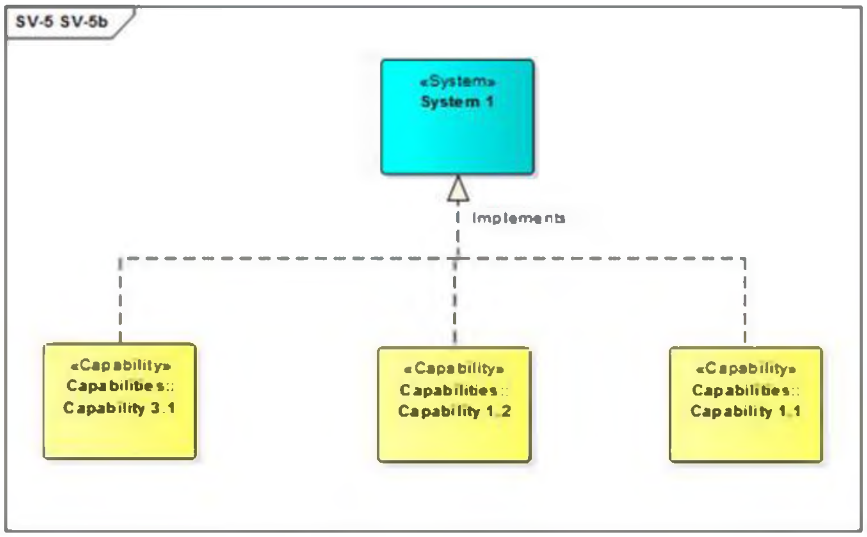


Figure 3.2. SV-5b Showing the Capabilities Allocated to a Specific System

workaround for the available time frame. The SARM measured how long it takes to restructure the SoS architecture so that the Mission can be successfully completed. Details of the SARM are in the next section.

4. SOS RESILIENCY ARCHITECTURE MODEL (SARM)

This section describes the SoS Architecture Resiliency Model (SARM). SARM is a key component of determining Mission Resilience. Mission Resilience is defined as the ability to still complete the Mission despite a cyber-attack. This research developed an approach and a model to assess Mission Resilience. The approach assumes a cyber-attack results in the loss of a system level capability that is required to support a SoS capability required for the Mission Thread. The SoS architecture must be re-structured to accommodate the loss and so that the Mission can still be accomplished. This approach is illustrated in the SARM.

SARM is an agent-based model [42] which was extended from the SoS architecture model [53]. The mathematical foundation for the SoS architecture model was described in 2012 [54] [55]. The model structure was explained in 2013 [53]. The SARM is an extension of the SoS architecture model [42] that adds resilience from a Mission success perspective. The SARM is an agent-based model that includes a Threat agent.

In an agent-based model, the agents are the building blocks. An agent is abstraction of the characteristics and behavior of some entity. The abstraction becomes a real representation in the model when the agent is instantiated. Until the agent is instantiated it remains an abstraction or just an idea and is not actually a physical piece of the model. Each instantiation becomes its own process.

SARM has three agents: SoS agent, System agent, and Threat agent. The SoS agent gets the SoS capabilities from the DoDAF capability views. The SoS coordinates with the constituent systems to get the system capabilities from them that are needed to support the SoS level capabilities. The system agent has constraints that can affect its ability to provide capabilities to the SoS. Both the SoS agent and the system agent have inherent behavior that represents the activities in interfacing between the two.

The Threat agent attacks systems which results in the removal of a system capability. The capability loss triggers the SoS to perform a re-architecture of the SoS so that the Mission can continue. The SARM keeps track of the time it takes to re-architect which is used to determine the Mission Resilience.

The SARM makes use of multiple underlying models. The SARM pulls all these models together to determine the Mission Resilience. These underlying models consist of the SoS behavior model, the SoS architecture assessment, and the system behavior model. SARM is designed so that other models can be used for any of these underlying models. The SARM starts with the SoS capabilities which are delineated in DoDAF views.

The DoDAF CV-2 views (Figure 3.1) show the SoS capabilities. The SV-5b views (Figure 3.2) show the system level capabilities that support the SoS level capabilities. And the SV-1 views show the system interfaces. Figure 4.1 below is an example of an SV-1 view.

The SoS agent first determines what capabilities are needed from which systems as defined in the SV-5b views. The SoS agent then requests those system capabilities from the respective systems. When the SoS agent receives the system responses the SoS architecture assessment mode is invoked to determine whether the Mission can be completed. If the Mission can be completed, then the resilience capacity is calculated based on the time it took to get the architecture to this point. If the Mission cannot be

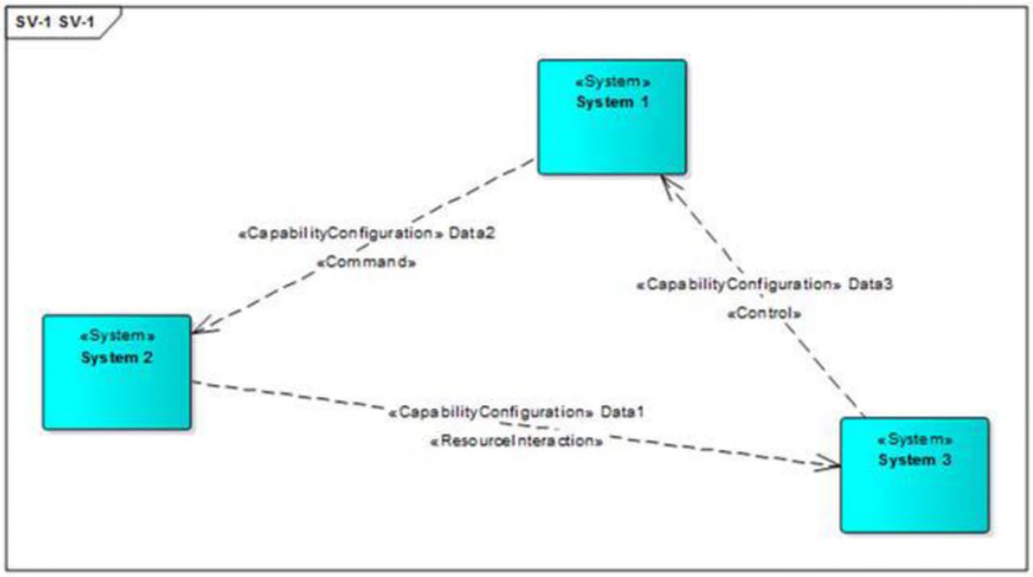


Figure 4.1. SV-1 Showing the System Interfaces

completed the process is repeated starting with the SoS re-architecture and the cycle continues until Mission Success can be achieved. It is possible that certain architectures with certain cyber-attacks might not ever reach Mission Success in which case the Mission Resilience is zero or negative. Figure 4.2 shows the SARM structure.

The first paragraphs in this section provided a high-level description of the structure of the SARM. The following sections describe the details of the components that make up the SARM.

4.1. SOS AGENT

This section describes the details of the SoS agent. The SoS development starts with a desired set of capabilities and a desired performance for each capability.

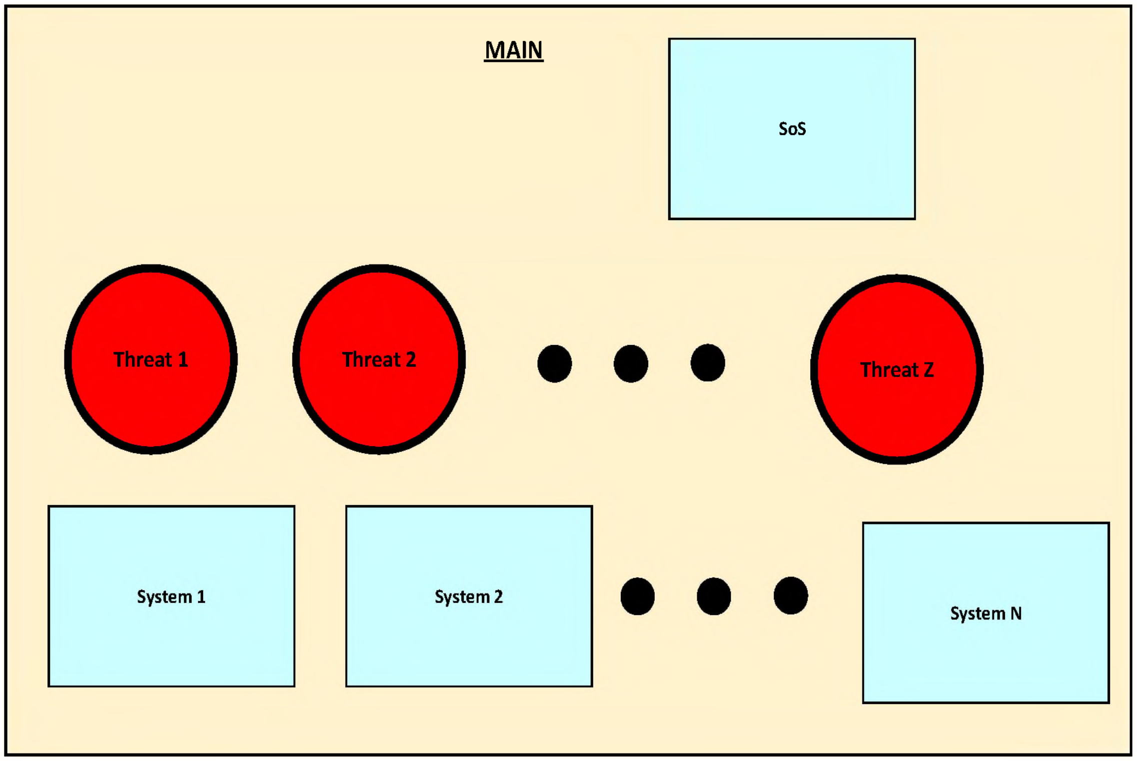


Figure 4.2. SARM Structure

In addition, each capability is weighted according to its degree of importance to the warfighter mission. Mathematically these are represented in equation (5):

$$SoS.C_i = (C_1, C_2, \dots, C_n) \tag{5}$$

where SoS.C_i represents the set of capabilities necessary to complete the Mission. These are the desired capabilities that the SoS requests of the systems in order to complete the Mission despite the loss due to a cyber-attack. Each C_i represents a SoS level capability that is comprised of a set of system level capabilities as shown in equation (3).

Each SoS capability is weighted, and the weights are represented by equation (6):

$$SoS.w_i = (w_1, w_2, \dots w_n) \tag{6}$$

where,

i = Capability number

n = number of capabilities needed for the Mission

w_i = weight for the SoS level capability i

The weights (w_i) are fuzzy values which are {None, Low, High, Heavy}. The fuzzy membership function for weight is shown in Figure 4.3.

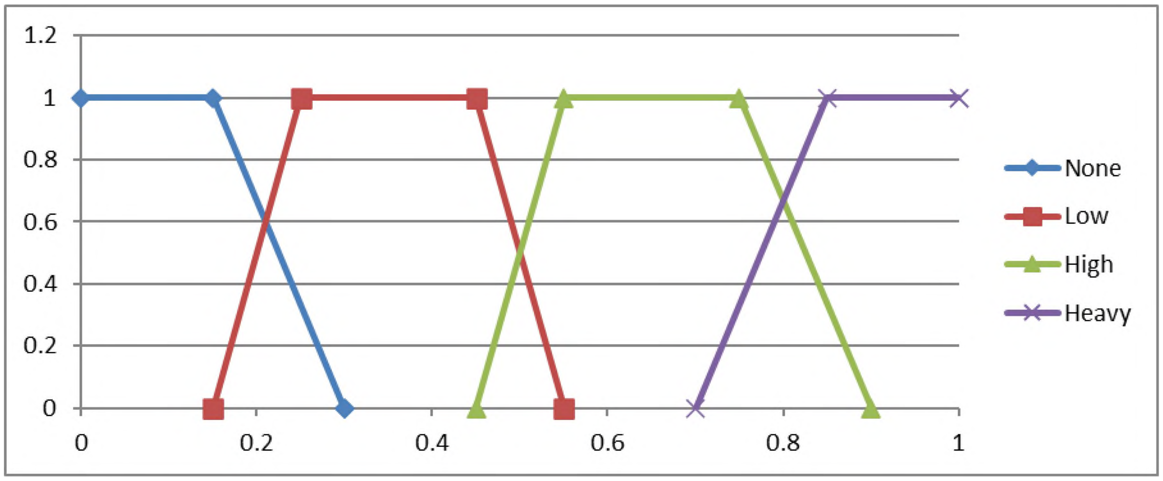


Figure 4.3. Weight Fuzzy Membership Function

Associated with each SoS capability is a level of performance necessary for Mission Success. Desired performance for each capability is represented as equation (7).

$$SoS.P_i = (P_1, P_2, \dots P_n) \tag{7}$$

Each performance (P_i) associated with a capability requires one or more systems to provide a performance (pi) for that capability. When those one or more systems provide the p_i for that C_i, then the P_i is present for that C_i at the SoS level. Each SoS level performance (P_i) has an associated SoS funding F_i and SoS deadline D_i. Just as the SoS performance is allocated to performance (p_i) from one or more systems, the SoS funding (F_i) and SoS deadline (D_i) are also allocated to funding (f_i) and deadline (d_i) to one or more systems. This represented in equations (8), (9), and (10).

$$P_i = \sum_{k=1}^{N} p_k \tag{8}$$

$$F_i = \sum_{k=1}^{N} f_k \tag{9}$$

$$D_i = max_{k=1}^{N}\{d_k\} \tag{10}$$

where,
i = Capability number
n = number of capabilities
N = number of systems

The p_i, f_i, and d_i for each Ci are passed to each system responsible for that C_i by the SoS agent. The p_old_i, f_old_i, and d_old_i are the performance, funding, and deadline values for each C_i used by the SoS agent in the previous negotiation cycle.

The p_gap_i, f_gap_i, and d_gap_i for each C_i are the differences between what the SoS agent requested of the systems and what the systems agreed to provide. These are the delta deadline, delta funding, and delta performance returned by the constituent systems. Mathematically these are represented by equations (11), (12), and (13).

$$p\_gap_i = |p_i - p\_old_i| \tag{11}$$

$$f\_gap_i = |f_i - f\_old_i| \tag{12}$$

$$d\_gap_i = |d_i - d\_old_i| \tag{13}$$

As mentioned above, the first step is the fuzzification of the performance, funding, and deadline gap values. Table 1 shows how the gap values are fuzzified.

Table 4.1. Fuzzification for Gap Values

	None	Low	High	Extreme
Performance Gap	0	< 2	< 7 and > 2	> 7
Funding Gap	0	< 3.5	< 6.5 and > 3.5	> 6.5
Deadline Gap	0	< 20	< 80 and > 20	> 80

The fuzzified values act as indices into the FAM to extract the adjustments to be made to the performance, funding, and deadline. Mathematically this is represented as in equations (14), (15), and (16).

$$p\_gap\_adjust_i = FAM\_Perf(p\_gap\_fuzzy_i, f\_gap\_fuzzy_i, d\_gap\_fuzzy_i) \tag{14}$$

$$f\_gap\_adjust_i = FAM\_Fund(p\_gap\_fuzzy_i, f\_gap\_fuzzy_i, d\_gap\_fuzzy_i) \tag{15}$$

$$d\_gap\_adjust_i = FAM\_Dead(p\_gap\_fuzzy_i, f\_gap\_fuzzy_i, d\_gap\_fuzzy_i) \tag{16}$$

The final step in the FDA is defuzzification of the fuzzy values for performance adjustment, funding adjustment, and deadline adjustment. The FDA uses the centroid method of defuzzification which is represented mathematically in equation.

$$y = \frac{\sum_{i-1}^{slice} x_i * \mu_X(x_i)}{\sum_{i}^{slice} \mu_X(x_i)} \mathrm{y} = \frac{\sum_{\mathrm{i}-1}^{\mathrm{slice}} \mathrm{x_i} * \mu_\mathrm{X}(\mathrm{x_i})}{\sum_{\mathrm{i}}^{\mathrm{slice}} \mu_\mathrm{X}(\mathrm{x_i})} \tag{17}$$

where
slice = the total number of pieces
$\mu_X(x_i)$ = the fuzzy membership function

As can be seen from the above defuzzification equation, defuzzification depends on the fuzzy membership function for each fuzzy variable. Performance gap has the fuzzy values {None, Low, High, Extreme}. The fuzzy membership function for performance gap is shown in Figure 4.4.

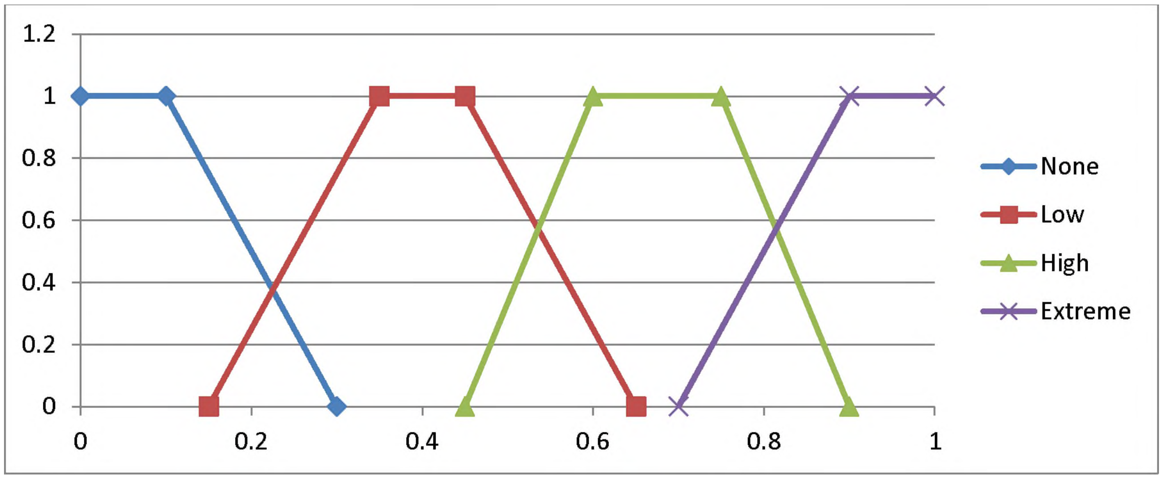


Figure 4.4. Performance Gap Fuzzy Membership Function

Funding gap has the fuzzy values {None, Low, High, Extreme}. The fuzzy membership function for funding gap is shown in Figure 4.5.

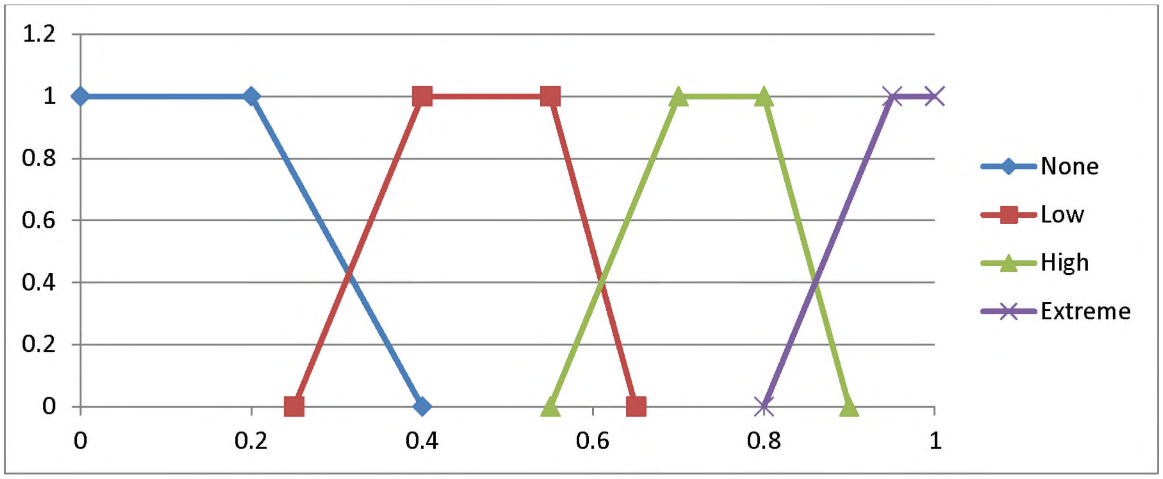


Figure 4.5. Funding Gap Fuzzy Membership Function

Deadline gap has the fuzzy values {None, Low, High, Extreme}. The fuzzy membership function for deadline gap is shown in Figure 4.6.

Performance adjustment has the fuzzy values {Nothing, Decrease, Increase, Increase_Much}. The fuzzy membership function for performance adjustment is shown in Figure 4.7.

Funding adjustment has the fuzzy values {Nothing, Decrease, Increase, Increase_Much}. The fuzzy membership function for funding adjustment is shown in Figure 4.8.

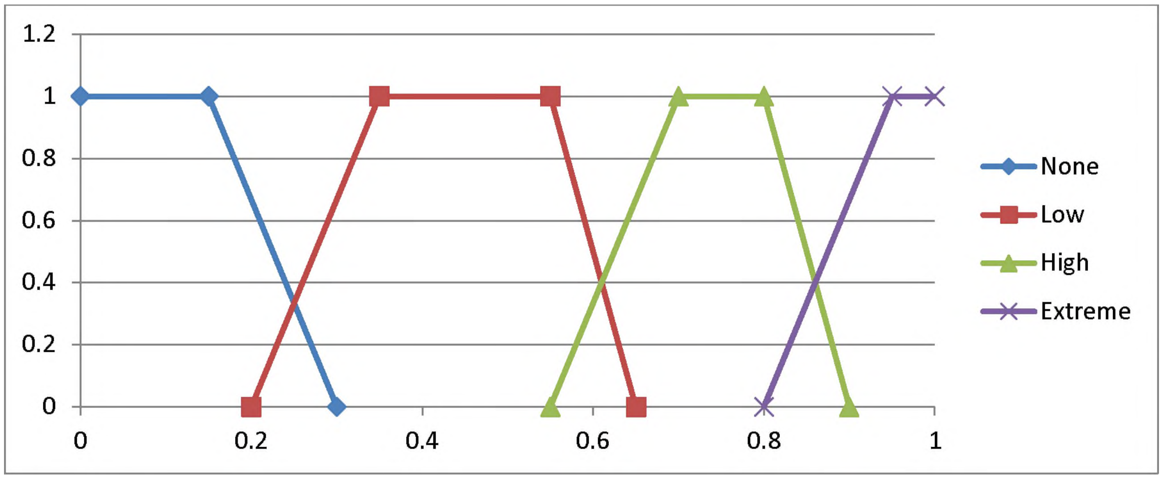


Figure 4.6. Deadline Gap Fuzzy Membership Function

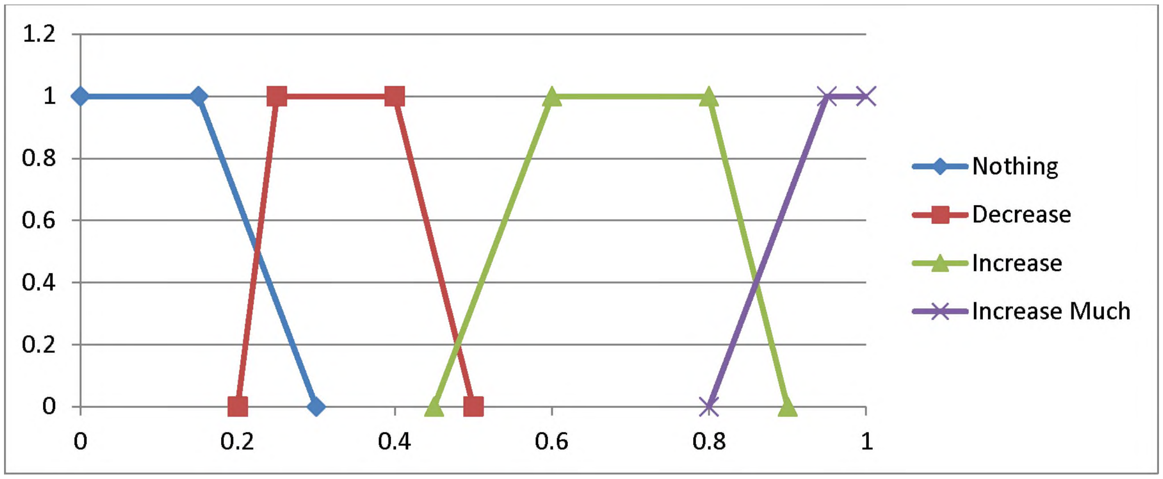


Figure 4.7. Performance Adjustment Fuzzy Membership Function

Deadline adjustment has the fuzzy values {Nothing, Decrease, Increase, Increase_Much}. The fuzzy membership function for deadline adjustment is shown in Figure 4.9.

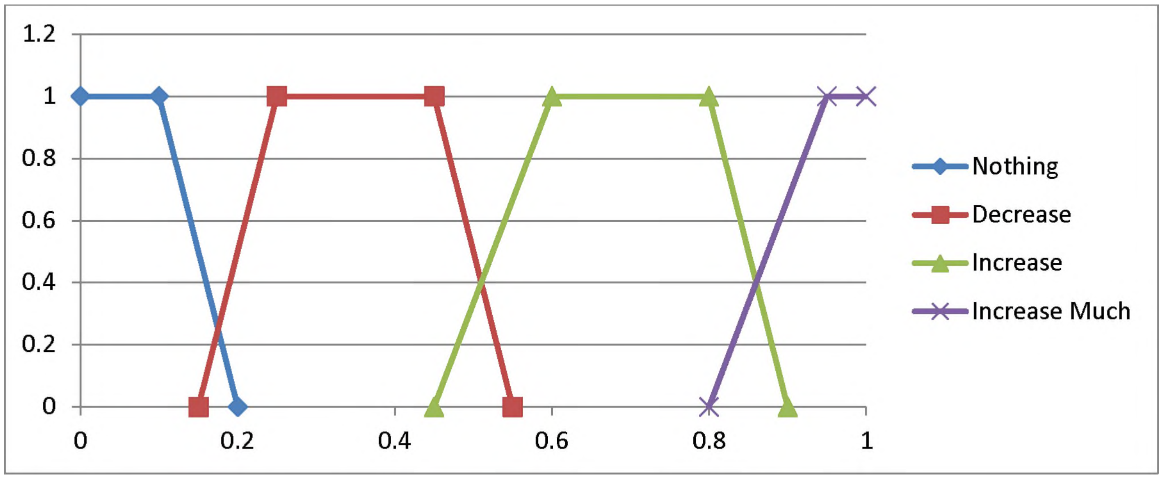


Figure 4.8. Funding Adjustment Fuzzy Membership Function

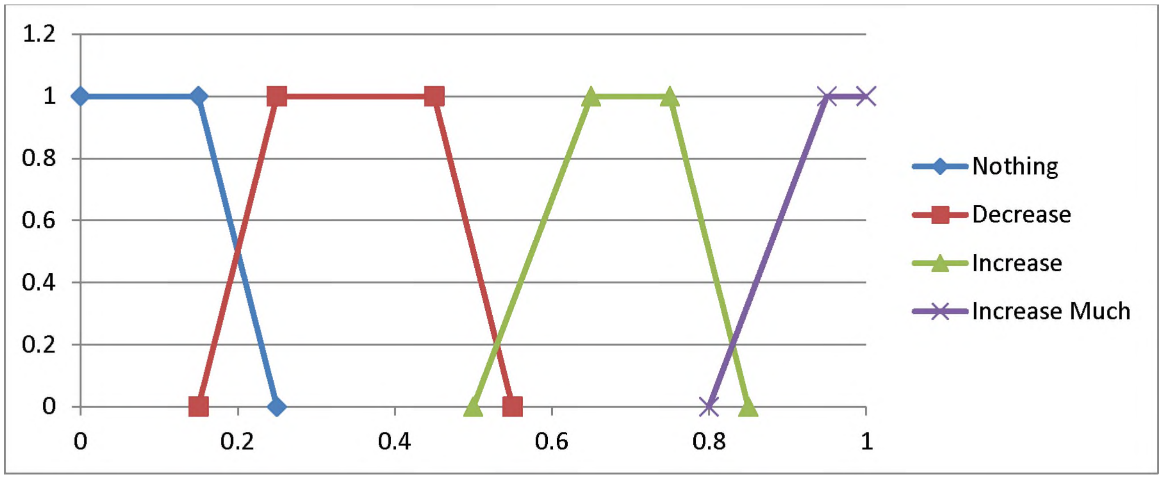


Figure 4.9. Deadline Adjustment Fuzzy Membership Function

The SoS agent structure is illustrated in Figure 4.10. The SoS state transition diagram shows the states of the SoS agent. The SoS State Transition Diagram is provided in Figure 4.11. The SoS has five states: Initialization, Evolve, Plan, Implement, and Cleanup.

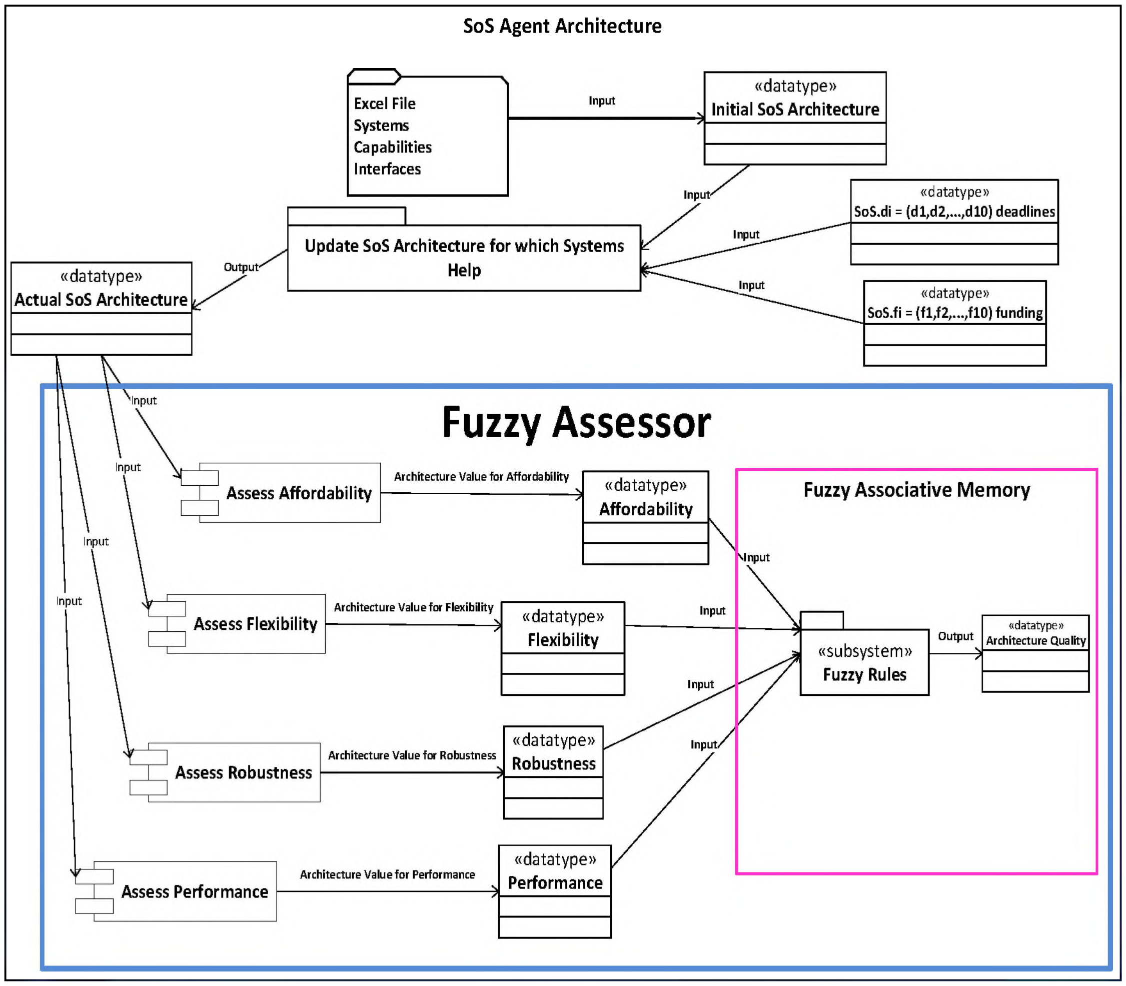


Figure 4.10. SoS Agent Structure

The Initialization state is the first state and where all the variables are initialized. In the Evolve state the SoS determines the SoS architecture based on the DoDAF diagrams. The Plan state sends the request for capabilities to each system and creates the SoS architecture from the responses. The Implement phase determines the feasibility of the SoS architecture and decides whether the architecture will support Mission Success or not. If yes, then the SoS goes into the Evolve state waiting for another set of capabilities from the DoDAF diagrams. If no, then the SoS goes back to the Plan state to request a different set of capabilities from the constituent systems.

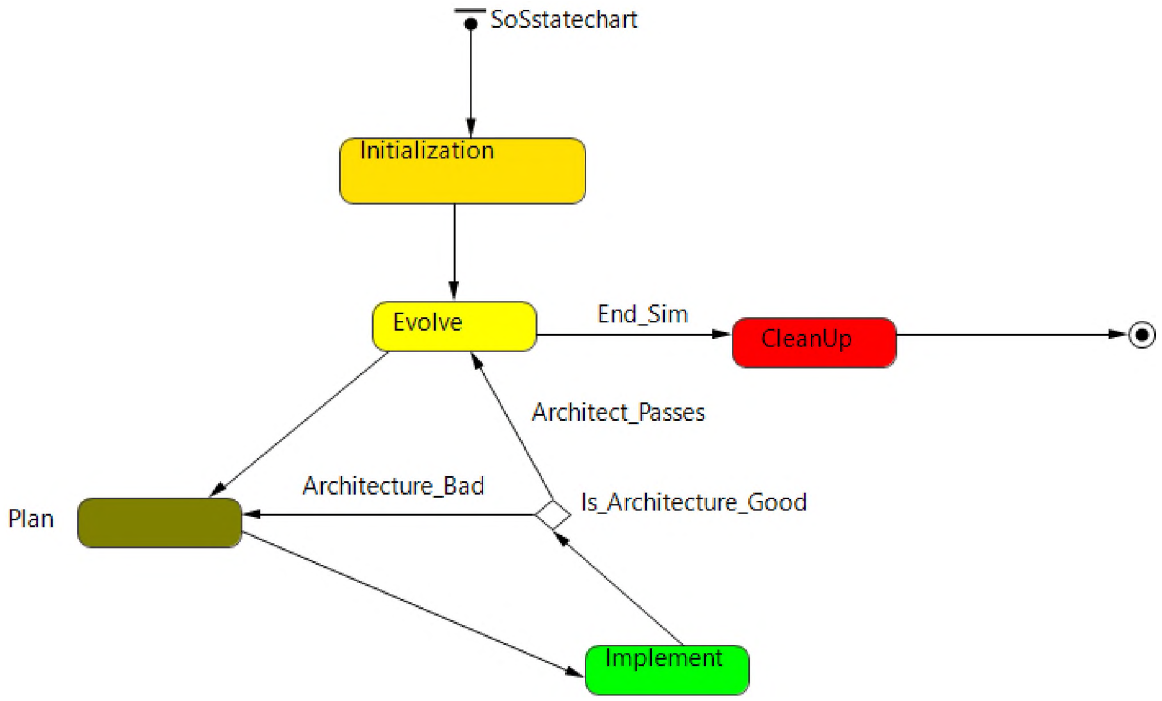


Figure 4.11. SoS Agent State Transition Diagram

The SoS architecture defines the capabilities provided by the constituent systems. The SoS has an initial desired SoS architecture but the final actual SoS architecture depends on what the constituent systems can provide. The SoS architecture assessment provides a qualitative view of the SoS architecture. Because this is a qualitative assessment, a fuzzy assessor is appropriate. The next section describes the fuzzy assessor.

4.1.1. SoS Architecture Assessment. One of the steps in system engineering is architecture development. The architecture development process includes evaluation of the architecture for characteristics of a good architecture. The architecture evaluation is a qualitative assessment like the requirements assessment described in a paper by Acheson [57]. The architecture evaluation uses a fuzzy assessor [16] that is an extension of the requirement assessor [57]. The architecture assessor uses architecture attributes instead of the requirements attributes used in the requirements assessor. The architecture assessor also uses different membership functions as shown in the figures in the next subsection.

One of the architecture characteristics is resilience. Resilience in architecture reflects the resilience of the system. Early detection of low resilience allows time for re-architecting to increase resilience.

The purpose of the SoS is the fielding of capabilities to the warfighter. The set of capabilities of the SoS is the SoS architecture. The SoS starts with the set of capabilities which is a desired SoS architecture. The SoS requests required capabilities from the constituent systems. The architecture assessment is a qualitative assessment using a fuzzy assessor. The resultant architecture is the architecture based on what capabilities and performance the systems agree to provide based on their constraints.

This section describes the SoS architecture assessor used to perform the qualitative architecture evaluation in this research.

For this research, the attributes of affordability, flexibility, robustness, and performance were chosen, and fuzzy membership functions are defined for them. In this research the following definitions for the architecture attributes are used. Affordability is a fuzzy variable and is based on the capabilities acquired versus the amount of fund the

SoS manager must expend. Ideally the SoS manager would like to get the most amount of capabilities for the least amount of money. For this reason, affordability is defined as the ratio of the capabilities acquired over the amount of funding the SoS manager must spend to acquire the capabilities from the individual systems. When the SoS manager does not need to spend very much money to get additional capabilities from the constituent systems, the SoS architecture is considered very affordable. Flexibility is a fuzzy variable and is defined as percentage of systems that participate in the SoS architecture. The more systems that participate the more flexible the SoS architecture would be to include additional future capabilities. Systems that already participate in the SoS would have established relationships with other systems and thus would make incorporation of future capabilities easier. Robustness is a fuzzy variable and is defined as the ability of the SoS to continue to provide the desired capabilities even under adverse circumstances. The SoS capabilities depend on system-to-system interfaces. If one interface fails, then it is possible that a combination of the other interfaces would still provide the required SoS capability. For this reason, the SoS robustness is a function of the number of system-to-system interfaces for those systems that participate in the SoS. Performance is a fuzzy variable and is defined as the overall performance of the SoS. Performance is defined as the aggregation of all the performances from all the capabilities supplied by the constituent systems compared to the aggregation of performances requested by the SoS. This comparison is the ratio of acquired performances over the requested performances.

The fuzzy assessor first determines the affordability, flexibility, robustness, and performance of the SoS architecture. The values for these are fuzzified and then become the fuzzy inputs to the fuzzy associative memory (FAM) that is part of the fuzzy assessor.

The FAM contains the fuzzy rules used to qualitatively evaluate the SoS architecture and produces a SoS architecture quality value that is a fuzzy value. This fuzzy quality value is then defuzzified and is compared to the expected SoS architecture quality value. When the fuzzy quality value reaches the expected quality value the cycle of negotiations stops.

4.1.1.1. Mathematical representation. In the following equations, the variable N is the total number of systems and M is the total number of capabilities at the SoS level.

In the fuzzy assessor, the architecture attributes of affordability, flexibility, robustness, and performance are used to evaluate the SoS architecture. Each of these architecture attributes are crisp values that must be fuzzified. The affordability fuzzy variable has the values of {Expensive, Costly, Reasonable, Very Affordable}. Its fuzzification is shown in Table 2. The total SoS funding is used to determine the value of the affordability variable and is determined by equation (18).

$$Afford(SoS.Arch) = \frac{\sum_{j=1}^{M}\sum_{i=1}^{N} c_j c_i}{Total\ SoS\ Funding} \tag{18}$$

Table 4.2. Fuzzification of Affordability

Fuzzification of Affordability				
Ratio of Capabilities Provided to SoS Funding Spent				
	Expensive	Costly	Reasonable	Very Affordable
Architecture	> 1.5	between 1 and 1.5	between 0.5 and 1	< 0.5

The flexibility fuzzy variable has the values of {Rigid, Pliable, Flexible, Very Flexible}. The SoS flexibility is determined using the aggregate ability of the constituent systems to adapt to new SoS requests as shown in equation (19). The fuzzification of flexibility is shown in Table 3.

$$Flex(SoS.Arch) = \frac{\sum_{i=1}^{N}(S_i == Ability)}{\sum_{i=1}^{N}(S_i)} \qquad (19)$$

Table 4.3. Fuzzification of Flexibility

Fuzzification of Flexibility				
Percentage of Systems Participating in the SoS Architecture				
	Rigid	Pliable	Flexible	Very Flexible
Architecture	< 0.3	between 0.3 and 0.5	between 0.5 and 0.85	> 0.85

Robustness is a fuzzy variable which has the values of {None, Little, Robust, Very Robust}. The number of system-to-system interfaces and the number of systems that cooperate are used to determine the SoS robustness. Mathematically this is given by equation (20).

$$Robust(SoS.Arch) = \left(\sum_{k=1}^{N}\sum_{i=1}^{N} S_{i,k}\right) * \frac{\sum_{i=1}^{N}(S_i == Ability)}{\sum_{i=1}^{N}(S_i)} \qquad (20)$$

where
$S_{i,k}$ represents an interface between system I and system k
$S`_i$ represents system i

Robustness is fuzzified according to information in Table 4.

Table 4.4. Fuzzification of Robustness

Fuzzification of Robustness				
Percentage of System Interfaces Participating in the SoS Architecture				
	None	Little	Robust	Very Robust
Architecture	< 0.3	between 0.3 and 0.5	between 0.5 and 0.85	> 0.85

The performance fuzzy variable has the values of {Poor, Low, Good, Exceptional}. The total SoS performance is a ratio of acquired performance over requested performance and is expressed by equation (21). The fuzzification of performance is shown in Table 5.

$$Perf(SoS.Arch) = \frac{\sum_{j=1}^{M}\sum_{i=1}^{N} p_i(C_j)}{\sum_{j=1}^{M}\sum_{i=1}^{N} p'_i(C_j)} \tag{21}$$

Where p_i represents the acquired performance from each system and p'_I represents the requested performance from each system.

Table 4.5. Fuzzification of Performance

Fuzzification of Performance				
Percentage of System Interfaces Participating in the SoS Architecture				
	Poor	Low	Good	Exceptional
Architecture	< 0.3	between 0.3 and 0.5	between 0.5 and 0.85	> 0.85

The fuzzy assessor is implemented using a FAM where the fuzzy variables of affordability, flexibility, robustness, and performance are indices into the FAM to get the SoS architecture quality. This is represented mathematically in equation (21).

$$SoS.Q = FAM\big(Afford(SoS.Arch), Flex(SoS.Arch), Robust(SoS.Arch), Perf(SoS.Arch)\big) \tag{21}$$

The fuzzy assessor is a function of the SoS architecture that provides the SoS architecture quality as a fuzzy value. The SoS architecture quality has the fuzzy values {Unacceptable, Marginal, Okay, Exceeds} which must be defuzzified. The centroid method of defuzzification is used and is given in equation (22).

$$y = \frac{\sum_{i-1}^{slice} x_i * \mu_X(x_i)}{\sum_i^{slice} \mu_X(x_i)} y = \frac{\sum_{i-1}^{slice} x_i * \mu_X(x_i)}{\sum_i^{slice} \mu_X(x_I)} \tag{22}$$

where
slice = the total number of pieces
$\mu_X(x_i)$ = the fuzzy membership function

The fuzzy membership function for SoS architecture quality is shown in Figure 4.12.

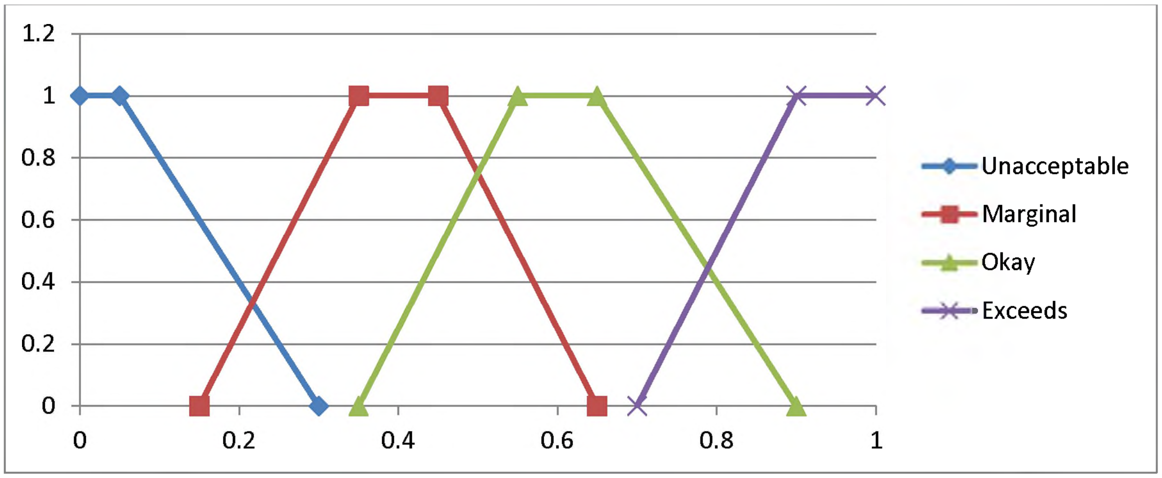


Figure 4.12. SoS Architecture Quality Fuzzy Membership Function

The defuzzification yields a SoS architecture quality value that is a crisp number. The next section describes the structure of the fuzzy assessor.

4.1.1.2. Fuzzy assessor structure. Figure 4.13 shows the structure of the SoS architecture fuzzy assessor. The fuzzy assessor is implemented inside of the agent-based model within the SoS agent. The fuzzy assessor is initialized during the SoS_Initialize state and is used during the Implement_SoS_Architecture state to check the quality of the actual SoS architecture.

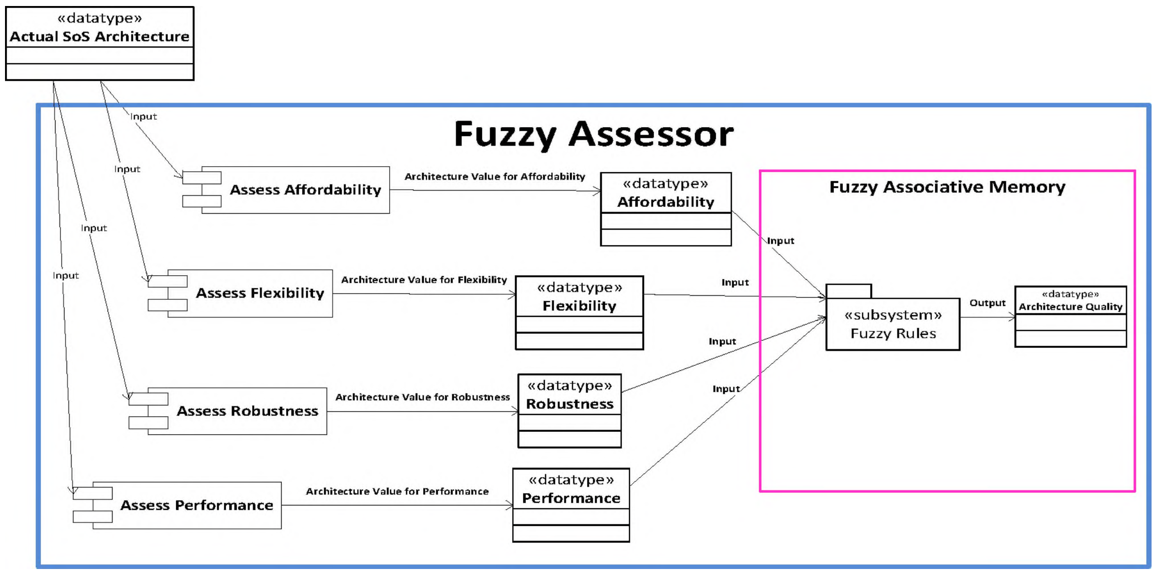


Figure 4.13. Structure of the SoS Fuzzy Assessor

The fuzzy assessor qualitatively evaluates the actual SoS architecture for architecture characteristics of affordability, flexibility, robustness, and performance. Fuzzy values are determined for the affordability, flexibility, robustness, and performance. These fuzzy values are used as indices into the fuzzy associative memory (FAM). The FAM contains the fuzzy rules that are the inference engine for the fuzzy assessor. The output of the FAM is the fuzzy SoS architecture quality value which is then defuzzified to produce a crisp number.

4.1.2. SoS Behavior. The SoS behavior is part of the SoS agent. One part of this research is the SoS behavior exhibited during the plan state when the SoS interfaces with the constituent systems. The SoS must work with the systems to get the system capabilities necessary to overcome the cyber-attack and achieve Mission Success. Each system has its own constraints so there is no guarantee that the system can provide the required capabilities that the SoS needs. This section describes the SoS behavior in interfacing with the constituent systems. This is a kind of negotiation that the SoS performs with each constituent system. For this negotiation Fuzzy Decision Analysis (FDA) is employed. The FDA is described in the next section.

4.1.2.1. Fuzzy decision analysis background. Decision analysis is an applied methodology used to make decisions [58]. One of the criticisms of decision analysis is that the inputs are very often imprecise which results in imprecise outputs [59]. One approach to handle the imprecise inputs is using fuzzy set theory [59]. This approach put forth by Watson assumes that the inputs are imprecise or fuzzy and proposes the use of fuzzy sets to handle this imprecision. Carlsson shows that using fuzzy logic the imprecise inputs still provide meaningful results [60]. Carlsson used ordered weighted averaging to prove that small changes in the inputs result in small changes in the outputs. Carlsson demonstrates how fuzzy decision analysis enhances the corporation decision making of two major Finnish forest industries [60]. The point here is that fuzzy decision analysis is useful when the inputs are imprecise.

Fuzzy decision analysis has been successfully used in multiple domains as part of the decision-making process. In one case, fuzzy decision analysis was used in choosing a configuration management tool [61]. The results showed that it worked well in choosing the best tool.

In another case, fuzzy decision analysis was used in the planning process during a power plant re-licensing activity [62]. In order to be re-licensed, the power plant had to collaborate with multiple stakeholders each seeking different outcomes. Fuzzy decision analysis was chosen because of its ability to handle the large number of stakeholders [62]. In this situation, fuzzy decision analysis was used to negotiate among the various stakeholders to provide a solution that was acceptable to each one.

Fuzzy decision analysis was also used in an automated drug delivery system [63]. In the automated drug delivery system, fuzzy decision analysis determined the patient status and then determined the amount of each of four drugs to dispense into the patient. This drug delivery system was more responsive and more effective at regulating the drugs than existing systems [63]. In this case defuzzification invoked the center-of-gravity method also known as the centroid method.

In the electrical power grid domain, fuzzy decision analysis was used to select the best transmission network [64]. Three objectives were input to the fuzzy decision analysis: investment cost, congestion cost, and amount of load shedding. The fuzzy decision analysis incorporated a linear monotonically decreasing membership function for all three objectives. A min-max was used to compare the fuzzy values to the reference value (desired value). The fuzzy decision analysis was used on a real power grid with results that were superior to other methods.

More recently, fuzzy decision analysis was used in negotiation between the SoS and the constituent systems [56]. In this journal paper, fuzzy decision analysis is used by the SoS during the negotiation with the constituent systems. The SoS requests capabilities from the constituent systems based on the desired SoS architecture. Each system responds with what funding, deadline, and performance is necessary for that system to support the requested capabilities. The systems' response of funding, deadline, and performance is input to the fuzzy decision analysis which uses that input to determine the corresponding action the SoS will take. This same fuzzy decision analysis is implemented in the agent-based model to represent the SoS behavior during negotiation with the constituent systems.

The literature shows that fuzzy decision analysis can handle imprecise inputs and can be effectively used in making decisions. For this reason, fuzzy decision analysis was chosen to represent the SoS behavior in the negotiation with the constituent systems.

4.1.2.2. Fuzzy decision analysis implementation. As mentioned in the previous section, fuzzy decision analysis has been used in situations where there is uncertainty in the inputs and there are many objectives. Fuzzy decision analysis has provided good results in these situations.

Figure 4.14 shows the fuzzy decision analysis process. For each capability, the SoS receives values for funding, performance, and deadline from the constituent systems These values for funding, performance, and deadline are crisp values and are input into the fuzzy negotiation model. The fuzzy negotiation model (FNM) is part of the fuzzy decision analysis that includes the fuzzy inference engine in the form of a fuzzy associative memory (FAM). The first step in the fuzzy negotiation model is the fuzzification of the funding, performance, and deadline values. Fuzzification transforms the crisp values for funding, performance, and deadline into fuzzy values. The weights are already the fuzzy values of {None, Low, High, Heavy}.

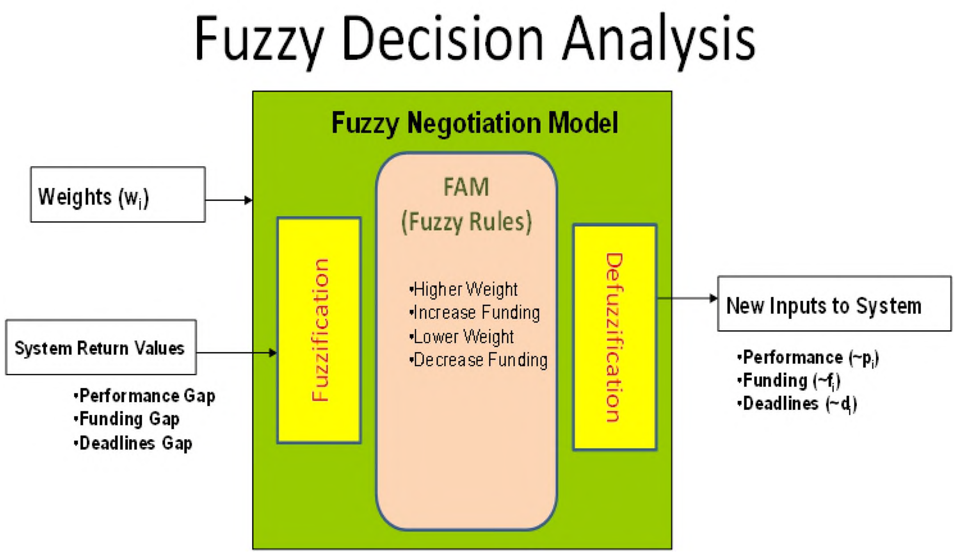


Figure 4.14. Fuzzy Decision Analysis Process

The FNM takes as input the performance gap, funding gap, deadline gap, and the weight assigned to each capability. The FNM outputs the adjustment to performance, adjustment to funding, and adjustment to deadline which are used to calculate new performance, funding, and deadline values that are sent to the systems.

Once the funding, performance, and deadline values are fuzzified, the fuzzy values are used in the fuzzy associative memory (FAM) along with the fuzzy weight value to extract what actions the SoS should take in the negotiation with the constituent systems. These actions are fuzzy values that must then be defuzzified into crisp values that are adjustments to the funding, performance, and deadline for the next negotiation cycle. The FAM contains the fuzzy inference engine with the fuzzy rules that determines how the funding, performance, and deadline should be adjusted during the negotiation. The FAM is implemented as a multidimensional array (Inputs, Rules, Operation) whose contents are the Outputs. Equation (23) shows the inputs and equation (24) shows the outputs.

Inputs=(Performance_Gap, Funding_Gap, Deadline_Gap, Weight) (23)

Outputs=(Funding_Adjustment, Deadline_Adjustment, Performance_Adjustment) (24)

Rules are part of the FAM. The rules are:

- Performance_Gap=None
- Funding_Gap=Extreme
- Deadline_Gap=None
- Weight=None
- Operations=(AND, OR, NEGATION)

The FAM is implemented in an Excel spreadsheet with each Tab/Worksheet being a different Operation. Table 6 is an excerpt from the FAM for illustration purposes.

The fuzzy decision analysis with the fuzzy negotiation model is described in the paper published in the International Journal of Soft Computing and Software Engineering [56].

The overall approach in the FNM is to move the funding away from the capabilities with the lower weights toward the capabilities with the higher weights.

Table 4.6. Fuzzy Associative Memory (FAM)

Inputs				**Outputs**		
Performance Gap	**Funding Gap**	**Deadline Gap**	**Weight**	**Funding**	**Deadline**	**Performance**
None	None	None	None	Decrease	Do_Nothing	Do_Nothing
None	None	None	Low	Do_Nothing	Do_Nothing	Do_Nothing
None	None	None	High	Do_Nothing	Do_Nothing	Do_Nothing
None	None	None	Heavy	Do_Nothing	Do_Nothing	Do_Nothing
None	Low	Extreme	High	Increase_Much	Increase	Do_Nothing
None	Low	Extreme	Heavy	Increase_Much	Increase	Decrease
Low	High	Extreme	High	Do_Nothing	Do_Nothing	Do_Nothing
Low	High	Extreme	Heavy	Do_Nothing	Do_Nothing	Do_Nothing
High	Low	Extreme	High	Increase_Much	Increase	Do_Nothing
High	Low	Extreme	Heavy	Increase_Much	Increase_Much	Decrease
Extreme	High	Low	Low	Do_Nothing	Do_Nothing	Do_Nothing
Extreme	High	Low	High	Do_Nothing	Do_Nothing	Do_Nothing

Those capabilities with the lower weights would receive an increase in deadline which is the equivalent of extending the schedule. This is typical in DoD programs with limited funding for the current fiscal year but that do not cause any adverse effects on the warfighter if the schedule is extended. The capabilities with the higher weights are those that are most important or critical to the SoS and whose deadlines cannot be extended without serious impact to the warfighter.

In this approach, those capabilities that are lower weights and whose delta funding, delta performance, and delta deadline are "None" would lose some of their funding in favor of using that funding on the capabilities with the higher weights. A fuzzy value of "None" means that the constituent system accepts the delta funding, delta performance, and delta deadline requested by the SoS. Those capabilities with higher weights and whose delta funding, delta performance, and delta deadline are “None” would lose a little funding in favor of those capabilities whose delta funding, delta performance, and delta deadline are “Extreme”. A fuzzy value of “Extreme” means that there is a large gap between what the SoS requested and what the constituent system can provide. Thus, the fuzzy inference rules move the funding, performance, and deadlines toward the middle ground.

4.2. SYSTEM AGENT

This section describes the system agent. The system agent is instantiated multiple times once for each system in the simulation. The system agent sends a response message to the SoS indicating whether it can provide capabilities to the SoS. Based on what the SoS receives from the systems, it updates the SoS architecture. Figure 4.15 shows the system agent interface to the SoS.

Each system can be represented as:

$$S = \{A, B, I\} \tag{25}$$

where
S is the system
A is the set of attributes = $\{a_1, a_2, \ldots a_j\}$
B is the behavior of the system
I is the set of interfaces to S = $\{i_1, i_2, \ldots i_j\}$

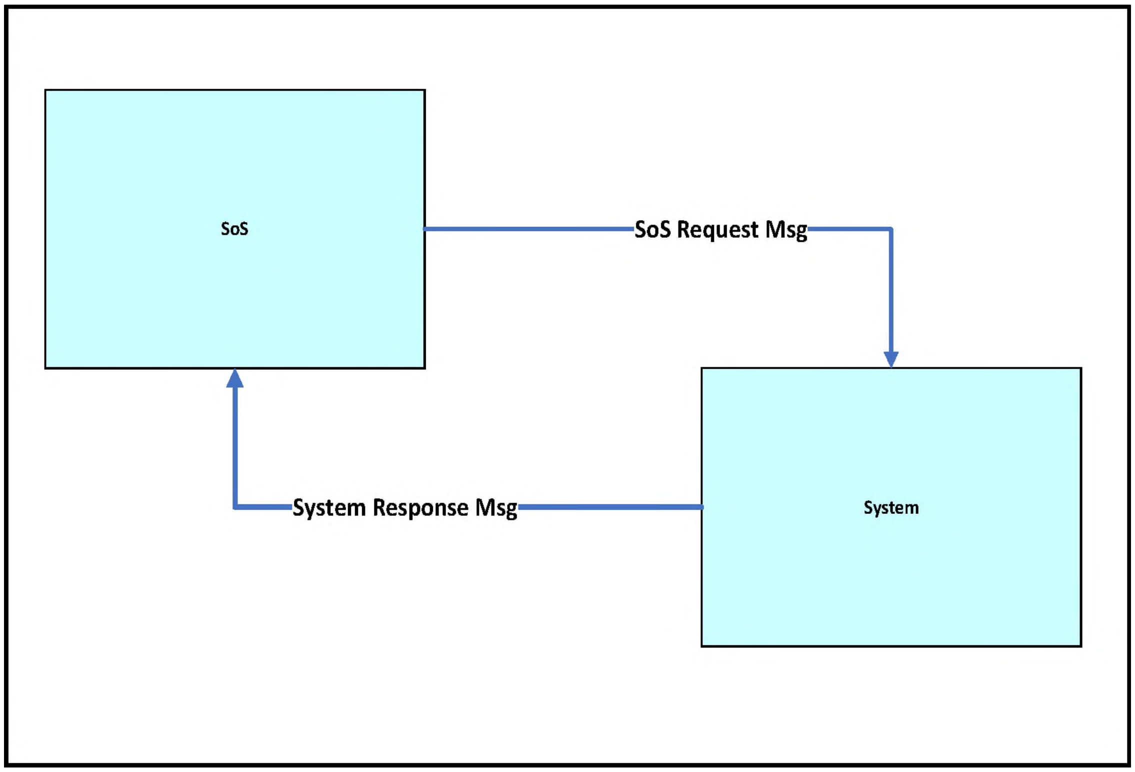


Figure 4.15. System Interface to the SoS

The systems provide capabilities to the SoS. Within the system agent are operations to get the capabilities and the level of help that the system can provide. Once the system receives the request from the SoS, the system determines whether it can provide those capabilities. Which capabilities the system can provide depends on the system state and the system model used. The SARM is designed so that an external model developed in another tool such as MATLAB could be used. Or the system model could be embedded in the system agent. For this research, the system agent uses a fuzzy decision system to determine the level of support the system can provide to the SoS. The fuzzy inference engine (FIE) is part of the fuzzy decision system and takes fuzzy inputs and then outputs the results to be sent to the SoS.The FIE takes the following inputs:

- System constraints (ability to evolve and change)
- System resources
- System funding

The FIE outputs a cooperation level that the system can provide to the SoS. The cooperation is {None, Little, Cooperate, High}. Figure 4.16 shows the system agent structure.

After the system determines the cooperation level, the System Response Message is sent to the SoS. The System Response Message contains the cooperation indication. If the system cooperation level is None or little, then the system includes suggestion for funding, deadline, and performance in the response message. The SoS evaluates the SoS architecture using these system funding, deadline, and performance values using the Fuzzy Architecture Assessor (FAA). The interface between the SoS and the constituent systems reflects the ability of the constituent systems to support the Mission as requested by the SoS. The system agent behavior and can be modeled in multiple ways using a probability function, Monte Carlo analysis, or FDA. In this research an FDA and a probability function were both used with similar results.

Figure 4.17 illustrates the System State Transition Diagram. The System agent has three states: Maybe, Helping, and NoHelping. The Maybe state is the initial state and where the system stays until it receives a request from the SoS for required capabilities. Once the request is received the system uses a FAM to determine whether it can support the SoS request. An excerpt of the system FAM is included in the Appendix.

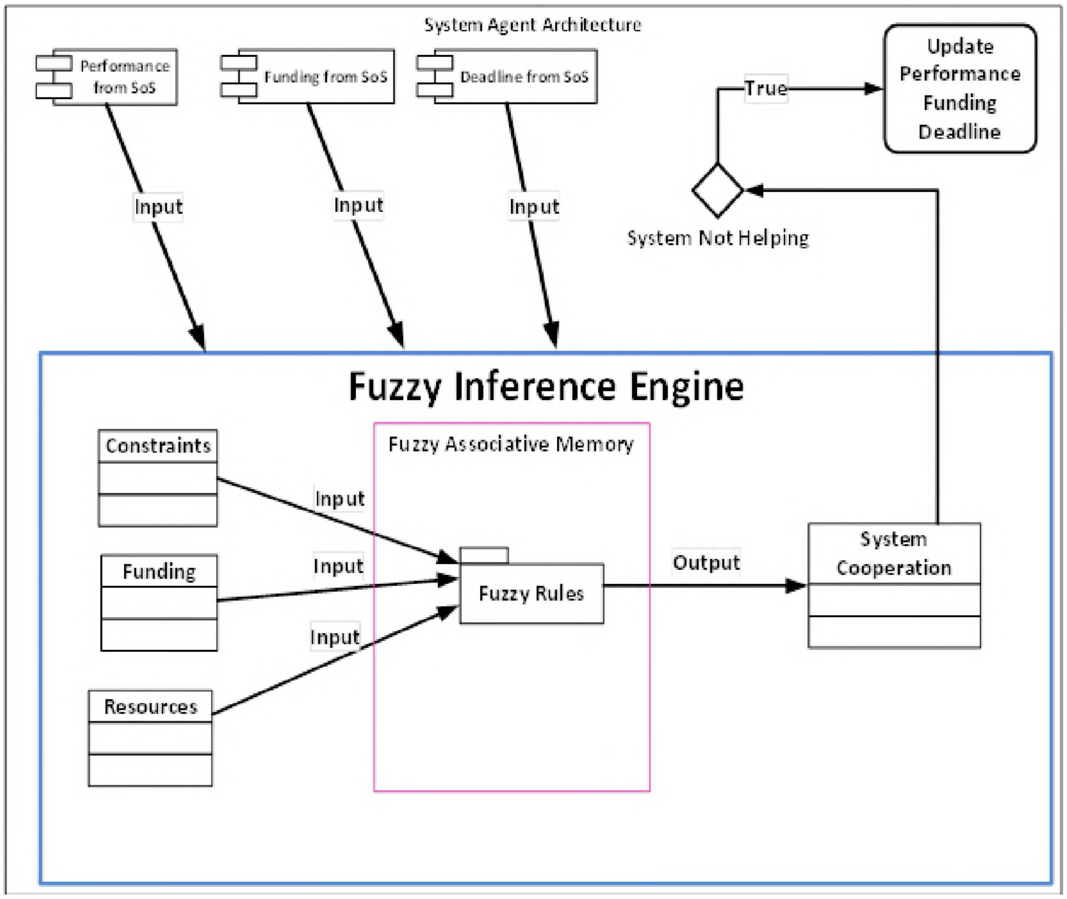


Figure 4.16. System Agent Structure

After the decision is made, the system exits the Maybe state and the system moves to either the Helping or the NotHelping state. In both the Helping and NotHelping states the system sends the response to the SoS indicating whether the system can provide the requested capabilities. The system then returns to the Maybe state.

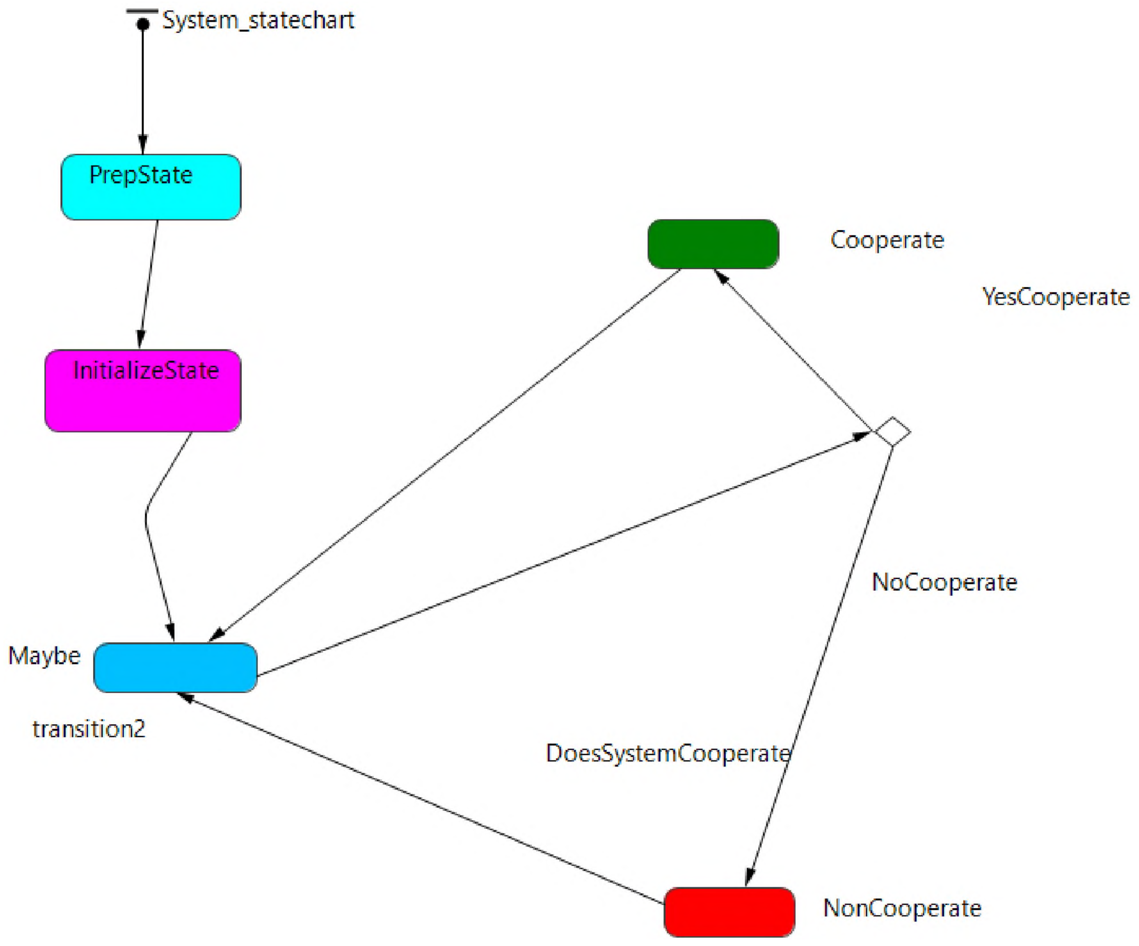


Figure 4.17. System Agent State Transition Diagram

4.3. THREAT AGENT

This section describes the threat agent. The threat agent represents the cyber-attack on the SoS. The threat agent is instantiated multiple times once for each threat in the simulation. The threat agent can attack a system or a system interface. Figure 4.18 shows this interaction.

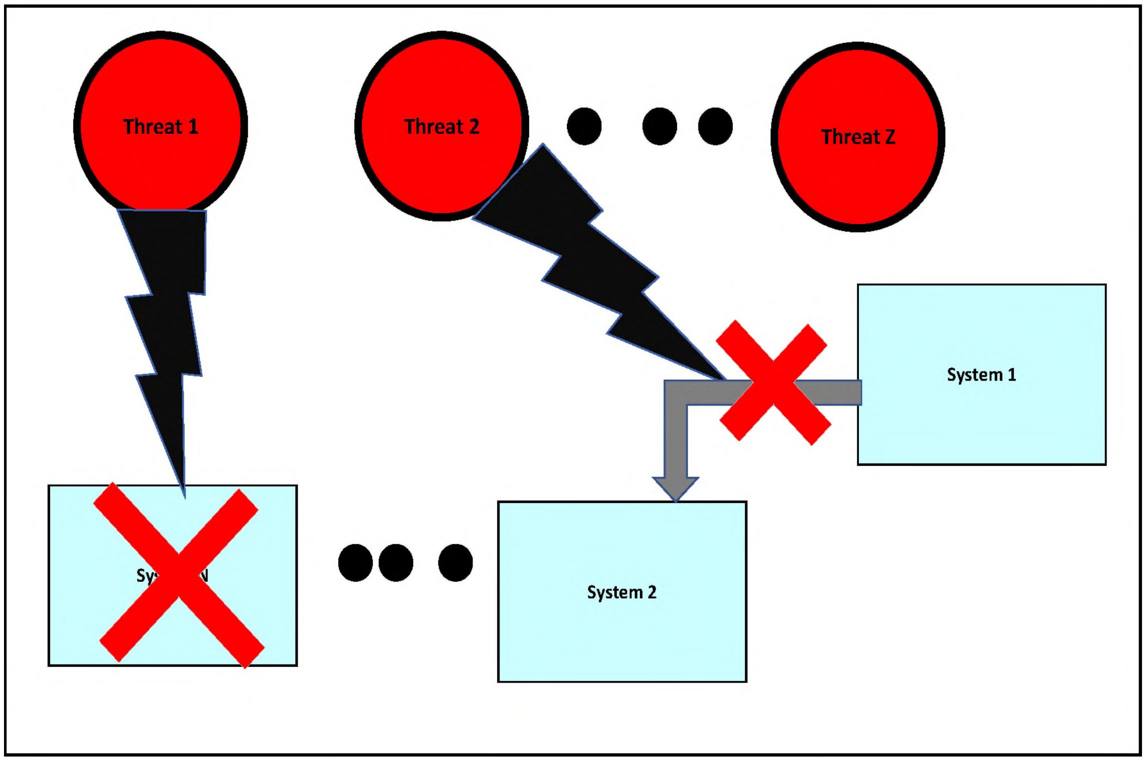


Figure 4.18. Interaction of the Threats with the Systems

Figure 4.19 illustrates the Threat State Transition Diagram. The Threat agent has three states: Thinking, Kill_Interface, and Kill_System. The Thinking state is the initial state and where the Threat determines whether to attack a system interface or a system. The Threat then exits the Thinking state and moves to the Kill_Interface or Kill_System state. In the Kill_Interface state the Threat destroys one of the system to system interfaces. In the Kill_System state the Threat destroys a system.

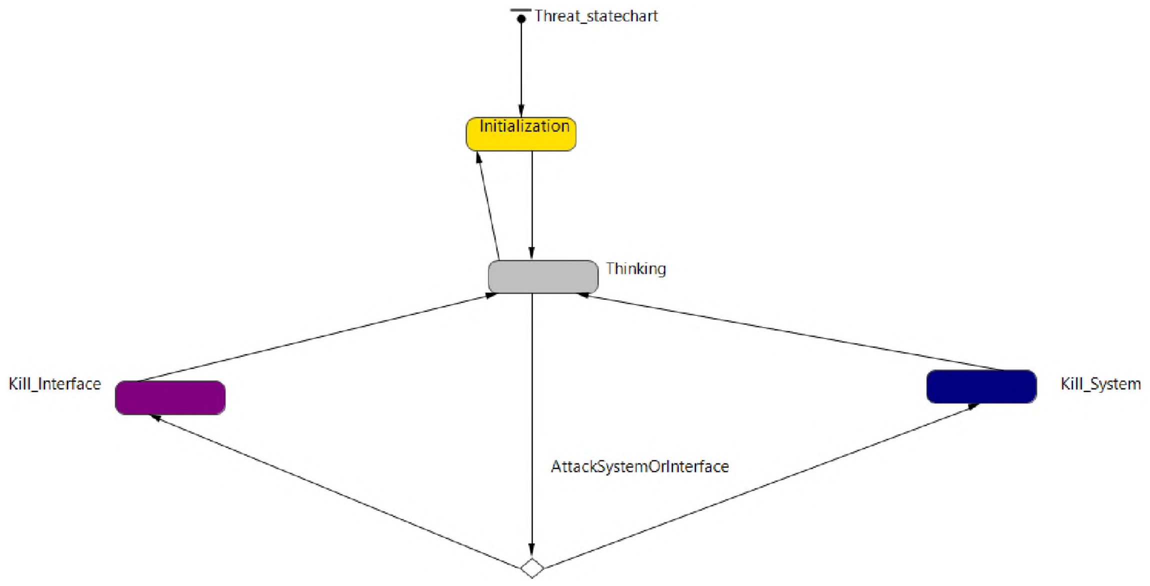


Figure 4.19. Threat Agent State Transition Diagram

Threats are in the Environment and Threats affect the weights (wi) associated with each capability (Ci). Depending on the Threats some Capabilities would be more important than other Capabilities and thus the associated weights for those capabilities would be higher.

$$w_i \propto Threats \tag{26}$$

A Fuzzy Inference Engine implemented as a FAM will use the Threats and National Priorities to determine the weights (wi) for each capability (Ci). FAM will contain the Fuzzy Inference Rules.

Environmental influences are represented as the variables:

- Th = Threats
- NP = National Priorities

Rules for NP and Th are:

1. If no NP and no Th then there is no adjustment to the weights.
2. User Inputs for NP and Th will adjust the weights (wi) for each capability to reflect the influence of NP and Th on the SoS behavior.

5. RESULTS

This section contains the results of this research. The results show that the Mission Resiliency can be measured using a model to represent the SoS and the systems that are key to the Mission. Each system represents a part of the Mission Thread and is necessary for Mission Success. The loss of a system or interface represents a cyber-attack resulting in a degraded state that must still complete the Mission. Mission Resiliency can be considered as how well the SoS can still complete the Mission after a cyber-attack. The SARM is the proof of concept for the approach to Mission Resilience described in this dissertation.

Implementation of the SARM is like the implementation of the SoS architecture development model for the Infantry Immersion Training System (IITS) domain [65]. The IITS provides a specific Mission and SoS to use in the SARM. The SARM was executed using the IITS SoS which consists of the soldier's gear, the monitoring equipment, the urban environment, and the command center. The command center is the SoS and the soldier's gear, the monitoring equipment, and the urban environment are systems 1, 2, and 3, respectively. The simulation was executed using just one threat. The SARM inputs for this simulation are identified in Table 7. The simulation results are shown in Figure 5.1. The results show the number of cycles for each epoch and a general decrease in the number of cycles. This implies an increase in resilience since less cycles means the SoS is faster at adjusting the architecture after a cyber-attack.

The SARM was also executed using a generic SoS with 20 constituent systems and 10 threats. The inputs for the generic simulation are identified in Table 8.

Table 5.1. SARM Inputs for IITS Simulation

Systems	Capabilities	Interfaces
1	1	2
2	1	2
3	2	3

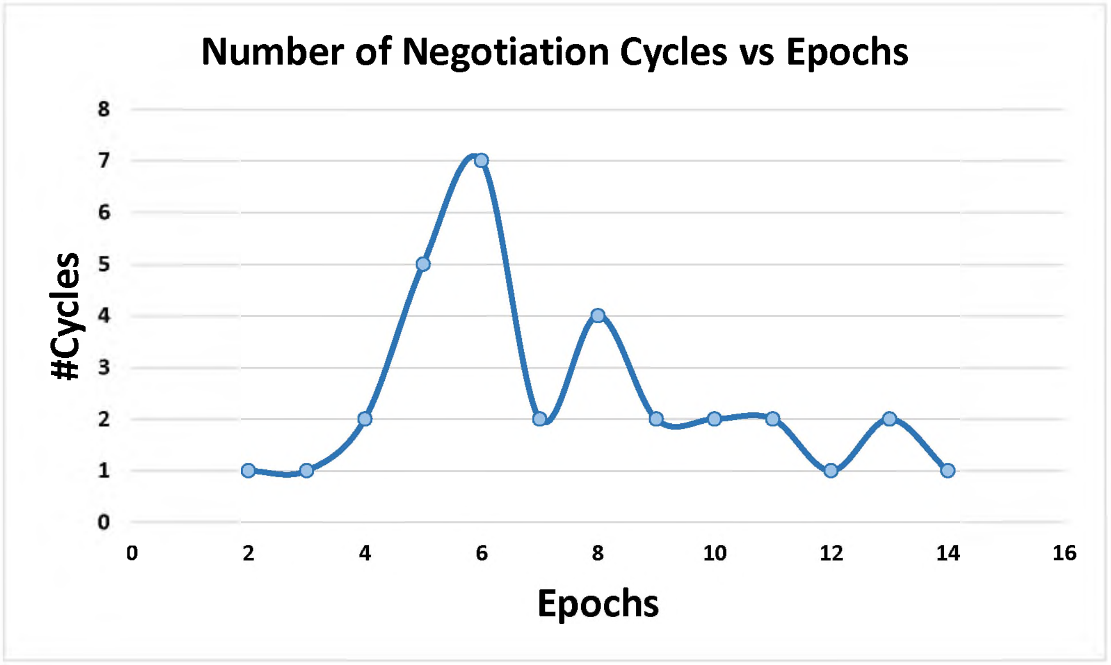


Figure 5.1. SARM IITS Simulation Results

The SARM generic simulation results are shown in Figure 5.2. The generic simulation shows the resilience for a SoS with the number of systems, capabilities, and

Table 5.2. SARM Inputs for Generic Simulation

Systems	Capabilities	Interfaces
1	1	2
2	1	1
3	2	1
4	2	2
5	1	3
6	2	3
7	2	3
8	2	3
9	2	1
10	2	1
11	1	1
12	1	2
13	2	3
14	2	2
15	1	1
16	2	1
17	1	2
18	2	3
19	1	1
20	2	1

interfaces shown in Table 8. In this case, the results again show the number of cycles per epoch which generally decrease as the number of epochs increases.

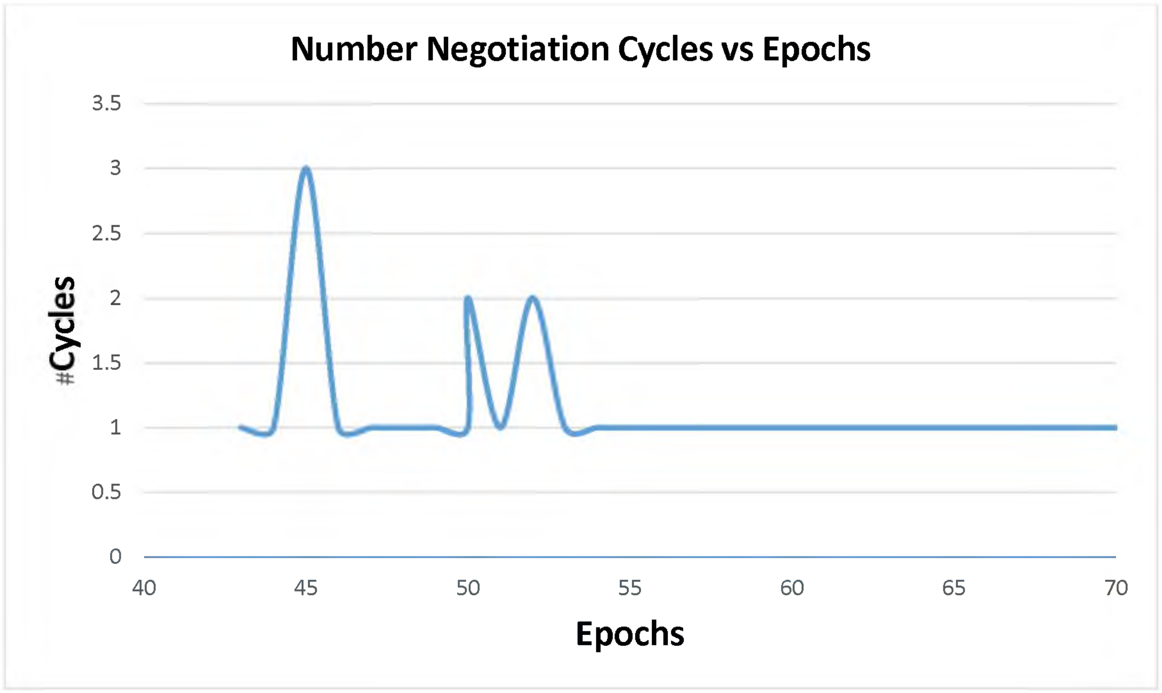
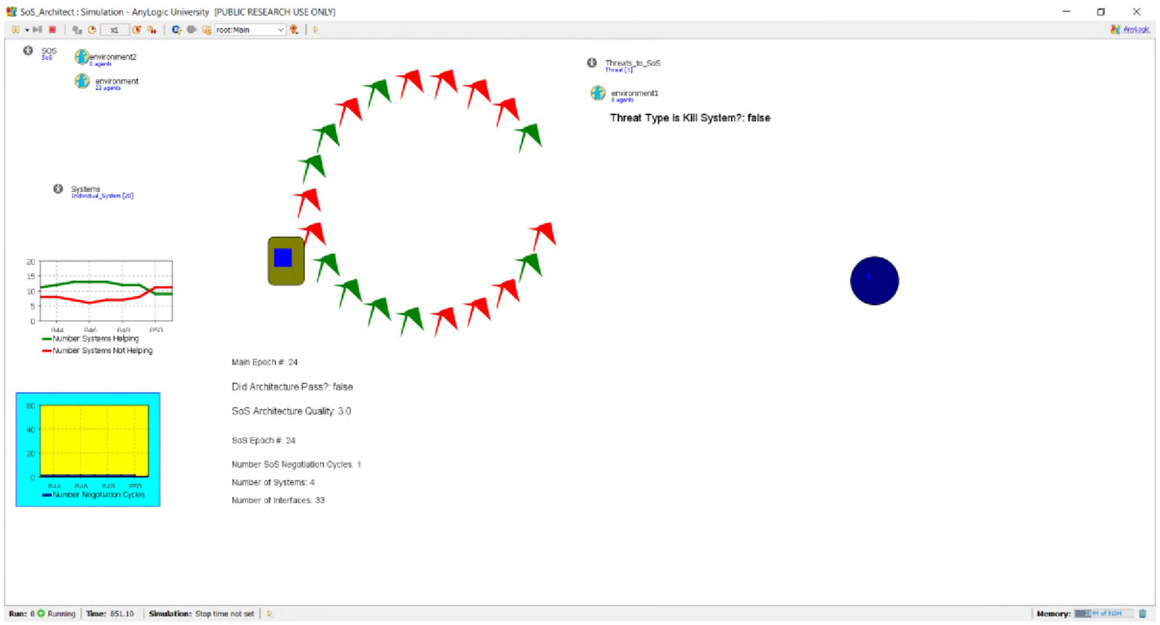


Figure 5.2. SARM Generic Simulation Results

The SARM was executed using a probability for system cooperation instead of the FAM. The results using the probability were like the results using the FAM.

The SARM simulation has the displays shown in Figure 5.3 and Figure 5.4. The display shows the plot of number of systems that agree to the SoS request as well as the number of systems that are not able to support the SoS request. This plot shows how the greater the number of systems that are not able to help the larger the number of negotiation cycles and the lower the resilience capacity.

The displays in Figure 5.3 and Figure 5.4 also show a plot of the number of negotiation cycles over time. This plot and the graph directly above show how the number of systems not supporting the SoS request impacts the number of negotiation cycles. The number of negotiation cycles is inversely proportional to the resilience.



Threat Type is Kill System?: false

Main Epoch #: 24

Did Architecture Pass?: false

SoS Architecture Quality: 3.0

SoS Epoch #: 24

Number SoS Negotiation Cycles: 1

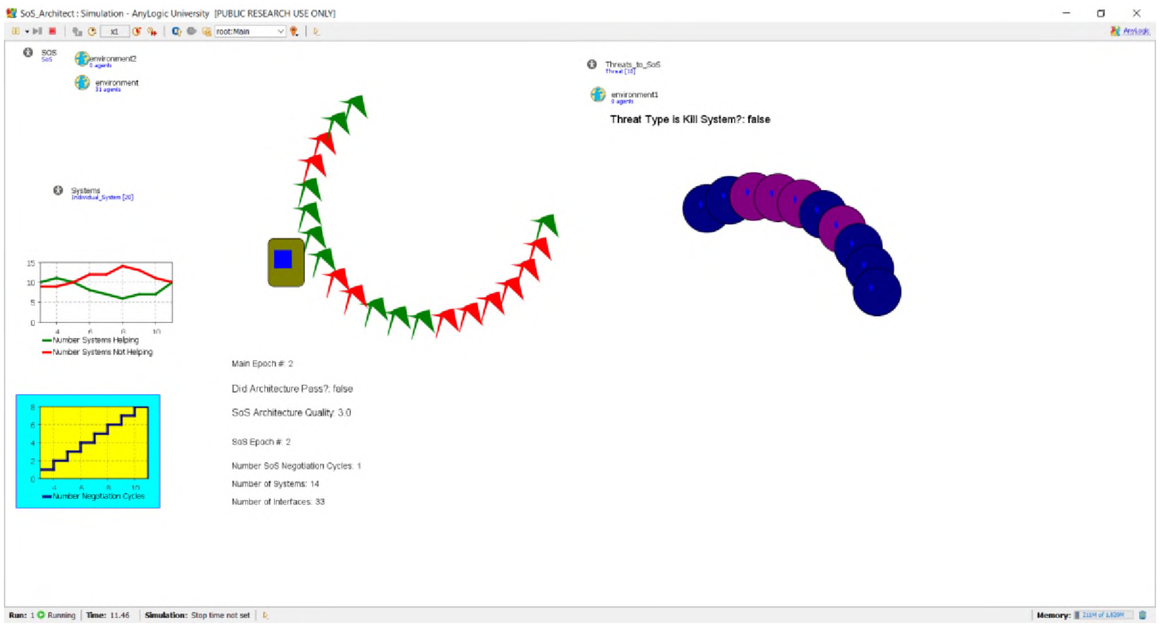
Number of Systems: 4

Number of Interfaces: 33

Run: 0 Running | Time: 851.10 | Simulation: Stop time not set | Memory:

Figure 5.3. Model Simulation Display

The following section describes the conclusions.



Threat Type is Kill System?: false

Main Epoch #: 2

Did Architecture Pass?: false

SoS Architecture Quality: 3.0

SoS Epoch #: 2

Number SoS Negotiation Cycles: 1

Number of Systems: 14

Number of Interfaces: 33

Run: 1 Running | Time: 11.46 | Simulation: Stop time not set

Figure 5.4. Simulation with Multiple Threats

6. CONCLUSIONS

This section contains the conclusions drawn from the results.

6.1. FULFILLED RESEARCH OBJECTIVES

The research objective of this effort was to develop an approach to Mission Resiliency and a prototype model as a proof of concept. The SoS architecture resiliency model (SARM) is the prototype model developed during this research. The SARM does provide a proof of concept to the approach to resiliency. The results in the previous section support this approach to Mission Resilience.

6.2. CONTRIBUTIONS FROM THIS RESEARCH

To summarize, the contributions from this research are:

- Method to evaluate Enterprise or SoS architecture resilience
- SARM
- FAA
- FDA and FNM representing the SoS behavior
- FAM that contains fuzzy rules which can be altered for different scenarios

6.3. CONCLUSIONS BASED ON THE RESULTS

The following conclusions can be derived from the results. The results show that the more systems that are not able to support the SoS request the less resilient the system and the more likely that the Mission will fail. As the simulation executes the on-screen graphs show the relationship between constituent system support and completing the Mission. The graphs show the number of negotiation cycles increasing as fewer systems support the SoS capabilities meaning the resilience capacity is decreasing.

As mentioned earlier, the objective of this research was to develop an approach to resiliency assessment based on mission success. The literature review described earlier research in resiliency. The literature review showed that earlier resiliency research was focused on specific domains and not on the Mission. The research in this dissertation focuses on Mission Resilience thus filling a gap in the literature.

7. FUTURE WORK

This research focuses on Mission Resiliency where the Mission is the focus of the assessment. The approach uses the SoS architecture which is a key element of the Mission. This research provides the model and approach to do the assessment using a simple context. Future work would include the expansion of the model and assessment to allow for additional threats and systems in the model.

The SARM was developed to address one specific Mission. This was necessary to lay the foundation for more complex scenarios. Another area for future work is the addition of multiple Missions. This would raise the complexity of the SARM exponentially and so is more appropriate for future work.

The SARM has an interface to the SoS architecture model that contains the DoDAF architecture views that describe the SoS architecture. This interface is not automatic and requires some manual assistance. Future work could include streamlining this interface and making it more automatic and robust.

The SARM uses a FAM in the system agent to determine whether the system can support the SoS request. Another approach would be to also use an FDA to determine how to adjust the funding, deadline, and performance sent back to the SoS. The SARM can use other models to represent the system behavior. In other papers [56], the SoS development model used models developed in Matlab to represent the system behavior. The same can be done for SARM.

8. DEFINITIONS AND ASSUMPTIONS

This section provides definitions and terminology that will assist the reader in understanding the rest of the document. Some of the explanations in the following sections are using the terminology and definitions presented herein. An acronym list is also included here.

8.1. DEFINITIONS

The following definitions are used within this document:

- Agent – is the independent process that is unique and can exhibit individualized behavior.
- Agent-Based Model – model whose basic elements are agents.
- Model – is a representation of an entity [(Dainth, 2012)].
- SoS Manager – is the person or organization responsible for the SoS development.
- Resilience Capacity – measure of resilience in terms of the SoS ability to successfully complete the Mission despite a cyber-attack.

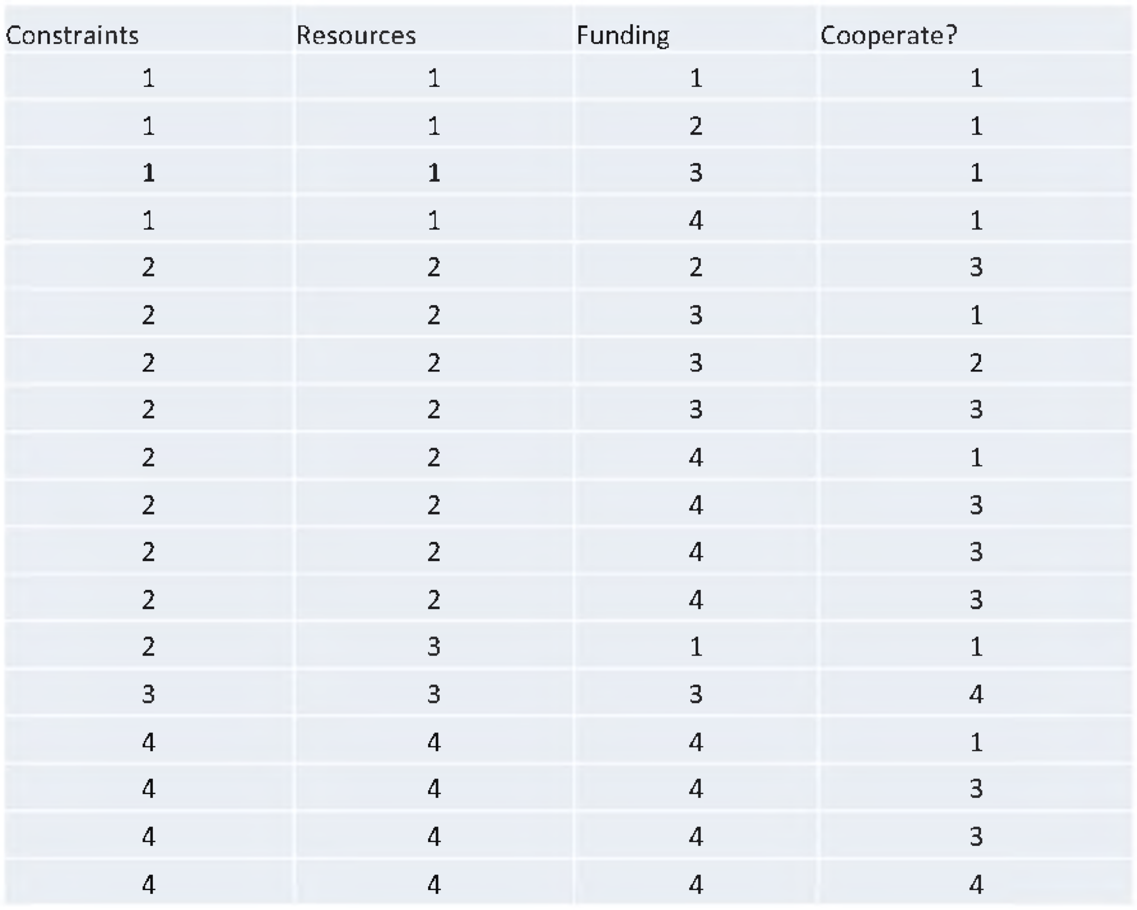
8.2. ASSUMPTIONS

This section lists the assumptions made during this research. In order to complete this research in a timely manner, certain assumptions were made to focus the scope of this effort. In this regard, the following assumptions were made:

- No system-to-system interactions
- Focused on DoD Mission and SoS
- Systems defined by their capabilities and interfaces
- SoS architecture quality results in Mission Success
- Only one Mission per simulation

APPENDIX

Table A- 1. System FAM



Constraints	Resources	Funding	Cooperate?
1	1	1	1
1	1	2	1
1	1	3	1
1	1	4	1
2	2	2	3
2	2	3	1
2	2	3	2
2	2	3	3
2	2	4	1
2	2	4	3
2	2	4	3
2	2	4	3
2	3	1	1
3	3	3	4
4	4	4	1
4	4	4	3
4	4	4	3
4	4	4	4

BIBLIOGRAPHY

1. Sanger, David E., “Hillary Clinton’s Email Was Probably Hacked, Experts Say”, NY Times, July 6, 2016, https://www.nytimes.com/2016/07/07/us/hillary-clintons-email-was-probably-hacked-experts-say.html.

2. BBC News, “Clinton campaign 'hacked' along with other Democratic groups”, July 29, 2016, https://www.bbc.com/news/election-us-2016-36927523.

3. Goel, Vindu (December 14, 2016). "Yahoo Says 1 Billion User Accounts Were Hacked". The New York Times. Retrieved December 14, 2016. https://www.nytimes.com/2016/12/14/technology/yahoo-hack.html.

4. Perlroth, Nicole (September 22, 2016). "Yahoo Says Hackers Stole Data on 500 Million Users in 2014". The New York Times. Retrieved September 22, 2016. https://www.nytimes.com/2016/09/23/technology/yahoo-hackers.html.

5. Goel, Vindu (March 1, 2017). "Yahoo's Top Lawyer Resigns and C.E.O. Marissa Mayer Loses Bonus in Wake of Hack". The New York Times. Retrieved March 15, 2017. https://www.nytimes.com/2017/03/01/technology/yahoo-hack-lawyer-resigns-ceo-bonus.html.

6. Lyle, Amaani (2016). “‘Not If, But When’: NSA Official Discusses Importance of Cyber Vigilance”. DOD News October 20, 2016. Accessed November 7, 2020. https://www.defense.gov/Explore/News/Article/Article/980031/not-if-but-when-nsa-official-discusses-importance-of-cyber-vigilance/.

7. White, Stephanie M.,” Modeling a System of Systems to Analyze Requirements,” *IEEE Systems Conference*, Vancouver, Canada, March 23-26, 2009.

8. Holt, J.; Perry, S.; Brownsword, M.; Cancila, D.; Hallerstede, S.; Hansen, F.O., "Model-based requirements engineering for system of systems," System of Systems Engineering (SoSE), 2012 7th International Conference on , vol., no., pp.561,566, 16-19 July 2012.

9. Long Li; Yajie Dou; Bingfeng Ge; Kewei Yang; Yingwu Chen, "Executable System-of-Systems architecting based on DoDAF meta-model," System of Systems Engineering (SoSE), 2012 7th International Conference on , vol., no., pp.362,367, 16-19 July 2012.

10. Baldwin, W.C. and B. Sauser, Modeling the Characteristics of System of Systems. 2009.

11. Bowen, R.; Sahin, F., "A System of Systems approach to model an Artificial Immune System using Discrete Event Specification," System of Systems Engineering, 2009. SoSE 2009. IEEE International Conference on , vol., no., pp.1,6, May 30 2009-June 3 2009.

12. Michael, J.B.; Man-Tak Shing; Miklaski, M.H.; Babbitt, J.D., "Modeling and simulation of system-of-systems timing constraints with UML-RT and OMNeT++," Rapid System Prototyping, 2004. Proceedings. 15th IEEE International Workshop on , vol., no., pp.202,209, 28-30 June 2004.

13. Martin, J.N., "Using Architecture Modeling to Assess the Societal Benefits of the Global Earth Observation System-of-Systems," Systems Journal, IEEE , vol.2, no.3, pp.304,311, Sept. 2008.

14. Parisi, C.; Sahin, F.; Jamshidi, M., "A discrete event XML based system of systems simulation for robust threat detection and integration," System of Systems Engineering, 2008. SoSE '08. IEEE International Conference on , vol., no., pp.1,8, 2-4 June 2008.

15. DoDAF version 2.02 volume 1. “Introduction, Overview, and Concepts Manager’s Guide”. May 28, 2019. Accessed September 30, 2020. https://dodcio.defense.gov/Portals/0/Documents/DODAF/DoDAF%20V2%20-%20Volume%201.pdf.

16. Acheson, P. and Dagli, C. (2012). Fuzzy Assessor Using Type 1 and Type 2 Fuzzy Sets. Procedia Computer Science, Volume 8, pages 159-164, March 2012.

17. Webster (2016), “Resilience" Merriam-Webster.com. Merriam-Webster, n.d. Web. 10 May 2016.

18. Reggiani, Aura, (2013). Network resilience for transport security: Some methodological considerations, Transport Policy, Volume 28, July 2013, Pages 63-68, ISSN 0967-070X.

19. Mebarki, Ahmed (2017). “Resilience: Theory and metrics – A metal structure as demonstrator”, Engineering Structures Journal, Vol. 138, pages 425-433. ISSN 01410296. DOI 10.1016/j.engstruct.2017.02.026.

20. Jawaid, M., Thariq, M., & Saba, N. (2019). “Mechanical and Physical Testing of Biocomposites, Fibre-Reimforced Composites and Hybrid Composites”. Woodhead Publishing, Cambridge, MA. Page 5, Paragraph 1.2.3.1.

21. Brown, A., (1955). “Second-Order Transition Temperature and Fiber Properties”. Textile Research J., 25, 891 (1955). https://journals.sagepub.com/doi/10.1177/004051755502501101.

22. Holling, C.S. (1966). “Engineering Resilience versus Ecological Resilience”, Engineering Within Ecological Constraints, Pp31-44.

23. Holling, C.S. (1973). “Resilience and Stability of Ecological Systems”, Annual Review of Ecology and Systematics, Pp 1-23.

24. Berkes, F., & Folke, C. (Eds.). (1998). Linking Social and Biological Systems. Cambridge: Cambridge University Press.

25. Holling, C.S. 2009.Pp. 31–44 in Engineering Within Ecological Constraints.

26. Connell, Sean D. & Ghedini, Giulia (2015). Resisting regine-shifts: the stabilizing effect of compensatory processes. Trends in Ecology and Evolution, Vol. 30, Issue 9, pages 513-515.

27. Yodo N. & Wang P. (2016). Resilience Modeling and Quantification for Engineered Systems Using Bayesian Networks. ASME. J. Mech. Des. 2016;138(3):031404-031404-12. doi:10.1115/1.4032399.

28. Carlson, L., Bassett, G., Buehring, W., Collins, M., Folga, S., Haffenden, B., Petit, F., Phillips, J., Verner, D., & Whitfield, R., (2012). "Resilience: Theory and Applications", Argonne Lab, Decision and Information Sciences Division. January 2012.

29. Juan-García, P., Butler, D., Comas, J., Darch, G., Sweetapple, C., Thornton, A., Corominas, Ll. (2017). “Resilience theory incorporated into urban wastewater systems management. State of the art”, Water Research Journal,vol. 115, pages 149-161.

30. Beheshtian, A., Donaghy, K.P., Geddes, R.R. & Rouhani, O.M. (2017). Planning resilient motor-fuel supply chain. International Journal of Disaster Risk Reduction, vol. 24, pages 312-325.

31. Nan, Cen & Sansavini, Giovanni (2017). “A quantitative method for assessing resilience of interdependent infrastructures”, vol. 157, pages 35-53.

32. Singh, V. K., Govind`arasu, M., Porschet, D., Shaffer, E., & Berman, M. (2019). Distributed Power System Simulation using Cyber-Physical Testbed Federation: Architecture, Modeling, and Evaluation. In (Vol. 1, pp. 26-32): IEEE.

33. Confrey, J., Etemadi, A. H., Stuban, S. M. F., & Eveleigh, T. J. (2020). Energy Storage Systems Architecture Optimization for Grid Resilience With High Penetration of Distributed Photovoltaic Generation. IEEE Systems Journal, 14(1), 1135-1146. doi:10.1109/JSYST.2019.2918273.

34. Bradtmöller, Marcel, Grimm, Sonja, & Riel-Salvatore, Julien (2017). "Resilience theory in archaeological practice – An annotated review", Quaternary International Journal, vol. 446, 2 August 2017, pages 3-16. http://dx.doi.org/10.1016/j.quaint.2016.10.002.

35. Weiberg, Erika (2017) "What can Resilience Theory do for (Aegean) Archaeology". Accessed June 29, 2020. https://www.diva-portal.org/smash/get/diva2:572734/FULLTEXT01.pdf.

36. Wang, W., & Ma, X. (2020). Blockchain-Based Botnets for Command-and-Control Resilience. In (1 ed., pp. 217-236): Routledge.

37. Leng, J., Buyuktosunoglu, A., Bertran, R., Bose, P., Chen, Q., Guo, M., & Janapa Reddi, V. (2020). Asymmetric Resilience: Exploiting Task-Level Idempotency for Transient Error Recovery in Accelerator-Based Systems. In (pp. 44-57): IEEE.

38. Kalra, C., Previlon, F., Rubin, N., & Kaeli, D. (2020). ArmorAll: Compiler-based Resilience Targeting GPU Applications. ACM transactions on architecture and code optimization, 17(2), 1-24. doi:10.1145/3382132.

39. Ascia, G., Catania, V., Monteleone, S., Palesi, M., Patti, D., Jose, J., & Salerno, V. M. (2020). Exploiting Data Resilience in Wireless Network-on-chip Architectures. ACM journal on emerging technologies in computing systems, 16(2), 1-27. doi:10.1145/3379448.

40. Campos, P. (2020). Resilience, education and architecture: The proactive and "educational" dimensions of the spaces of formation. International Journal of Disaster Risk Reduction, 43, 101391. doi: 10.1016/j.ijdrr.2019.101391.

41. Hardiilla, D., & Nugroho, A. C. (2020). Resilience Capacity Planning: A Strategy Requirement for Vernacular Architecture Existences as a part of Sustainable Development in Lampung. IOP conference series. Earth and environmental science, 409, 12012. doi:10.1088/1755-1315/409/1/012012.

42. Acheson, P., & Dagli, C. (2016). Modeling Resilience in System of Systems Architecture. Procedia Computer Science, 95, 111-118. doi: 10.1016/j.procs.2016.09.300.

43. Noei, Shirin, Sargolzaei, Arman, Abbaspour, Alireza, & Yen, Kang (2016). "A Decision Support System for Improving Resiliency of Cooperative Adaptive Cruise Control Systems", Procedia Computer Science, Volume 95, 2016, Pages 489-496, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2016.09.326.

44. Moya, Ignacio, Chica, Manuel, Sáez-Lozano, José L., & Cordón, Óscar (2017). "An agent-based model for understanding the influence of the 11-M terrorist attacks on the 2004 Spanish elections", Knowledge-Based Systems Journal, vol. 123, pages 200-216.

45. Kilian, J. C., & Schuck, T. M. (2016). Architecture and System-of-Systems Design for Integrated Missile Defense.

46. Jackson, Scott, (2010). Architecting Resilient Systems Accident Avoidance and Survival and Recovery from Disruptions. John Wiley & Sons, Inc. Hoboken, New Jersey.

47. Resilience Capacity Index. Building Resilient Regions. Institute of Governmental Studies, The University of California, Berkley. Accessed 11 July 2016. http://brr.berkeley.edu/rci/.

48. Chen, Yee Ming and Shih-Chang Wang. (2008). "An evolutionary compensatory negotiation model for distributed dynamic scheduling". Applied Soft Computing, volume 8, issue 2, March 2008, pages 1093-1104.

49. Hyten, G. J. (2014). AFSPC: Defending Our Edge, General John Hyten speaking at the Air Force Association Air & Space Conference, Washington, D.C. September 16, 2014. https://www.afspc.af.mil/About-Us/Leadership-Speeches/Speeches/Display/Article/731711/afspc-defending-our-edge/.

50. Hyten, John E., (2015). NDIA Luncheon, Peterson Air Force Base, Colorado. May 21, 2015. Accessed 11 July 2016. http://www.afspc.af.mil/About-Us/Leadership-Speeches/Speeches/Display/Article/731706/2015-ndia-luncheon.

51. "AFSPC Commander outlines vision for command at Space Symposium". Published April 13, 2016. Downloaded on 9 May 2016 from http://www.afspc.af.mil/News/Article-Display/Article/730810/afspc-commander-outlines-vision-for-command-at-space-symposium.

52. Mission Engineering and Integration (MEI) Guidebook (2019). Office of the Under Secretary of Defense (OUSD) for Acquisition and Sustainment (A&S), https://www.dau.edu/tools/Lists/DAUTools/Attachments/389/MEI%20Guidebook%2020200226.pdf, accessed August 15, 2020.

53. Acheson, P., Dagli, C., & Kilicay-Ergin, N. (2013). Model Based Systems Engineering for System of Systems Using Agent-based Modeling. Procedia Computer Science, 16, 11-19. doi:10.1016/j.procs.2013.01.002.

54. Acheson, P., Pape, L., Dagli, C., Kilicay-Ergin, N., Columbi, J., & Haris, K. (2012). Understanding System of Systems Development Using an Agent-Based Wave Model. Procedia Computer Science 2012, vol 12, p21-30.

55. N. Kilicay-Ergin, P. Acheson, J. Colombi and C. H. Dagli, "Modeling system of systems acquisition," 2012 7th International Conference on System of Systems Engineering (SoSE), Genova, 2012, pp. 514-518, doi:10.1109/SYSoSE.2012.6384152.

56. Acheson, Paulette, Cihan Dagli, and Nil Kilicay-Ergin. (2012). "Fuzzy Decision Analysis in Negotiation between the System of Systems Agent and the System Agent in an Agent-Based Model", International Journal of Soft Computing and Software Engineering [JSCSE], March 2013, ISSN: 2251-7545 & DOI: 10.7321/jscse.

57. Acheson, P. (2011). The Application of Fuzzy Logic in Linguistic Variables to Determine Good Requirements. CSER 2011.

58. Howard, R.A., 1968. "The foundations of decision analysis," IEEE Transactions on Systems Science and Cybernetics, vol. 4, pp. 221-219, Sept 1968.

59. Watson, Stephen R., Jonathan J. Weiss, and Michael L. Donnell. 1979. "Fuzzy Decision Analysis," IEEE Transactions on Systems, Man, and Cybernetics, vol.9, no.1, pp.1-9, Jan. 1979.

60. Carlsson, Christer and Robert Fuller. 2002. "Fuzzy Reasoning in Decision Making and Optimization." Springer-Verlag Berlin Heidelberg.

61. Yongchang Ren; Qiang Quan; Tao Xing; Xiaoji Chen; , "Fuzzy Decision Analysis of the Software Configuration Management Tools Selection," Information Science and Engineering (ISISE), 2010 International Symposium on , vol., no., pp.295-297, 24-26 Dec. 2010.

62. Bender, M.J.; Swanson, S.; Robinson, R.; , "On the role of fuzzy decision support for risk communication among stakeholders," Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on , vol.1, no., pp.317-322 vol.1, 12-15 Oct 1997. doi: 10.1109/ICSMC.1997.625769.

63. Huang, J.W.; Roy, R.J.; , "Multiple-drug hemodynamic control using fuzzy decision theory," Biomedical Engineering, IEEE Transactions on , vol.45, no.2, pp.213-228, Feb. 1998. doi: 10.1109/10.661269. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=661269&isnumber=14396.

64. Flores, B.F.; Salonga, J.H.M.; Nerves, A.C.; , "Multi-objective Transmission Expansion Planning Using an Elitist Non-dominated Sorting Genetic Algorithm with Fuzzy Decision Analysis," Modelling Symposium (AMS), 2011 Fifth Asia , vol., no., pp.168-173, 24-26 May 2011doi: 10.1109/AMS.2011.39. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5961412&isnumber=5961217.

65. Acheson, P. (2011). Model for Distributed Infantry Immersion Training System, IEEE Systems Engineering Conference 2011.

66. Jackson, S., Stephen Cook, and Timothy L. J. Ferris. (2015). A Generic State-Machine Model of System Resilience. 18(1), 5.

67. Bray, Hiawatha (2019). “Ransomware attacks: Not if, but when, agencies say”. Boston Globe Staff, July 15, 2019. https://www.bostonglobe.com/business/2019/07/15/ransomware-attacks-not-but-when-agencies-say/QNFNrjUkxc1f643qHc8p1H/story.html.

VITA

Paulette Bootz Acheson graduated in December 2020 from Missouri University of Science and Technology with a PhD in Systems Engineering. She graduated from California State University, Fullerton in June 2000 with a BS in Biology. She graduated from University of Southern California with an MS in Applied Mathematics in May 1998. In May 1978 she graduated from California State University, Fresno with a BA in Mathematics.

Paulette Bootz Acheson worked for Aerospace Corporation as a Systems Engineer and Software Engineer beginning 2010. She worked for Boeing as a Systems Engineer and Software Engineer from 2003 to 2009. From 1997-2000 she implemented algorithms to monitor GPS receivers as a contractor for Raytheon. From 1990 to 1996 she worked as a Software Engineer for Northrop-Grumman doing design, code, unit test, integration, system test of aircraft systems. From 1982-1989 she performed system and software verification and validation as a contractor to Interstate Electronics in Anaheim. She worked as a Software Engineer for Hughes Ground Systems from 1980-1981. She worked as an Associate Engineer for McDonnell Astronautics from 1978 to 1980 developing programs to process telemetry data.

Paulette Bootz Acheson taught Computer Science at California State University, Long Beach from 2001-2002. She taught Computer Science and Statistics at Cypress College in 1999.