Summer 2015

Computational intelligence based complex adaptive system-of-systems architecture evolution strategy

Siddharth Agarwal

COMPUTATIONAL INTELLIGENCE BASED COMPLEX ADAPTIVE SYSTEM-OF-SYSTEMS ARCHITECTURE EVOLUTION STRATEGY

by

SIDDHARTH AGARWAL

A DISSERTATION

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

SYSTEMS ENGINEERING

2015

ABSTRACT

The dynamic planning for a system-of-systems (SoS) is a challenging endeavor. Large scale organizations and operations constantly face challenges to incorporate new systems and upgrade existing systems over a period of time under threats, constrained budget and uncertainty. It is therefore necessary for the program managers to be able to look at the future scenarios and critically assess the impact of technology and stakeholder changes. Managers and engineers are always looking for options that signify affordable acquisition selections and lessen the cycle time for early acquisition and new technology addition. This research helps in analyzing sequential decisions in an evolving SoS architecture based on the wave model through three key features namely; meta-architecture generation, architecture assessment and architecture implementation. Meta-architectures are generated using evolutionary algorithms and assessed using type II fuzzy nets. The approach can accommodate diverse stakeholder views and convert them to key performance parameters (KPP) and use them for architecture assessment. On the other hand, it is not possible to implement such architecture without persuading the systems to participate into the meta-architecture. To address this issue a negotiation model is proposed which helps the SoS manger to adapt his strategy based on system owners behavior. This work helps in capturing the varied differences in the resources required by systems to prepare for participation. The viewpoints of multiple stakeholders are aggregated to assess the overall mission effectiveness of the overarching objective. A search and rescue mission (SAR) SoS example problem illustrates application of the method. Also a dynamic programing approach can be used for generating meta-architectures based on the wave model.

ACKNOWLEDGMENTS

Foremost, I would like to express my gratitude to my advisor Dr. Cihan H. Dagli for the continuous support of my Ph.D. study and research, for his patience and immense knowledge. Besides my advisor, I would like to thank the rest of my thesis committee: Dr. David Enke, Dr. Abhijit Gosavi, Dr. Ruwen Qin, and Dr. Robert Paige and Louie E. Pape for their encouragement, insightful comments, and hard questions.

Most importantly, I would like to thank my wife Neha. Her friendship, care, support, encouragement, patience and unwavering love were undeniably the bedrock upon which the past four years of my life stand. Her immense devotion and love needs no testimony. She has been my motivation and strength since I met her.

I am and will forever remain deeply indebted to my parents Mr. Niraj Agarwal and Mrs. Annu Agarwal, for giving birth to me at the first place, for their immeasurable sacrifices, supporting me spiritually and financially throughout my life. They have inculcated in me the right set of values and helped me become a better human being in all walks of life. No achievement for me whether big or small can ever be measured without their greater share of contribution. I wish to thank, my brother-in-law Manjit and his wife Priyanka, my in-laws Shri. Sunil Koolwal and Smt. Kalpana Koolwal, who have brought great joy and satisfaction to my life.

Finally I would like to thank my maternal grandmother late Mrs. Usha Mittal for always encouraging and believing in me against all odds. I believe my paternal grandparents late Shri. Prem Chand Agarwal and Smt. Shanta Agarwal have always loved me more than anyone else. Thanks and good wishes to all my SMART lab mates for the stimulating discussions, sleepless nights, and the fun we had. At the same time I would like to thank everyone who has touched me in some way, knowingly or unknowingly, and assisted me reach this mark and I wish them all the best in their future endeavors.

This material is based upon work supported, in whole or in part, by the U.S. Department of Defense through the Systems Engineering Research Center (SERC) under Contract H98230-08-D-0171. SERC is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. INTRODUCTION

In the real world, systems are complex, non-deterministic, evolving, and have human centric behaviors. The connections of all complex systems are non-linear, globally distributed, and evolve both in space and in time. Because of non-linear properties, system connections create an emergent behavior. It is imperative to develop an approach to deal with such complex large-scale systems. The approach and goal is not to try and control the system, but design the system such that it controls and adapts itself to the environment quickly, robustly, and dynamically. These complex entities include both socioeconomic and physical systems, which undergo dynamic and rapid changes. Some of the examples of complex systems include transportation systems (Trentesaux, Knothe, Branger, & Fischer, 2015), health systems (Obal, & Lin, 2015), internet of things (Maia, et al., 2014), smart cities development (Wang, 2015), energy security systems (Hadian & Madani, 2015), defense frameworks (Marti et al., 2015), and manufacturing infrastructures (Nahavandi, et al., 2015).

1.1 BACKGROUND AND MOTIVATION

A complex system is a system featuring a large number of interacting components, whose capability is not a linear sum of its components. Besides this system exhibits self-organization and emergent properties. Complex Adaptive Systems (CAS) can be referred to as special cases of complex systems. CAS can adapt (through learning) and evolve within a dynamic environment.

Langton's egg diagram (Langton, 1990) depicts three primary classes fixed (Class I), periodic (Class II) and chaotic (Class III) as shown in Figure 1.1. Complexity (Class IV) lies at the edge of periodicity and chaos. This figure can help us understand that all these classes are continuous and have a thin margin of separation.

A number of definitions exist that define a system-of-systems (SoS). A definition that relates SoS to complex systems is that “systems of systems are large scale concurrent and distributed systems that are comprised of complex systems” (Kotov, 1997).

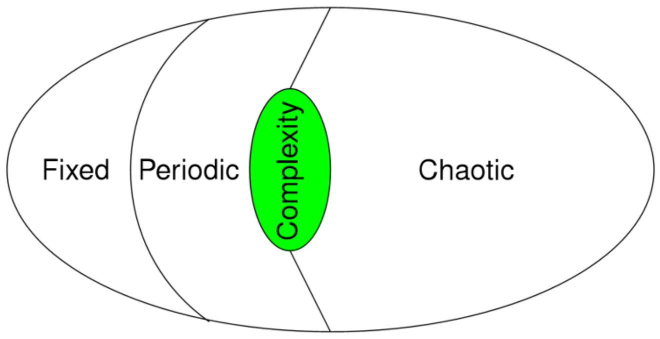


Figure 1.1. Relative location of periodic, chaotic, and “complex” transitions

SoS can be designated as complex systems due to features such as:

- Emergent behavior that provides the creativity, diversity, and complexity
- Organized complexity allows system to achieve its goals
- Dynamic stability is maintained through constant self-adjustment

Another concept that has emerged recently has been the class of Cyber Physical Systems (CPS) (Zhang, 2015). CPS is a SoS which integrates physical system with cyber capability in order to improve the performance (Dong, P., Han, Y., Guo, X., & Xie, F. (2015). Cyber capability includes a model of the process that can be utilized to make decisions over the system.

Although classically Maier (1998) suggests categories of SoS development but infinitely many SoS exist on the edges of the categories thus making it a continuum. These SoS may vary based on their degree of managerial control over the participating systems and their structural complexity. Figure 1.2. is an attempt to show the above argument. The author claims many SoS with different configurations can fill this gap.

SoS achieves the required goal by introducing collaboration between existing system capabilities that are required in creating a larger capability based on the meta-architecture selected for SoS. The level of the degree of influence on individual systems architecture through the guidance of SoS manager in implementing SoS meta-architecture can be classified as directed, acknowledged, collaborative and virtual. Acknowledged SoS have documented objectives, an elected manager and defined resources for the SoS. Nonetheless, the constituent systems retain their independent ownership, objectives, capital, development, and sustainment approaches. Acknowledged SoS shares some similarities with directed SoS and collaborative SoS.

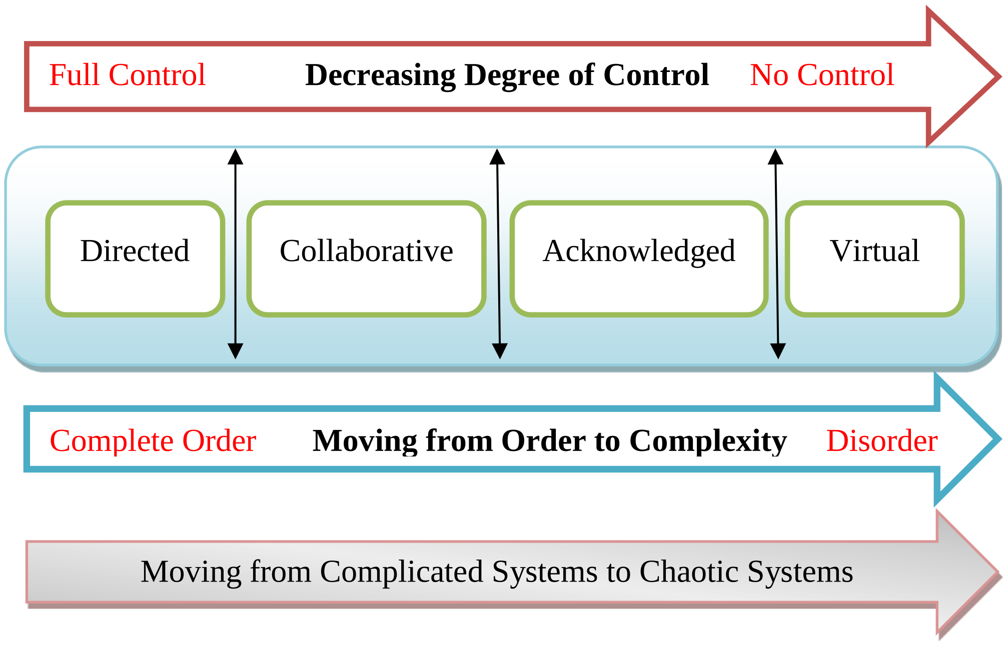

Full Control
Decreasing Degree of Control
No Control
Directed
Collaborative
Acknowledged
Virtual
Complete Order
Moving from Order to Complexity
Disorder
Moving from Complicated Systems to Chaotic Systems

Figure 1.2. Types of SoS based on Degree of Control and Degree of Complexity

To model any system an approach is needed that specifies the underlying properties and is able to successfully recreate the dynamics in the system. Systems architecting can be defined as specifying the structure and behavior of an envisioned system. Classical system architecting deals with static systems whereas the processes of System of Systems (SoS) architecting has to be first done at a meta-level. The architecture achieved at a meta-level is known as the meta-architecture. The meta-architecture sets the tone of the architectural focus (Malan & Bredemeyer, 2001) and it drives the process of architecting further. It narrows the scope of the fairly large domain space and boundary. Meta-architecture helps in communicating the information to the stakeholders at a very high level. Although the architecture is still not fixed but meta-architecture provides multiple alternatives for the final architecture. Thus architecting can be referred to as filtering the meta-architectures to finally arrive at the architecture. The SoS architecting involves multiple systems architectures to be integrated to produce an overall large scale system meta-architecture for a specifically designated mission (Jamishidi, 2008).

Architecture simulation and modeling techniques for Acknowledged SoS are still in their initial stages. The process includes producing a meta-architecture using multi-objective evolutionary algorithms. Multiple objective decisions making (MODM) increases in difficulty with growing number of objectives (Key performance parameters). The probability of finding dominated solutions based on three or more objectives is very low. To solve this problem the architectures assessment technique uses a fuzzy type II modular rule base approach (fuzzy networks) that allows multiple key performance parameters to be evaluated at the same time. The fuzzy rule base defines the preference of the decision maker in our case the Acknowledged SoS manager.

Furthermore, meta-architectures are often not fully realizable in real conditions. It is often difficult to secure implementation of the generated meta-architectures for System of Systems (SoS) in actual situations given the negotiation complexity and individual systems behavior. In SoS where individual systems have their own self-interests, negotiation becomes an important aspect of SoS acquisition. During a negotiation, each party communicates its own desires and hence the problem of interest is to find that point of mutually beneficial agreement. This major issue is resolved by introducing negotiation modules between individual systems and SoS manager based on domain specific information. The domain is defined by the set of issues being negotiated over which include price (value for capability being acquired), performance (task execution capacity) and deadline (delivery date). The meta-architecture generated is negotiated for possible implementation by the acknowledged SoS manager through machine learning based negotiation model. The negotiation is modeled as a bilateral counteroffer, resembling one SoS manager and an individual system. The agreements after each negotiation round are not an obligation on either party.

Additional motivation for pursuing this research is from an engineering research viewpoint since no single method of automated negotiation that is applicable to all situations. This research also aims to fill a gap by utilizing machine learning fuzzy logic techniques to design a protocol applicable in large scale systems settings. Automated tools can be used in conjunction with the human negotiator to aid in a negotiation task. The negotiation strategies are needed to enhance the efforts of people during negotiations. Furthermore, large-scale projects that involve different departments (systems) are developed to bring sustainability and prosperity. The program managers need to have a strategy for negotiating and implementing such projects in an ecological and equally beneficial way. Additionally, negotiations involve conflicts over the consumption of joint resources or task assignments and conflicts between a buyer and a seller.

This research contributes to the state of the art in Acknowledged SoS-based negotiations in two key areas. It presents the first attempt to combine multiple behaviors of systems participating in a complex adaptive SoS operational scenario. Secondly, research proposes the use of neural network (Agarwal & Ganguli, 2011) architectures as techniques for SoS manager to adapt his negotiation strategy while dealing with multiple constituent systems on multiple issues such as deadline, funding and performance This is a very quick and effective approach to adapt communication strategies in SoS environment. Our attempt is to present an integrated acknowledged SoS architecting model whose capabilities include extensive SoS meta-architecture generation covering the entire design space, flexible and robust architecture assessment, and final architecture securement through simulated negotiations.

The major objectives of this reasearch are:

- To develop a simulation for acknowledged SoS architecture selection and evolution.
- To have a structured, repeatable approach for planning and modeling.
- To study and evaluate the impact of individual system behavior on SoS capability and architecture evolution process.

1.2. IMPACT OF THE RESEARCH

This research has impacts on expanding the application of systems engineering across a wide section of academic and industry domains.

1.2.1. Contribution to the State of Systems Engineering Knowledge. In the subject of dynamical systems-of-systems (SoS), there has been a great deal of growth developing theories that describe the behavior of individual systems. However, comparatively less research establishes how various systems form coalitions and negotiate with the SoS manager to establish an architecture over time. A stochastic architecting technique using computational intelligence (particle swarm & fuzzy logic) in an integrated environment is implemented. This allows the SoS managers to be able to look at the future scenarios and critically assess the impact of technology and stakeholder changes. This aids the manager in looking for options that signify affordable acquisition selections and lessen the cycle time for early acquisition and new technology addition. Furthermore, to include and discard system capabilities, a negotiation strategy is required. A negotiation strategy usually consists of three main modules: modeling the opponent behavior (clustering), decision-making criteria (fuzzy logic) and finally generating a counter-offer (time based equations). This overall structure provides a useful basis for developing SoS architecting technique that can evolve and adapt to changes in its environment. For an SoS manager the challenging problem is to capture the hidden objective function of the opponent or autonomous systems. This research proposes a novel strategy based on hybrid clustering and neural networks that can be used in a multi-issue negotiation setting. The experimental results show that the proposed method is effective in a variety of application domains against the state-of-the-art negotiating agents. The research focusses and improves some key areas in systems engineering such as:

1) Systems Architecting,
2) Optimization,
3) Decision-making under ambiguity
4) Incorporating machine learning tools and;
5) Domain-specific modeling and simulation

1.3. DISSERTATION ORGANIZATION

This dissertation is organized as follows:

Section 1, introduction, briefly introduces the motivation of this research. Section 2, literature review, discusses the application of evolutionary algorithms in solving many objective problems. This section also gives a background on automated negotiation. This section provides search-based architecture development framework, presents the proposed architecture development framework along with the discussions of some enabling technologies for each of its components.

Section 3, overview of integrated model developed to address the problem. It provides review of some background knowledge needed to develop the approaches proposed in this research such as evolutionary algorithms, fuzzy logic, and machine learning algorithms.

Section 4 presents how the proposed approaches are implemented to design search and rescue system-of-systems.

Section 5 encompasses discusses of results and what-if analysis. It also provides some insights into possible future expansions of the current work. Section 6, conclusion and future work, discusses strengths and limitations of the proposed approach. The next chapter provides a review on SoS and its types, current and past SoS projects, SoS acquisition, a review of techniques to handle many objective optimization problems, and finally a background on automated negotiation concepts and importance.

2. LITERATURE REVIEW ON SOS & AUTOMATED NEGOTIATION

2.1. SYSTEM-OF-SYSTEMS

System-of- Systems (SoS) consists of multiple complex adaptive systems that behave autonomously but cooperatively (Dahman, Lane, Rebovich, & Baldwin, 2008). The continuous interaction between them and the interdependencies produces emergent properties that cannot be fully accounted for by the “normal” systems engineering practices and tools. System of Systems Engineering (SoSE), an emerging discipline in systems engineering is attempting to form an original methodology for SoS problems (Luzeaux, 2013). The first task that must be completed in a large scale problem is identifying it as a SoS problem. A recent book highlights the case studies in the area of SoS (Gorod, White, Ireland, Gandhi, & Sauser, 2014). Figure 2.1. (Agarwal et al., 2015), describes the basic framework of system-of-systems (SoS).

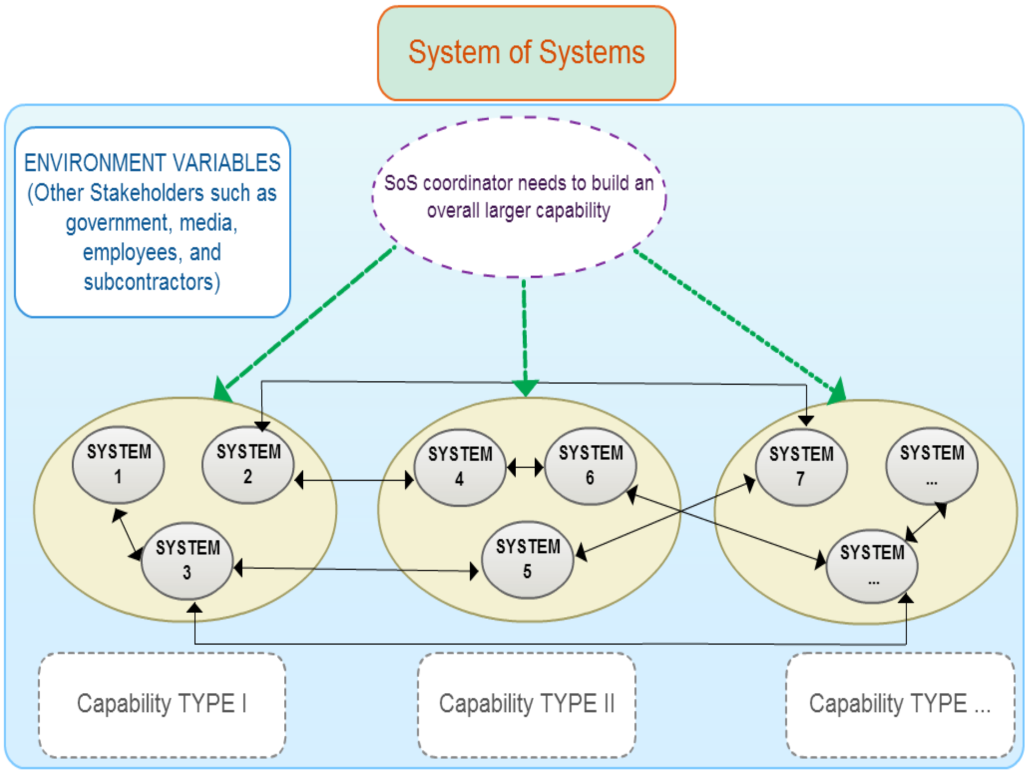

System of Systems
ENVIRONMENT VARIABLES (Other Stakeholders such as government, media, employees, and subcontractors)
SoS coordinator needs to build an overall larger capability
SYSTEM 1
SYSTEM 2
SYSTEM 3
SYSTEM 4
SYSTEM 6
SYSTEM 5
SYSTEM 7
SYSTEM ...
SYSTEM ...
Capability TYPE I
Capability TYPE II
Capability TYPE ...

Figure 2.1. An Acknowledged Systems-of-Systems

Three major elements include an SoS coordinator, environment variables and individual systems. Each system carries a specific capability and many systems can have the same capability. Together all systems participate to achieve a larger purpose under the supervision of SoS coordinator.

Figure 2.2. illustrates the logical system architecture design process. This figure is adapted from Kaplan (2006). The figure describes the relationships between individual systems and overall SoS effectiveness. The figure also describes how scenarios, operations, capabilities functions and systems are related to each other.

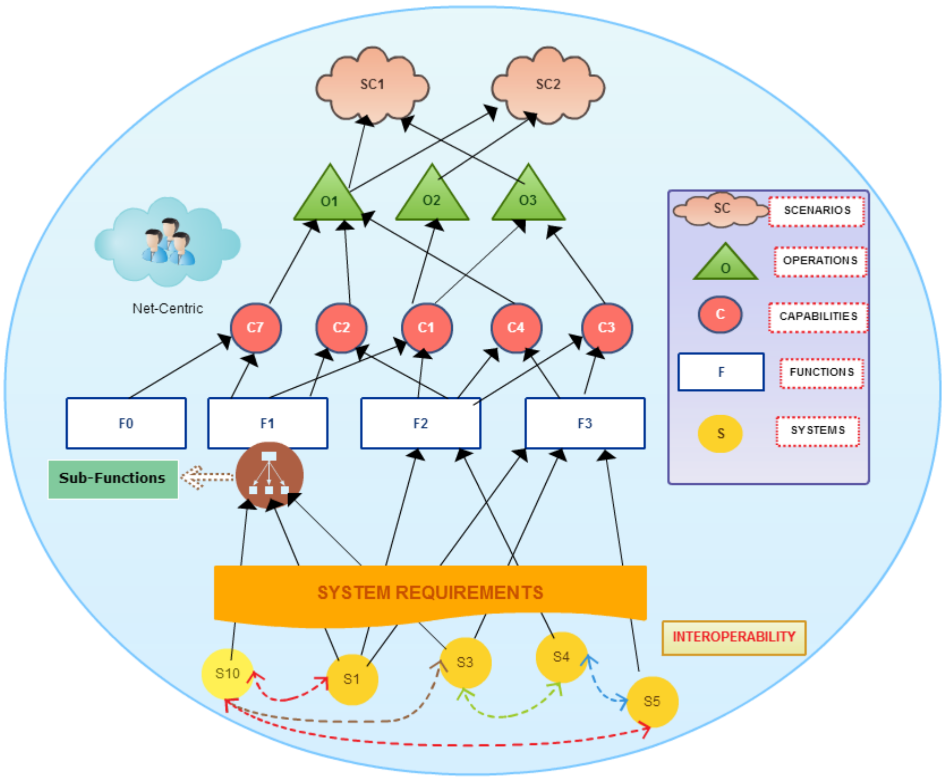
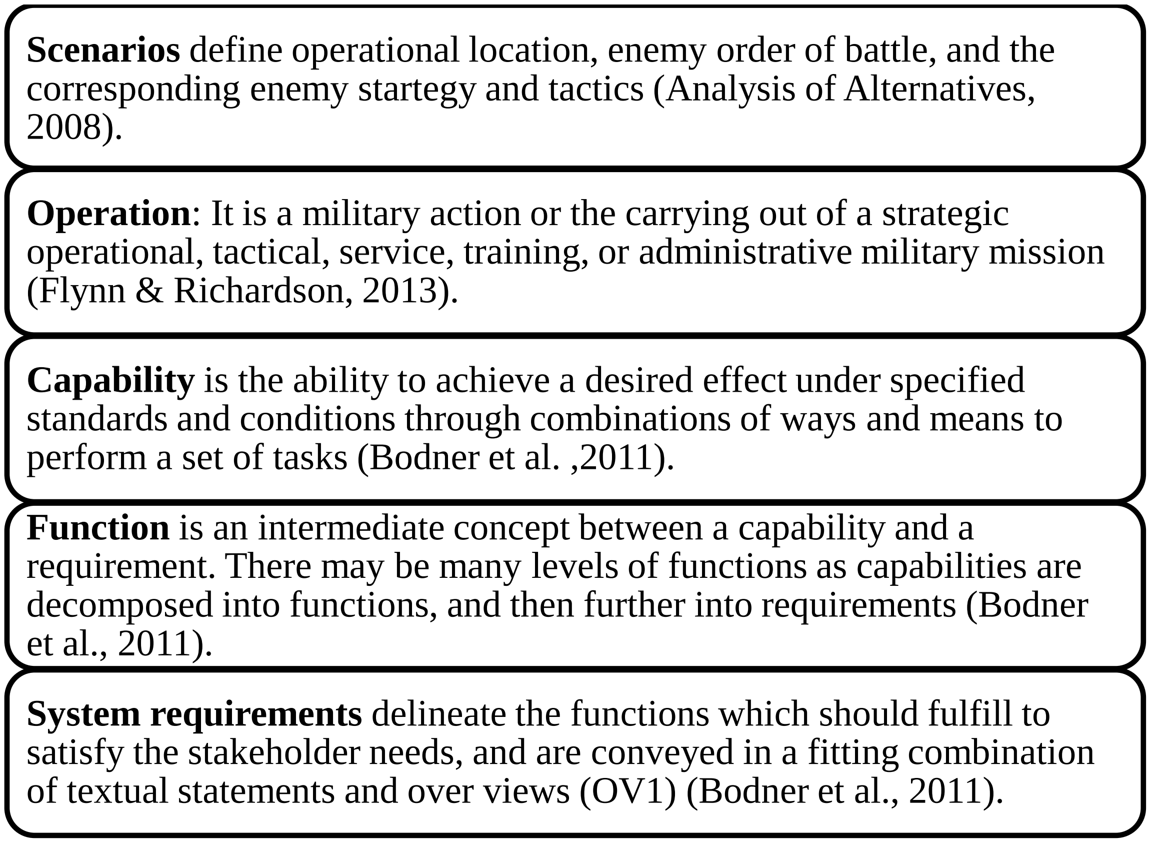


Figure 2.2. Systems engineering architecture design

Capabilities are decomposed into functions, which are further broken down into requirements for individual systems. This figure synthesizes an architectural framework for operational scenarios. This design allows for incremental flexibility in capabilities and functions. It is appropriate to be executing only the functions that are necessary and making additions as needs evolve. Figure 2.3. defines various concepts utilized in Figure 2.2.



Scenarios define operational location, enemy order of battle, and the corresponding enemy startegy and tactics (Analysis of Alternatives, 2008).

Operation: It is a military action or the carrying out of a strategic operational, tactical, service, training, or administrative military mission (Flynn & Richardson, 2013).

Capability is the ability to achieve a desired effect under specified standards and conditions through combinations of ways and means to perform a set of tasks (Bodner et al. ,2011).

Function is an intermediate concept between a capability and a requirement. There may be many levels of functions as capabilities are decomposed into functions, and then further into requirements (Bodner et al., 2011).

System requirements delineate the functions which should fulfill to satisfy the stakeholder needs, and are conveyed in a fitting combination of textual statements and over views (OV1) (Bodner et al., 2011).

Figure 2.3. Systems engineering logical architecture design

2.1.1. Types of System-of-Systems. Maier (1998) discussed both the applicable conditions required to ascertain that a problem is indeed SoS. One of the SoS types of immediate importance is the Acknowledged SoS, which has recognized objectives, a designated manager with limited authority, and resources for the SoS (Ncube, Lim, & Dogan, 2013). Acknowledged SoS shares several attributes with both Collaborative SoS and Directed SoS (Bergey et al., 2009). Figure 2.4. illustrates this concept (Dahmann, Baldwin & Rebovich, 2009). The other broad type of SoS, Virtual lacks a central management and a centrally approved purpose for SoS, and has independent development processes (Dahman & Baldwin, 2008). Acknowledged system of systems (SoS) accomplishes best when the contributing systems have no direct control over them yet they deliver capabilities required to meet the purpose of the SoS operating in an interdependent environment. Acknowledged SoS have political and economic interdependence, the need to share resources and interconnect systems for global partnerships.

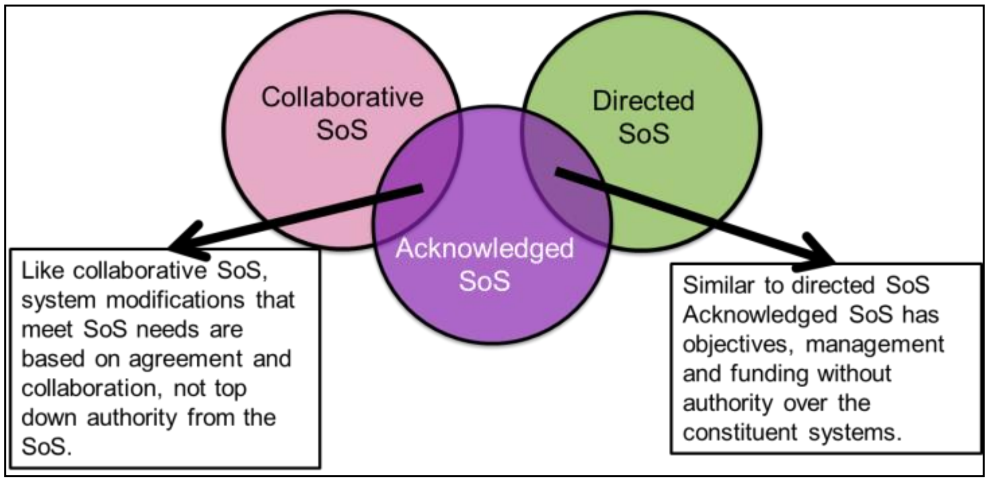

Collaborative SoS
Directed SoS
Acknowledged SoS
Like collaborative SoS, system modifications that meet SoS needs are based on agreement and collaboration, not top down authority from the SoS.
Similar to directed SoS Acknowledged SoS has objectives, management and funding without authority over the constituent systems.

Figure 2.4. Properties of Acknowledged SoS

The System of Systems (SoS) have been found to exhibit properties similar to complex adaptive systems (Sage & Cuppan, 2001). Russell Ackoff (1971) offered a systematic view on the concepts and terms related to the science of complex systems. He recommended that a systems approach be used to analyze the system as a whole rather than analyzing its parts individually. Ackoff classified systems into four major types according to not only their behavior but also the outcome of the behavior itself: state maintaining, goal seeking, multiple-goal-seeking and purposeful.

Whereas, four major types of SoS are usually defined as the following (Dahman et al., 2011):

- Directed: Have SoS objectives, management, funding and authority; systems are subordinated to SoS

- Acknowledged: Have SoS objectives, management, funding and authority; however systems retain their own management, funding and authority in parallel with the SoS
- Collaborative: No objectives, management, authority, responsibility, or funding at the SoS level; Systems voluntarily work together to address shared or common interest
- Virtual: Like collaborative, but systems don’t know about each other

2.2. LITERATURE REVIEW OF SOS PROJECTS

In this section a brief description of major SoS projects currently being pursued in a variety of domains are discussed. This section will help the reader get an overview of the scope of research being conducted. The descriptions do not necessarily follow any order in which the projects came into inception. DANSE SoS stands for Designing for Adaptation and Evolution in System of Systems (Arnold et al., 2013). DANSE project addresses the challenging technical, management, and political problems within organizations. The main features include combining the strengths of several infrastructures and objects present because of advances in communications, sensors and actuating competencies. DANSE is among several projects in SoS funded by the European Commission as part of the Seventh Framework Program. The purpose of the DYMASOS (Dynamic Management of Physically Coupled Systems of Systems) project is to explore methods for the distributed management of large physically connected systems along with distributed autonomous management and global coordination (Paulen, & Engell, 2014). COMPASS stands for Comprehensive Modelling for Advanced Systems of Systems and aims to develop collaborative research on model-based techniques for developing and maintaining SoS (Coleman et al., 2012). For example, a flexible and responsive SoS can be developed for emergency management, given the fact that individual systems were not intended for collaboration. T-AREA-SoS (Trans-Atlantic Research and Education Agenda on Systems of Systems) was developed through cooperation between EU-US Systems of Systems (SoS) research (Siemieniuch et al., 2013) T-AREA-SoS aims to achieve European competitiveness and improve the societal impact through development and management of large complex systems.

The CYPHERS project aims at developing an integrated cyber-physical roadmap and strategy for Europe (CPS20, 2014). Its ultimate goal is to combine and expand Europe's capability in embedded and mobile computing as well as in control of networked embedded systems. Some projects that are closely related to CYPHERS are Hycon2: highly-complex and networked control systems; EMSIG: embedded systems special interest group; artist design: European network of excellence on embedded systems design; and CPSoS: cyber-physical systems of systems. AMADEOS aims critical systems certification for SoS (Montecchi, Lollini, & Bondavalli, 2014). Its abbreviation stands for Architecture for Multi-criticality Agile Dependable Evolutionary Open System of Systems. The AMADEOS project emphasizes on evolution, emergence, dependability and security, taking into consideration-embedded devices and the cloud as the projects execution platform. It has three significant objectives namely: to introduce a concept of global time that can be accessed and recognized by all elements of the SoS, ability to explain and formalize SoS evolvability and dynamicity, and handling emerging properties in SoS. The CPSOS is a support action, to be completed in 30 months, that aims at developing a roadmap on research and innovation in engineering and management of cyber-physical systems of systems (Reniers & Engell, 2014). CPSOS are cyber-physical systems which exhibit the features of systems of systems. The aim of CPSOS is to study and analyze computing and communication systems that interact with large complex physical systems. Local4Global- project stands for Systems of Systems that act locally for optimizing globally (Local4Global, 2013). One of its desired goal is to develop, comprehensively test and evaluate in real-life Traffic Systems of Systems (TSoS). In addition, the project needs to generate a generic, integrated and fully functional methodology for TSoS. The optimization method developed so far is demonstrated in two real scenarios: the climate control of a building and optimizing the traffic on a test site in the North of Munich. A traffic prediction project involving SoS techniques for smarter traffic predictions in collaboration with IBM for the city of Cologne, Germany (IBM Smart Traffic, 2010) was able to predict traffic volume and flow with over 90 percent accuracy up to 30 minutes in advance.

COBWEB - Citizens OBservatory WEB – is another project that is funded under the European Union's Seventh Framework Programme (FP 7) for developing community-based environmental systems using innovative and novel earth observations applications (Hodges, 2014). The projects major aim is to create a platform environment enabling citizens living under the biosphere reserves designated by UNESCO (United Nations Educational, Scientific and Cultural Organization) to collect environmental data using their mobile devices. FP7 is a large collaboration of experts from 13 partners and 5 countries. EU FP7 project, Road2SoS, has developed a roadmap of multi-site manufacturing SoS in order to explore the potential pathways to a future vision of a globally reconfigurable manufacturing SoS (Rauschecker, Ford, & Athanssopoulou, 2014). The aim is to have a global network of interoperable factories, permitting the dynamic allocation of manufacturing. GEOSS stands for global earth observation system of systems, aims to provide solutions for a number of problems around the world (Uhlir, Chen, Gabrynowicz, & Janssen, 2009). So far, it has been used in forecasting meningitis outbreaks, guarding biodiversity, and helping in improving climate observations in Africa and Central and South America. The environmental protection agency (EPA) in USA along with Group on Earth Observations (GEO) helps in advancement of GEOSS. GEOSS provides decision makers with correct and prompt scientific information for advancement of social benefits. Integrated Mobile Security Kit (IMSK) is used for assessing critical situations (Laudy, Petersson, & Sandkuhl, 2010). It helps to provide quickly an effective deployment of information fused with intelligence on mobile platforms for enhanced security. Some examples of its application are mass events such as football games and terrorism attacks. Lastly, the ministry of economics and technology in Germany sponsors Shared e-Fleet project (2013). It aims at higher utilization of systems electric vehicles so that they can be used commonly and very efficiently.

SoS has found applications in the field of emergency management response systems as well. An excellent paper using fuzzy logic and genetic algorithms describes the application of SoS methodology in post-disaster relief and recovery operation for an earthquake situation (Chandana, & Leung, 2010). The effectiveness of the proposed approach to disaster situation management is demonstrated using Chinese earthquake site. Liu (2011) also propose principles and rules for the design of an Emergency Management System of Systems (EMSoS) in China. A workshop entitled "Building a

Systems of Systems (SofS) for Disaster Management" was conducted in Australia by the CSIRO (Fraser, & Hawkins, 2014). CSIRO is known as Commonwealth Scientific and Industrial Research Organization. CSIRO is Australia's national science agency. The workshop aimed to underline a plan that would help in achieving situational awareness for natural disasters such as forest fires at a national extent.

2.3. SOS ACQUISITION PROCESS

The DoD 5000.2 is currently used as the acquisition process for complex systems. Schwartz (2010) described this process as an extremely complex systemic process that cannot always constantly produce systems with expected either cost or performance potentials. The acquisition in DoD is an SoS problem that involves architecting, placement, evolution, sustainment, and discarding of systems obtained from a supplier or producer.

Numerous attempts undertaken to modify and reform the acquisition process have found this problem difficult to tackle because the models have failed to keep pace with actual operational scenarios. Dombkins (1996) offered a novel approach to model complex projects as waves. He suggested that there exists a major difference in managing and modeling traditional projects versus complex projects. He further illustrated his idea through a wave planning model that exhibits a linear trend on a time scale; on a spatial scale, it tries to capture the non-linearity and recursiveness of the processes. In general the wave model is a developmental approach that is similar to periodic waves. A period, or multiple periods, can span a strategic planning time. The instances within the periods represent the process updates.

A recently proposed idea (Dahman, Lane, Rebovich, & Baldwin, 2008) that SoS architecture development for the DoD acquisition process can be anticipated to follow a wave model process. According to Dahman DoD 5000.2 may not be applicable to the SoS acquisition process. Acheson (2013) proposed that Acknowledged SoS be modeled with an Object-Oriented Systems Approach (OOSA). Acheson also proposes that for the development of SoS, the objects should be expressed in the form of a agent based model.

The environment and the systems are continuously changing. Let there be an initial environmental model which represents the SoS acquisition environment. As the SoS acquisition progresses through, these variables are updated by the SoS Acquisition Manager to reflect current acquisition environment. Thus, the new environment model at a new time has different demands. To fulfill the demands of the mission a methodology is needed to assess the overall performance of the SoS in this dynamic situation. The motivation of evolution is changes in the SoS environment (Chattopadhyay, Ross, & Rhodes, 2008). The environmental changes consist of:

- SoS Stakeholder Preferences for key performance attributes
- Interoperability conditions between new and legacy systems
- Additional mission responsibilities to be accommodated
- Evolution of individual systems within the SoS
- Capabilities of individual systems

The methodology for architectural evolution in SoS should be such that it addresses all the changes in the environment stated above.

2.4. ASSESSING SYSTEMS ARCHITECTURE

In principle, systems engineering may be thought of as a decision-making activity. The architecting process involves the hierarchical reduction of ambiguity where a set of alternatives is evaluated so that the most suitable alternatives are selected. SoS design problems are based on multi- objective functions for binary variables (Singh, 2011). The design is judged based on a number of key performance parameters that together form a highly non-linear hyper surface. These techniques were employed in this study. The multi-objective approach combines multiple objectives into the following single objective [13]:

Max $\mathbf{f}_k\ (\mathbf{x})^{\mathbf{T}}\ \forall\ k;\ \mathbf{g}_i\ (\mathbf{x})^{\mathbf{T}} \leq b_i\ \ \forall\ I;\ \mathbf{x}^T = \{\ x_1\ \ x_2\ \ \ldots\ \ x_n\ \}\ \boldsymbol{\epsilon}\ \mathbf{X}\ ;\ \mathbf{x}^T \geq 0$

x: vector of the variables; **f**: objective function(s); **g**: inequality constraints;

A solution to the multi-objective problem includes compromise that is acceptable to the decision maker with respect to all of the objectives pursued (Schutze, Lara, & Coello Coello, 2011).

2.5. HANDLING MANY OBJECTIVES

Multi-objective optimization algorithms are well known and fully developed for situations with two or three objectives. Coello (1999) gives a list of references on evolutionary multiobjective optimization. Some popular and established ways (Figure 2.5.) to solve such problems are weighted approach (Marler & Arora, 2010), goal programming (Deb, 1999), Pareto dominance (Horn, Nafpliotis, & Goldberg, 1994), ε–Pareto Dominance Optimization is applied to workflow grid scheduling (Garg & Singh, 2011), and ranking of objectives (Garza-Fabre, Pulido, & Coello, 2009).

Many objective optimization refers to conditions which more have than three objectives. Solving many objective optimization problems with the above listed methods can be difficult because nearly all solutions in a population grow into non-dominated, with increasing number of objectives. Secondly, the number of solutions required for approximation increases exponentially with the increase in dimensionality of the objective space (Schutze, Lara, & Coello Coello, 2011). As the number of objectives goes beyond five or more, the number of non-dominated solutions in a randomly generated population is more than 90% (He & Yen, 2014). The effectiveness of the recombination operators usually used in evolutionary algorithms is reduced (Deb & Jain, 2014).

Besides it is hard to visualize solutions in higher dimensional spaces, weakening in search ability of Pareto dominance based algorithms and a very high computational cost (Ishibuchi, Tsukamoto, & Nojima, 2008). Stochastic heuristic techniques such as evolutionary algorithms are often used to generate solutions and fuzzy logic may be used for assessing the fitness of these solutions (Agarwal, Pape, & Dagli, 2014). These techniques were employed in this study (Coello Coello, 2002).

Some methods to deal with many objective problems include using reference-point-based nondominated sorting approach (Deb & Jain, 2014), Pareto corner search evolutionary algorithm and dimensionality reduction (Singh, Issacs, & Ray, 2011).

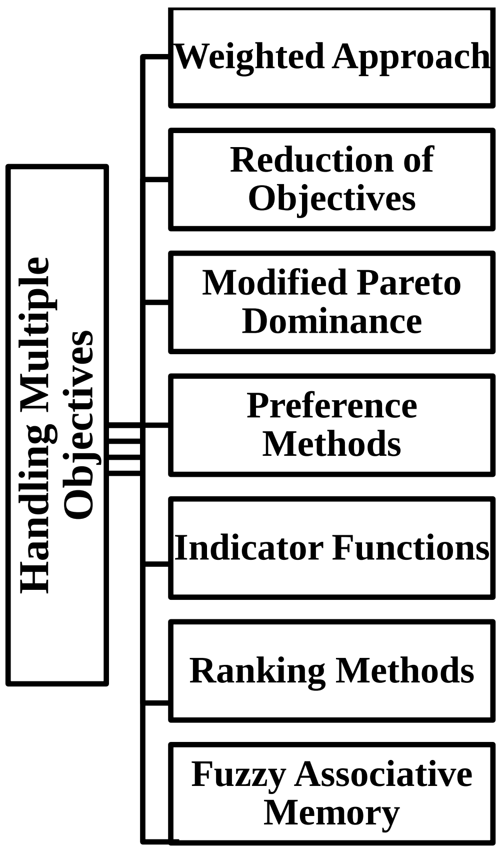


Figure 2.5. Different methods to handle multiple objectives in optimization

Other methods as listed in Figure 2.5. are objective reduction using linear and nonlinear algorithms (Saxena, Duro,Tiwari, Deb, & Zhang, 2013), designing a grid based evolutionary algorithm (Yang, Li, Liu & Zheng, 2013), fuzzy-based Pareto optimality (He & Yen, 2014), Borg multi-objective evolutionary algorithm (MOEA) proposes to combine all techniques such as ε-dominance, convergence speed measuring process called progress, random initialization, and auto-adaptive multi-operator recombination (Hadka & Reed, 2013), multiobjective optimization problem can be decomposed into a smaller number of scalar optimization sub-problems and then optimize them concurrently (Zhang, & Li, 2007), many researchers are using hypervolume indicator as a quality measure of the Pareto fronts (Bader & Zitzler, 2011) and besides there exist other performance metrics to compare Pareto fronts obtained by evolutionary algorithms (Yen & He, 2014).

2.6. AUTOMATED NEGOTIATIONS

The importance of studying negotiation is realizable in electronic commerce, and artificial intelligence. Negotiations have two major components viz the number of parties who are negotiating and the issues on which they are negotiating. Each party negotiates in its own interest to reach at least the same or a better outcome than the previous offer made to it (An, 2011). Cooperative negotiation has found uses in maintaining real time load of a mobile cellular network (Bigham & Du, 2003), modeling complex physiological phenomena (Gatti, & Amigoni, 2004, July) and resolving air traffic conflicts efficiently (Wollkind, Valasek, & Ioerger, 2004). A negotiation can occur between two individuals, or one individual negotiating with several individuals, and finally many individuals negotiating with many other individuals. These negotiations are called bilateral (Lin, Kraus, Wilkenfeld, & Barry, 2006), one-to-many (Rahwan, Kowalczyk, & Pham, 2002) and many-to-many (Nguyen, & Jennings, 2006) respectively.

A detailed classification of automated negotiations can be accessed from Buttner (2006). Automated negotiation is an integral part of systems across all domains (Jennings et.al, 2001). Automated negotiation can be defined as an iterative process of settling on an issue or multiple issues between the negotiating parties (Fatima, Wooldridge, & Jennings, 2002) as shown in Figure 2.6.

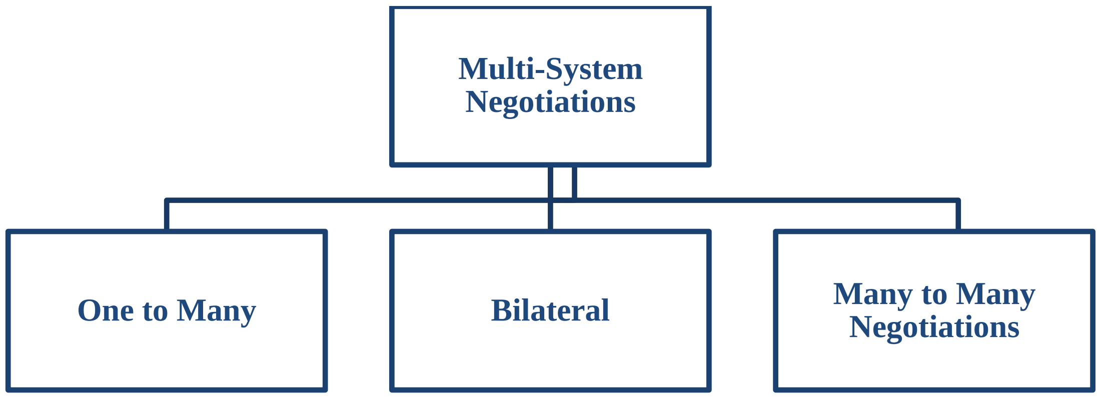


Figure 2.6. Automated negotiations protocol categories

According to (Zheng et al., 2013; Guttman & Maes, 1998) negotiation in multi-agents is a decision process for resolving multiple issues, which may or may not be mutually exclusive (refer to Figure 2.7.). Most of the current research is focused on assigning utility functions encompassing all issues or a function for each issue and then combining the utilities to estimate the overall benefit of an offer (Ito et al., 2009).

This assumption is usually with the utilities making the decision a linear problem, which is usually, not the case. The utility functions can be classified into linear and nonlinear. Agents that utilize linear utility functions can aggregate the utilities of the issue-values by weighted linear summation.

However, such an approach is considered naïve for modeling real world scenarios as aggregations are unrealistic. Multi-attribute utility theory (MAUT) (Dyer, 2005) believes that each outcome issue or attribute is independent. MAUT proposes to have a separate utility function for each of the issues. Although there have been studies that model pairwise attributes to capture the dependence among the variables (Siebert, 2010).

Besides the systems can exhibit diverse behaviors which cannot be estimated as functions and it is hard to predict their ranking of preference for a particular issue (Marsá-Maestre et.al., 2014). Game theory postulates negotiation as a non-zero sum game along multi-dimensional issues (Binmore & Vulkan, 1999). Multiple issue negotiations can be broadly categorized as separate negotiations where each issue is dealt individually by the negotiators, in simultaneous negotiations all issues are taken up together, where in sequential negotiations, a set sequence is assigned to the total issues and each issues is then taken up in that order (Fatima, Wooldridge, & Jennings, 2006).

The negotiation protocol describes the rules of encounter between the negotiation parties. A negotiation protocol can handle a single issue or multiple issues. The negotiation strategy is a specification of the sequence of actions (usually offers or responses) that the agent plans to make during negotiation. The solution space of negotiation strategies is very large. Strategies are usually based on the nature of the behavior of the agent and its opponent or teammate. Negotiation strategy tries to model the function (or a set of rules) for proposing the values of multiple issues at each point in time (refer to Figure 2.7.). The strategy used for a particular agent might turn out to be a poor choice for another.

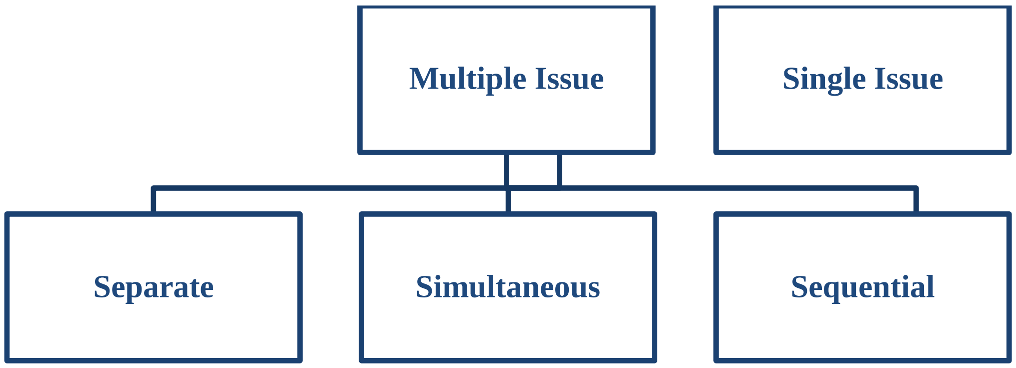


Figure 2.7. Categories of attributes in automated negotiations

A static approach can also decrease awards after a number of negotiations. Therefore, an agent can learn by adapting based on rewards, as opposed to trying to model the other agents.

Agents are classified based on information possessed at the time of negotiation into complete or partial information states. If the agent has the complete information of the environment, which includes the opponent agent's, negotiation strategy, the external factors that affect the negotiation and the effect of the agent's strategy on the opponent it is said that that agent is in a complete information state. Otherwise, if any information is unclear or missing the agent is assumed to be in a partial information state. Information in multi-agent systems are comprised of utility functions that the opponent agents use to evaluate various attributes, the reasoning models of opponent agents, and the constraints of opponent agents.

The better approach would be to calculate the opponent's behavior based on its previous offer, and then adapt the response accordingly (Chen & Weiss, 2013). Different adaptive strategies have been proposed earlier such as the ABiNeS: An Adaptive Bilateral Negotiating Strategy over Multiple Items for effectively handling different types of opponents (Hao & Leung, 2012). Other methods include game theoretic analysis (Jordan, Kiekintveld, & Wellman, 2007), use of genetic algorithms (Jian, Li-Chang, & Bo, 2008), differential evolution (Bi, & Xiao, 2012), Bayesian networks (Hindriks, & Tykhonov, 2008), neural networks (Carbonneau, Kersten, & Vahidov, 2008) and fuzzy logic (Luo, et al., 2003).

2.7. IDENTIFYING GAPS IN LITERATURE

The overall aim of this work is to ensure that SoS architecting process is concentrated on enabling a methodological insight which is requisite to provision knowledge-based decision making, throughout the acquisition process. The objective is also to enlarge our horizon to not only DoD based acquisitions but commercial acquisitions by making use of the this methodology of evolutionary architecting.

The methodology outlined in this research is a type of modeling approach to address various aspects of SoS acquisition environment: SoS architecture assessment, SoS architecture evolution, and SoS acquisition process dynamics including behavioral aspects of constituent systems. The major gaps are highlighted below and then further explained to elucidate the concepts:

- There are no validated and tested quantitative models for SoS architecture development
- The concept of meta-architecture has not been previously used in SoS architecture generation
- Architecture assessment methods previously suggested do not effectively handle the preferences amongst the various key performance attributes. Although some papers have recommended methods using type I fuzzy logic (Pape & Dagli, 2013) and computing with words (Singh, 2011).
- An integrated model that combines meta-architecture generation and negotiations with the stakeholders is also missing from the literature
- System behaviors have not been previously incorporated in SoS negotiation process
- The SoS architecture problem is a many objectives optimization challenge with over more than 20 objectives

The SoS architecting model proposed employs mathematical models. Model based engineering is a fundamental part of the systems engineering process by supporting design, evaluating architectural solutions, and enabling the assessment of the system performance. By using meta- architecture generation techniques, architecture quality assessment techniques and implementation through negotiation all three points are addressed. All the techniques are integrated in this dissertation to form a model that acts as a decision aid to the SoS manager.

There is a need within systems of systems, for making decisions to mold legacy systems, add new systems, and/or change the configuration of these systems and their interconnections (DeLaurentis, Crossley, & Mane, 2011). This requires both proper definition of the design problem and good analysis/synthesis (Yingchao, 2012). This need is addressed through the wave approach of architecture evolution that takes care of sequential decision making (Agarwal et al., 2015). This dissertation provides a series of quantitative techniques and pathways to add new capabilities and systems within a SoS. Besides the dissertation aims to provide an integrated model that can bring together the techniques to form a tool that aids in decision making.

Meta-architecture is a set of systems and interfaces selected to form a SoS based the KPAs of the problem domain. The problem of selection is posed a many-objective optimization problem. The objectives are the KPAs and the decision variables are the set of systems and interfaces. This concept helps in conveying the SoS architecture idea to the stakeholders at a very high level. It can be combined with various other executable architecting techniques to evaluate the efficacy of SoS before it is finally implemented.

Architecture assessment techniques previously proposed have not been able to capture the essence of non-linear tradeoffs existing amongst the various attributes. This problem is dealt by incorporating preferences in key performance parameters (KPP) for architecture assessment. The architectural issues can be converted to KPPs which later can be used as objectives for solving the architecting problem. The preferences among KPPs are accumulated through many stakeholders to counter any unforeseen circumstances. Also the fuzzy rules created through these preferences produce non-linear surface to capture the decision space that may be highly non-linear (Agarwal, Pape, & Dagli, 2014).

The method of SoS architecture generation proposed here gives selects the best possible architecture by using the KPP or key performance attributes. The new rules or modified rules of the fuzzy inference engine can be changed any time during the architecting process thus making it easy and les computationally expensive to fix. Individual attributes may not have a clearly defined, mathematically precise, linear functional form from worst to best. The goodness of one attribute may or may not offset the badness of another attribute. Several moderately good attributes coupled with one very poor attribute may be better than an architecture with all marginally good attributes, or vice-versa. A fuzzy approach allows many of these considerations to be handled using a reasonably simple set of rules, as well as having the ability to include non-linear characteristics in the fitness measure. The simple rule set allows small adjustments to be made to the model to see how seemingly small changes affect the outcome.

Another component has been to use the evolutionary algorithm based approach which helps in evaluating many architecture alternatives to achieve a near optimal architecture. Evaluation of architectures is another SoS challenge area as it lends itself to a fuzzy approach because the criteria are frequently non-quantitative, or subjective (Pape & Dagli, 2013), or based on difficult to define or even unpredictable future conditions, such as “robustness.”

The proposed integrated model combines meta-architecture generation and negotiations with the stakeholders. Several projects have not been able to achieve enough progress due to integration problems related with the complexity of software interfacing. By doing so, this research makes a valuable contribution to the existing systems engineering body of knowledge (SEBOK) (Pyster, Olwell, Squires, Hutchison, Enck, Anthony, 2014).

This work also addresses another gap by integrating software engineering with systems engineering principles (Agarwal, Pape, Ergin, & Dagli, 2014) as pointed out by Squires, Olwell, Roedler, & Ekstrom (2012). This is because a systems engineer should be able to comprehend the popular methods of software architecting and design patterns.

By having the ability to incorporate multiple systems behaviors and achieving an architecture through negotiation we are able to capture the emergent phenomenon in forming a SoS. It is often hard select the properties that do not correspond one systems or component alone but to the whole SoS. Thus by having multiple behaviors and negotiations we aim to create a SoS which can achieve the overarching capability through its emergent properties (Agarwal, Saferpour, & Dagli, 2014).

SoS architecting is where problems are solved by first creating the meta-architectures that involves multiple key performance parameters (KPP) producing a non-linear hypersurface. The optimization algorithm has to trace this hypersurface to find the global minima or maxima. This process is very computationally expensive and tedious. Fuzzy associate memories can be used as a way combining multiple objectives in to one non-linear surface with many dimensions (Agarwal, Pape, & Dagli, 2014).

Too many KPPs can pose a challenge to SoS architecture generation mechanism. Since the relationship amongst the KPPs is non-linear, together they forma non-linear hypersurface which is hard for the optimization algorithm to trace.

Resiliency can be termed as the capability to acclimatize in a dynamic environment (Schwind et al., 2013), and through self-organization can help the systems swiftly recuperate from any adversarial events and disturbances (Vaneman, 2014). Resilient SoS architectures have the ability to bounce back through major breakdowns in functional and physical architectures. They have a higher chance of recuperating and take less time to recover (Vaneman & Triantis, 2007) and (Vaneman, 2014).

There have been different metrics used to measure resilience such as failure node analysis (Han, Marais, & DeLaurentis, 2012) in SoS. Another concept is to endure the loss of performance in one component system by reorganizing the tasks among the remaining systems (Uday & Marais, 2014). Therefore as one node undergoes breakdown, other nodes can modify their tasks to compensate for this loss. These metrics are not able to capture the overall capability of resilience for a SoS.

By using key performance measures such as robustness and modularity we introduce new measures based on graph theory that can ensure that the SoS architecture is resilient. Lack of robust behavior in applications is one source of failures (Hagen, 2007). Similarly modularity is very essential in design of complex engineering systems (Baldwin & Clark, 2006). It helps in making the SoS controllable, allows multiple passageways for working, and endows the system the strength to handle systemic failures. Thus it can be said both robustness and modularity are crucial components of

SoS resilience. Robustness metric used in the work ensures the SoS architecture has the capability to withstand any disruption and modularity metric ensures the capability recuperate based no high modularity of the SoS graphical model.

Interoperability can be defined as the ability of systems, units, or forces to provide services to and accept services from other systems, units, or forces and to use the services so exchanged to enable them to operate effectively together (Tran, Douglas & Watson, 2005) & (Lane & Valredi, 2011).

The proposed model is able to measure current level interoperability which is usually a major concern in SoS, to manage protocols and interfaces in general as systems come and go in SoSs, and measure communication across a given set of ground control systems. Therefore we not only measure interoperability within systems but also communication capability across a given set of communication systems. These systems are designed for an intermediate communication channel in case the systems are unable to communicate directly amongst themselves.

The later sections give a detailed overview of various techniques mentioned to address the gaps identified in this section such as model validation and testing, meta-architecture generation, architecture assessment, an integrated model approach, system behaviors incorporation and many objective optimization challenges.

3. THE INTEGRATED MODEL

As the world becomes more complex, the large scale systems exhibit properties such as decentralization in authority and geographical independence. Such systems are composed of diverse and autonomous elements (Samad & Parisini, 2011). Many such systems can be categorized as Acknowledged SoS due to similarities in the characteristics shared amongst them. Acknolwedged SoS usually have the SoS objectives, management, funding and authority provided to participating systems. Systems also maintain their own management, funding and authority autonomously. This dissertation focusses on Acknowledged SoS as they occur in many real life applications, some of which are discussed here.

Smart Grids can be modeled as Acknowledged SoS due to their resemblance with its properties (Miller, Pogaru, & Mavris, 2013), supply chain management systems (Chan, H. K. (2011), a recent example has been the development of an internet architecture which reconfigures itself with change in its environment (Liu, Nishimura, & Umehara, 2012,) Guo (2009) suggests model-based techniques for automotive electronic system development which involve embedded systems. Department of Defense (DoD) has long been a proponent of Acknowledged SoS research and it utility in assessing security risks to critical missions (Dahmann, J., Rebovich, G., & Turner, G. (2014). Disaster resilience or disaster relief management response systems can also be modeled as Acknowledged SoS (Cavallo & Ireland, 2014).

3.1. DEFINING THE SOS PROBLEM

The model presented in this section is applicable to Acknowledged system of systems. The architecture of an SoS follows an evolutionary development cycle to achieve the overarching capability required by the SoS. Especially for the Acknowledged SoS this process is guided through a small fund allocation to create a larger capability which is operational over a finite cycle time.

Furthermore the constituent systems do not need to either acquiesce to SoS requests or officially report to an SoS manager. Instead they can negotiate in their best interests. The capabilities possessed by the legacy systems can be incorporated in the next evolution cycle depending on the requirements.

The SoS domain manager must identify the decisive set of systems (with their respective capabilities) that will help SoS achieve an overall goal/purpose. An SoS can be achieved by combing individual systems and developing certain required interfaces among them. A detailed description of various SoS types is already given earlier in section 2.1.

Some methods have been proposed to model the evolutionary development of SoS. This dissertation adapts the approaches suggested and the integrated model builds on one of these approaches.

Dombkins (1996) has offered a novel approach which models complex projects as waves. Dombkins (2013) suggests and illustrates that there exists a major difference in managing and modeling traditional projects versus complex projects. Wave planning exhibits a linear trend on a time scale and on a spatial scale it tries to capture the non-linearity and recursiveness of the processes.

Wave model in general is a development approach similar to periodic waves. A period or multiple periods can span a strategic planning time and within the periods, there are instances that represent the process updates. Recently Dahmann proposed that SoS architecture development for the DoD acquisition process, can be anticipated to follow a Wave Model process (Dahman et al., 2011). According to Dahman DoD 5000.2 may not be applicable to SoS acquisition process. This research builds on the approach of wave model for Acknowledged SoS architecting. The evolution of SoS over a period of time under various uncertainties is depicted in Figure 3.1.

The Figure 3.2. explains the evolution of SoS from one wave to the next. The first wave or Wave 1 has N systems and M capabilities initially. As the SoS transitions to the next Wave 2 it has now T systems and K capabilities as some capabilities from Wave 1are added or rejected. Some systems such as S2 and capabilities such as C2 are retained in the next wave. Let us illustrate the wave model of SoS development through evolution of a big city to a smart city.

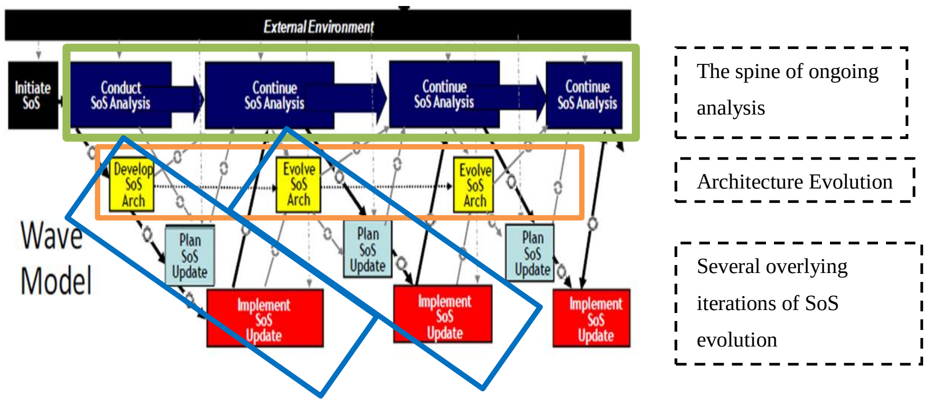


Figure 3.1. SoS wave model adapted from a figure

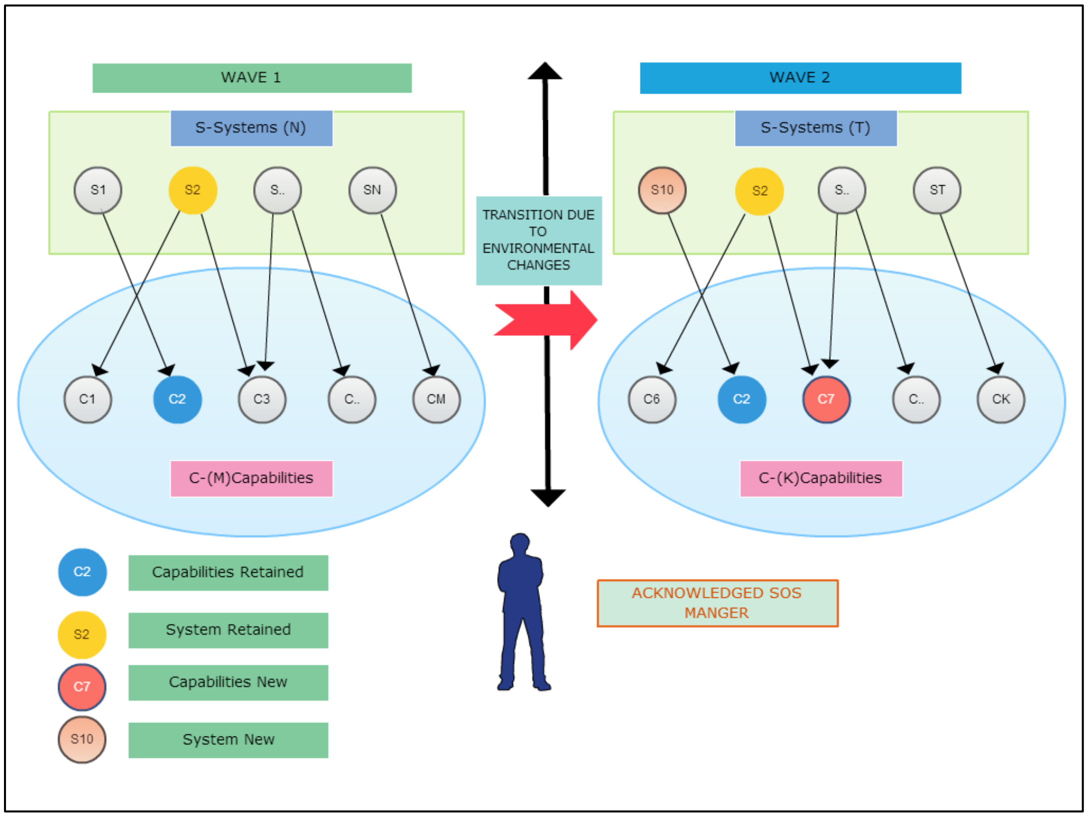


Figure 3.2. SoS transition into the next wave

Smart city can be described as a functioning large scale system where networked information is used to improve the living and day to day operations within city. Such operations cover a very broad domain: surfacing information to authorities, businesses, and citizens, optimizing energy and water pro (Celino, I., & Kotoulas, S. (2013).

Smart cities are very similar to an Acknowledged SoS where is there is a conscious effort made to develop a SoS. Similarly the various systems within the domain of a smart city such as Smart Grid, Smart Transportation, Smart Academic systems and so on acknowledge the Smart city objectives, have funding and management. Although individual systems operate autonomously yet they have shared interests. These systems have the knowledge and scope regarding the project and are merely guided by the smart city stakeholders. Smart city stakeholders work towards a comprehensive framework and have different viewpoints on multiple issues. The key issues in this case can be termed as the KPPs.

Just like the wave model of development Smart cities go through evolutionary phases. Based on the capabilities, current requirements, environmental changes, stakeholder views and performances the constituent systems are selected or left out in the next phase. To select the best set of systems and how they interface with each other to provide a network centric operation, optimization of resources is conducted at each step. The section below presents the general model for Acknowledged SoS architecting.

3.2 INTEGRATED MODEL VARIABLES AND PARAMETERS

The overall capability C (the overall goal) to be achieved by combining sub-capabilities):

c_j: $j \in \boldsymbol{J}, \boldsymbol{J}= \{1, 2, \ldots, M\}$: Constituent system capabilities j required to achieve C

s_i: $i \in \boldsymbol{I}, \boldsymbol{I}= \{1, 2, \ldots, N\}$: Candidate system i for the SoS

N:Total number of systems candidates

M:Total number of capabilities required

Let $\boldsymbol{A}$ be a N x $M - matrix\ of\ a_{ij}\ where$

$$a_{ij} = 1\ if \text{ capability } j \text{ is possessed by system } i$$

$$a_{ij} = 0\ otherwise$$

P_i: Performance of system i for delivering all capabilities

F_i: Funding of system i for delivering all capabilities

D_i: Deadline to participate in this round of mission development for system i

IF_{ik} *is the* interface between systems i *and* k *s.t.* $s \neq k, k \in I$

IC_i: The cost for development of interface for system i

OC_i: The cost of operations for system i

KP_r : $r \in \mathbf{R}$, $R = \{1, 2, \dots, Z\}$: The key performance attribute r of the SoS

FA: Funding allocated to SoS Manager

$p = \{1, 2, \dots, P\}$: number of negotiation parameters for bilateral negotiation

t_{max}: Total round of negotiations possible

t : Current round of negotiation (epochs)

$V_{pi}^{SoS}(t)$: The value of the attribute p for SoS manager at time t for system i

$V_{pi}^{S}(t)$: The value of the attribute p for system i owner at time t

TQ: Threshold architecture quality

3.2.1. Wave Model Processes. The wave model methodology provides for the evolution of the SoS needs, resources and environment over time while accounting for the differing approaches and motivations of the autonomous component system managers. The overall idea being to select a set of systems and interfaces based on the needs of the architecture in a full cycle called the wave.

Processes involved in the wave model can be explained through the first stage of Initializing the SoS **(**Dahmann, Rebovich, Lowry, Lane, & Baldwin, 2011). In terms of initializing, wave process requires SoS objectives and operational concept (CONOPS) and information on core systems to support desired capabilities. This basically starts with the overarching capability C desired by Acknowledged SoS manager and defining the c_j or sub-capabilities required to produce capability C and FA, funding allocated to SoS Manager. These also form the input for the participating systems s_i.

The second stage is called the Conduct SoS Analysis. For the wave process it represents starting an initial SoS baseline architecture for SoS engineering based on SoS requirements space, performance measures, and relevant planning elements.

The next step is the Develop/ Evolve SoS. In this case in terms of the Wave process essential changes in contributing systems in terms of interfaces and functionality in order to implement the SoS architecture are identified.

The next phase is Plan SoS Update in Wave process. In this phase the architect plans for the next SoS upgrade cycle based on the changes in external environment, SoS priorities, options and backlogs. There is an external stimulus from the environment which affects the SoS architecture.

Finally, the last stage in Wave process is Implement SoS Architecture which establishes a new SoS baseline based on SoS level testing and system level implementation.

The wave model has been implemented in Flexible and Intelligent Learning Architectures for SoS (FILA-SoS) version 1.0. This research hopes to improves certain models in version 1.0. This work aims to provide three independent models to be incorporated in version 2.0 that include, an alternative for meta-architecture generation based on swarm intelligence, a new architecture assessment technique based on type-II fuzzy logic systems, and a bilateral negotiation mechanism for SoS stakeholders based on clustering and machine learning techniques. Together the three models can help in designing an overall evolution strategy for complex adaptive SoS (CASoS).

3.2.2. Flexible and Intelligent Learning Architectures. The proposed model forms a part of the larger project called the Flexible and Intelligent Learning Architectures for SoS (FILA-SoS). FILA-SoS follows the Dahmann's proposed SoS Wave Model process for architecture development of the DoD acquisition process as depicted in Figure 3.1. FILA-SoS addresses the most important challenges of SoS architecting in regards to dealing with the uncertainty and variability of the capabilities and availability of potential component systems. The methodology also provides for the evolution of the system-of-system needs, resources and environment over time while accounting for the differing approaches and motivations of the autonomous component system managers. FILA-SoS assumes to have an uncertain and dynamic environment with fixed budget and resources for architecting SoS. The overall idea being to select a set of systems and interfaces based on the needs of the architecture in a full cycle called the wave. Within the wave there may be many negotiation rounds which are referred to as epochs. After each wave the systems selected during negotiation in the previous wave remain as part of the meta-architecture whilst new systems are given a chance to replace those left out as a result. The following paragraph explains the various stages in the wave model and how they are implemented in FILA-SoS.

The FILA-SoS has a number of independent modules that are integrated together for meta-architecture generation, architecture assessment, meta-architecture executable model, and meta-architecture implementation through negotiation (Figure 3.3.). The meta-architecture generation methods include fuzzy-genetic optimization (Pape, Agarwal, Giammarco & Dagli, 2014), multi-level optimization (Konur & Dagli, 2014), particle swarm optimization (Agarwal, Pape, & Dagli, 2014) and cuckoo search optimization (Agarwal, Wang, & Dagli, 2014). The architecture assessment method is based on type-1 fuzzy logic systems (FLS).

It is not possible to implement such meta-architecture without persuading the systems to participate, hence to address this issue a negotiation model is proposed based on game theory. The SoS negotiation protocol is based on game theory (Ergin, 2104). Individual systems providing required capabilities can use three kinds of negotiation models based on their negotiation strategies non-cooperative Linear Optimization model, cooperative fuzzy negotiation model, and Semi-cooperative Markov chain model (Dagli et al., 2013). Executable architectures are generated using a hybrid of Object Process Methodology (OPM) and Colored Petri Nets (CPN) (Agarwal, Wang, & Dagli, 2014), (Wang, Agarwal, & Dagli, 2014), and (Wang & Dagli, 2011).

Finally the process moves on to the next acquisition wave. The evolution of SoS should take into account availability of legacy systems and the new systems willing to join, adapting to changes in mission and requirement, and sustainability of the overall operation.

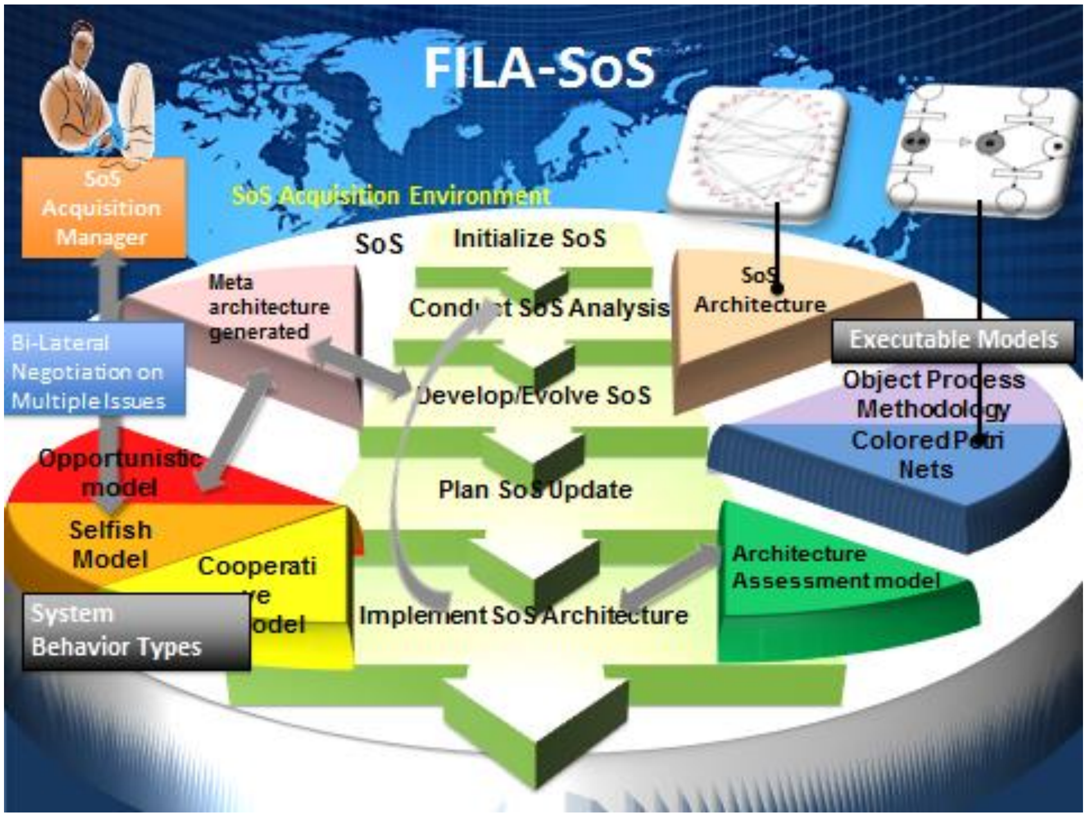


Figure 3.3. Overview of Integrated Model FILA-SoS Version 1.0

FILA-SoS is a novel method of making sequential decisions over a period for SoS development. FILA-SoS has a number of abilities that make it unique such as:

- Aiding the SoS manager in future decision making
- To assist in understanding the emergent behavior of systems in the acquisition environment and impact on SoS architecture quality
- To facilitate the learning of dynamic behavior of different type of systems (cooperative, semi-cooperative , non-cooperative)
- Identifying intra and interdependencies among SoS elements and the acquisition environment
- Modeling and application to a wide variety of complex systems models such as logistics and cyber-physical systems.

- Acting as a Test-bed for decision makers to evaluate operational guidelines and principles for managing various acquisition environment scenarios
- Appropriate to model SoS that evolve over a period of time under uncertainties by multiple wave simulation capability

The individual models presented in the previous paragraphs are part of the version 1.0 of FILA-SoS. The models are currently undergoing upgrades to answer and analyze SoS properties. The upgraded and new models will be incorporated in version 2.0 of FILA-SoS as shown in Figure 3.4.

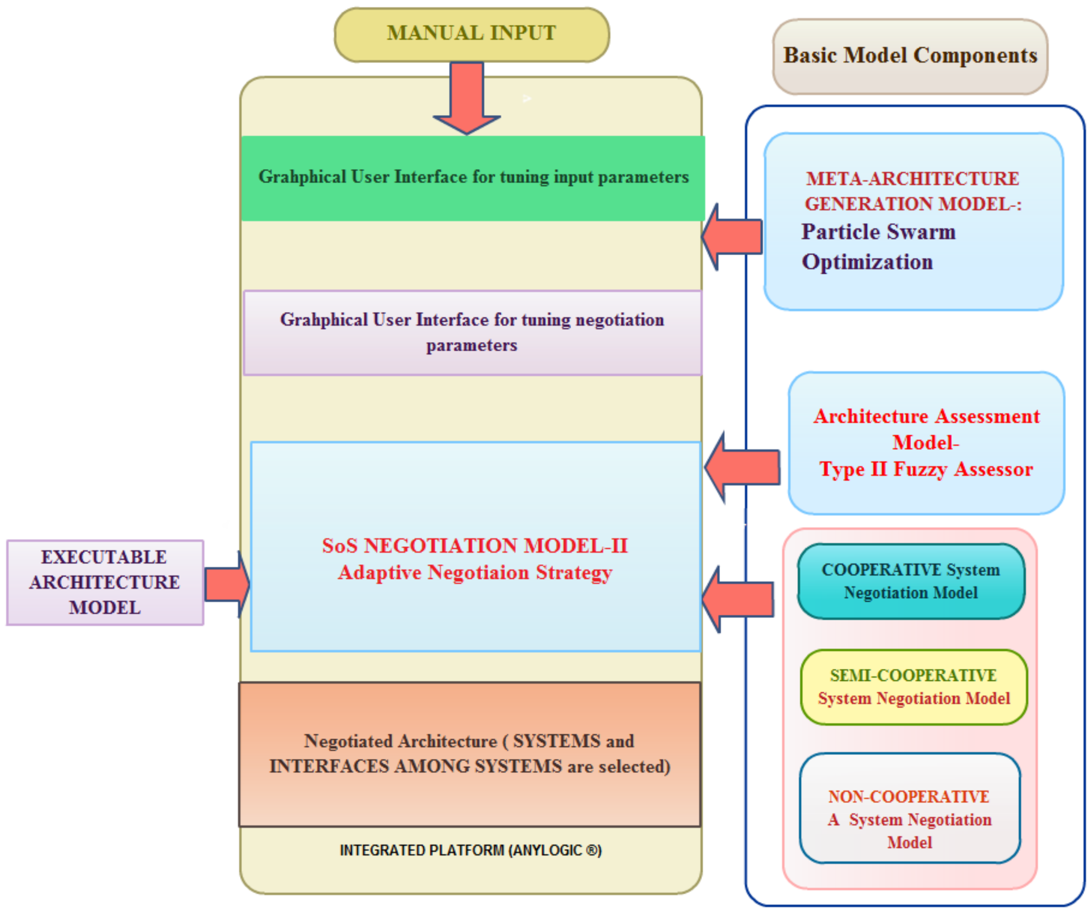

MANUAL INPUT
Basic Model Components
Grahphical User Interface for tuning input parameters
META-ARCHITECTURE GENERATION MODEL-: Particle Swarm Optimization
Grahphical User Interface for tuning negotiation parameters
Architecture Assessment Model- Type II Fuzzy Assessor
EXECUTABLE ARCHITECTURE MODEL
SoS NEGOTIATION MODEL-II Adaptive Negotiaion Strategy
COOPERATIVE System Negotiation Model
SEMI-COOPERATIVE System Negotiation Model
Negotiated Architecture (SYSTEMS and INTERFACES AMONG SYSTEMS are selected)
NON-COOPERATIVE A System Negotiation Model
INTEGRATED PLATFORM (ANYLOGIC ®)

Figure 3.4. Independent models used in FILA-SoS

FILA-SoS project spans 17 volumes (SERC, 2015). Each report describes the various aspects of the FILA-SoS integrated model.

The project reports span Volume 1 is the Integrated Model Structure report for FILA-SoS Version 1.0. It provides a short description of all independent models that make up the FILA-SoS integrated model. Integrated FILA-SoS developed is tested in three notional System-of-Systems, namely; Aircraft Carrier Performance Assessment, ISR (intelligence surveillance and reconnaissance) and SAR (search and rescue). FILA-SoS integrated model is currently being validated with a real life data from a medium sized SoS. The results of this validation are given in volume 17.

This dissertation aims to provide three independent models to be incorporated in version 2.0 that include, an alternative for meta-architecture generation based on swarm intelligence, a new architecture assessment technique based on type-II fuzzy logic systems, and bilateral negotiation mechanism for one SoS manager and many individual systems based on clustering and machine learning techniques. Together the three models can help in designing an overall evolution strategy for complex adaptive SoS (CASoS).

Firstly volume 2 describes Meta-Architecture Generation Multi-Level Model and volume 3 describes meta-architecture generation model known as the Fuzzy-Genetic optimization model. Both these models use a genetic algorithm to generate solutions. This dissertation proposes the use of a particle swarm optimization (PSO) algorithm. It has been recognized that GA is computationally expensive (Hassan, Cohanim, DeWeck, & Venter, 2005) and although PSO has the same efficiency as the GA but has a less computational cost attached to it.

Secondly for SoS architecture assessment, a type-1 fuzzy assessor has been used also described in Volume 4. This work extends the assessment technique by employing type-II fuzzy assessor.

Lastly, the SoS negotiation model is extended by incorporating a adaptive negotiation model.

It is named the Complex Adaptive System-of-System Architecture Evolution Strategy Model and is incorporated in FILA-SoS Version 2.0. This volume describes a computational intelligence based strategy involving meta-architecture generation through evolutionary algorithms, meta-architecture assessment through type-2 fuzzy nets and finally its implementation through an adaptive negotiation strategy.

The three models proposed in this research are described in the following section and are, Meta-Architecture formulation and generation, Meta-Architecture assessment and selection, and Meta-Architecture implementation through negotiation.

3.3. META-ARCHITECTURE FORMULATION AND GENERATION

Optimization algorithms can be categorized as gradient based and non-gradient based methods. Some of the non-gradient based methods include evolutionary algorithms (Horn, Nafpliotis, & Goldberg, 1994), swarm optimization (Engelbrecht, 2006), grid search (Bergstra & Bengio, 2012) and nonlinear simplex such as Nelder-Mead (Nelder & Mead, 1965). Evolutionary algorithm based techniques have proved to be useful for optimization problems with too many integer variables.

Meta-architecture is a set of systems and interfaces selected to form a SoS based the KPAs of the problem domain. The problem of selection is posed a many-objective optimization problem. The objectives are the KPAs and the decision variables are the set of systems and interfaces. Usually in a more than one objective optimization problem there is no single optimum but a set of non-dominated solutions (as explained in Section 2.5). solving such problems with more than three objectives turns it into a many-objective optimization problem. This problem is analyzed as a Pareto-Box problem (Köppen, Vicente-Garcia, & Nickolay, 2005).

3.3.1. The Pareto-Box Problem. A general approach for creating a Pareto solution can be expressed as follows:

- Let’s assume there are z objective functions to be optimized.
- The decision variables are expressed as a decision vector $\vec{x} = (x_1, x_{2,} \dots, x_n)$ in the decision space X.
- A function $f\colon X \rightarrow Y$ evaluates a specific solution expressed as a point in objective space Y.
- Assume the objective space to be a subset of the real numbers. That is $Y \subseteq R$.
- In a single-objective optimization problem, a solution vector $x^1 \in X$ is better than $x^2 \in X$ if $f(x^1) > f(x^2)$.

- In case of a vector-valued evaluation function, the vector $g: X \to Y$ and $Y \subseteq R^k$ where $g > 1$, to compare two solutions x^1 and x^2, the Pareto dominance is applied.
- An objective vector u, where $u = g(x^1) = [f_1(x^1), f_2(x^1), \dots, f_z(x^1)]$ dominates another vector v, where $v = g(x^2) = [f_1(x^2), f_2(x^2), \dots, f_k(x^2)]$ is expressed as $u \succ v$ if and only if $\forall i \in \{1, .., z\}, u_i \geq v_i, \wedge \exists i \in \{1, .., z\}: u_i > v_i$. This is in a maximization problem. In a minimization problem the signs of all the objective functions can be reversed and solved as a maximization problem.
- Accordingly a solution x^1 dominates x^2 $(x^1 \succ x^2)$ if $g(x^1) \succ g(x^2)$.
- The optimal solution in decision space can be expressed as $x^* \subseteq X$. Its image in objective space is $g^* \subseteq Z$.

The Pareto set X_E contains all optimal solutions also denoted efficient solutions. The Pareto front also denoted non-dominated frontier is the image of the Pareto set in objective space. The *Pareto Box problem* is explained further.

Given are x uniformly randomly selected y-dimensional points in the y-dimensional unit hypercube. If $e_x(y)$ denotes the expectation value for the size of the Pareto set of x randomly selected points in the $y - dimensional$ unit hypercube. Then, the following definitions hold (Köppen, Vicente-Garcia, & Nickolay, 2005):

Theorem 1. Given are x randomly selected points in the y-dimensional hypercube. For the expectation value of the size of the Pareto set of these x points we have the recursive relation:

$$e_{x-1}(y) = e_x(y) + \frac{1}{x} e_x(y-1) \; (x, y \geq 2)$$

(3.1)

which implies,

$$e_1(y) = 1$$

(3.2)

$$e_x(1) = 1$$

(3.3)

Theorem 2. The expectation value for the size of the Pareto set of $x \geq 1$ randomly selected points in the $y \geq 1$-dimensional hypercube is

$e_x(y) = \sum_{v=1}^{x} \frac{-1^{v+1}}{v^{y-1}} \binom{x}{v} \; \forall \, v \in \boldsymbol{V} = \{1,2, \dots., m\}$

(3.4)

Theorem 1 and 2 will help prove the central theorem 3 relating to limiting nature of the expectation values when there is an increase in number of sample points and increase in dimensions. For proofs of theorem 1 and 2 please refer to appendix of the paper (Köppen, Vicente-Garcia, & Nickolay, 2005).

Theorem 3. For fixed dimension $y > 1$ and the number of points $x \to \infty$ the expectation value $e_x(y) \to \infty$, the ratio of the non-dominated points ${e_x(y)}/{x} \to 0$ and for fixed $x > 1$ and dimension $y \to \infty$ the $e_x(y) \to x$

Proof.

$e_x(2) = \sum_{v=1}^{x} \frac{1}{\mathrm{v}} = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{m}$

(3.5)

Equation (3.5) is a harmonic series and has been proved divergent. Since the series is divergent meaning forever increasing it can be deduced from eq. (3.4) that for $n > 2$ the following condition will always remain true i.e. $e_x(y) \geq e_x(y-1) .. \geq e_x(2)$. Hence, as $x \to \infty$ the expectation value $e_x(y) \to \infty$. Besides as $x \to \infty$ and taking limits over the expression, ${e_x(y)}/{x} \to 0$. Similarly for the second part of the theorem, if x is fixed and $x > 1$ all terms in eq. (3.4) tend to zero as $y \to \infty$ except when $v = 1$. Because when $v = 1$, then since $1^{\infty} = 1$ the total term equals x or $e_x(y) \to x$.

As the dimensionality of the solution space increases, the probability of finding any dominated solution will decrease exponentially. This means that the Pareto set of x points will contain nearly all x points. This can also be expressed as for increasing number of sample points in the solution space, the number of non-dominated points will increase as well.

In a SoS architecting problem, component systems have multiple intra and inter system trade-offs that cannot be fitted into the mold of a single objective. Secondly, the number of solutions required for approximation increases exponentially with the dimensionality of the objective space (Shutze, Lara, & Coello, 2011). The SoS architect's aim is to maximize or minimize all the objective functionsKP_r, as the case may be.

The SoS optimization problem can be formulated as follows:

Optimize $\boldsymbol{F} = \left\{ f_{KP_1}(\boldsymbol{s}, \boldsymbol{IF}), \dots, f_{KP_r}(\boldsymbol{s}, \boldsymbol{IF}),\ \dots f_{KP_Z}(\boldsymbol{s}, \boldsymbol{IF}) \right\}\ \forall\, r = \{1, 2, \dots, Z\}$

where $f_{KP_r}(\boldsymbol{s}, \boldsymbol{IF})$ is the value of the key performance attribute r for decision variables $\boldsymbol{s}$ and $\boldsymbol{IF}$.

Subject to $\sum_i s_i\, a_{ij} \geq 1 \qquad \forall\, j \in \boldsymbol{J}$ (3.6)

$IF_{ik} = \{1\} \leftrightarrow \{ s_i = 1\ \wedge s_k = 1\} \qquad \forall\, i, k \in \boldsymbol{I}$ (3.7)

$a_{ij} \in \{0,1\} \qquad \forall\, i \in \boldsymbol{I}$ (3.8)

$s_i \in \{0,1\} \qquad \forall\, i \in \boldsymbol{I}$ (3.9)

$IF_{ik} \in \{0,1\} \qquad \forall\, i, k \in \boldsymbol{I}$ (3.10)

This is a Z dimensional muti-objective optimization problem. Constraints guarantees that at least one system for each capability is selected. Constraints also make sure that an interface between two systems selected if and only if the two systems are selected in the meta-architecture. Other constraints give the binary decision variables. Similar problem has been solved earlier as a multi-level bi-objective optimization (Konur & Dagli, 2014) using gradient based methods. The bi-objective model cannot handle many objectives of the general model described. There are two basic issues that need to be addressed here, namely ambiguity in the definition of the KPA, number of objectives and NP completeness of the mathematical model formulated. In this research evolutionary algorithms (EA) that use non-gradient descent optimization procedures are selected to deal with the NP completeness issues, fuzzy logic is used to represent the ambiguity in KPA and fuzzy inference is used to accommodate many objectives in formulating the fitness function. Fuzzy logic also helps in helping in the search ability of EA since search ability decreases with increasing objectives (Ishibuchi, Tsukamoto, & Nojima, 2008). Hence the above model is converted to a form where any EA can be used. Each individual chromosome is coded as a finite length vector of variables. The possible values of the variables denote the size of the alphabet. In this case the size of the alphabet is two because s_i and IF_{ik} are the binary decision variables. The details of the steps of chromosome representation are as follows.

Chromosome Representation: The chromosome is made up of two parts combined together to form a long string. The length of the individual chromosome is $L_{ch} = L_s + L_{if}$. L_{ch} is the length of the chromosome, $L_s$is the first part made by vector ***s*** as shown in Figure 3.5. The second part or L_{if} is made by linearizing the matrix $\boldsymbol{IF}$ as shown in Figure 3.6. and the full chromosome is shown in Figure 3.7. The architecture can be described as an undirected graph shown in Figure 3.8.

s_1	s_2	s_i	$\cdots$	s_N
Systems L_s=N				

Figure 3.5. A solution in the form of a string containing systems

IF 1 with 2	IF 1 with 3	IF 1 with N	IF 2 with 3	$\cdots$	IF i with k	$\cdots$	IF (N-1) with N
Interfaces $L_{Inf} = N * (N - 1) / 2$							

Figure 3.6. A solution in the form of a string containing interfaces

s_1	$\cdots$	s_i	$\cdots$	s_N	IF i with k	$\cdots$	IF (N-1) with N
Systems and Interfaces $L_s + L_{Inf} = N + N * (N - 1) / 2$							

Figure 3.7. A solution containing both systems and interfaces

With N participating systems the total number of variables become$(N + N * (N - 1) / 2)$. The solution string is binary in nature wherein a one represents the presence and a zero means the absence of a system or interface. This representation can be used to solve this problem with evolutionary algorithms, evolutionary strategies

(Beyer & Schwefel, 2002), swarm optimization or differential evolution (Storn & Price, 1997).

The general outline of EA consists of these steps (Back & Schwefel, 1996):

```
“ $t = 0$;
$Initialization\ \ P(0) = \{\overline{a}_1(0), \ldots., \overline{a}_\mu(0)\}, \in \boldsymbol{I^\mu}$
$Evaluation\ \ P(0) = \{\boldsymbol{\phi}(\overline{a}_1(0)), \ldots., \boldsymbol{\phi}(\overline{a}_\mu(0))\}$;
$While\ (termination\ condition\ for\ P(t) \neq true)\ do$
        $Recombination\ \ P'(t) = r\boldsymbol{\theta_r}(P(t))$;
        $Mutation\ \ P''(t) = m\boldsymbol{\theta_m}(P'(t))$;
        $Evaluation\ \ P''(t) = \{\boldsymbol{\phi}(\overline{a}''_1(0)), \ldots., \boldsymbol{\phi}(\overline{a}''_\mu(0))\}$;
        Selection $of\ indivdulas\ P(t+1) = s\boldsymbol{\theta_s}(P''^{(t)} \cup Q)$;
        $t = t + 1$;
End do; ”
```

Initially the generations are set to be zero. Then an initial population $P(0)$ of size μ is created with individuals represented by $\overline{a}$. The solutions or individuals are referred to as the chromosomes. Each individual in the population is evaluated by an objective function $\boldsymbol{\phi}$ to calculate the fitness value. Each of the consequent generations is created iteratively by applying operations, on the current population, that include recombination operator $r\boldsymbol{\theta_r}$, and mutation operator $m\boldsymbol{\theta_m}$. This process is run until the termination criterion is met and the algorithm stops creating new generations. The new individuals in the next generation have a new size γ. The new population $P''(t)$ is evaluated using the objective function $\boldsymbol{\phi}$. The selection process $s\boldsymbol{\theta_s}$ selects some individual of size μ to create the population for the next generation where $t = t + 1$.”

With respect to the problem at hand the decision variables are $\boldsymbol{s}$ and $\boldsymbol{IF}$. Recall that s_i *and* IF_{ik} are the binary decision variables in SoS. *Chromosome Initialization* will involve generating random binary values in all bits to start the population. *Fitness assessment* for a meta-architecture is explained in the following section 3.3 where this population is evaluated for Z objectives. *Termination criteria* should be such that algorithm should not converge prematurely. Whereas the termination was based on a minimum number of generations until the best solution quality does not change. Other techniques for termination include a hitting a bound on the threshold quality of solution.

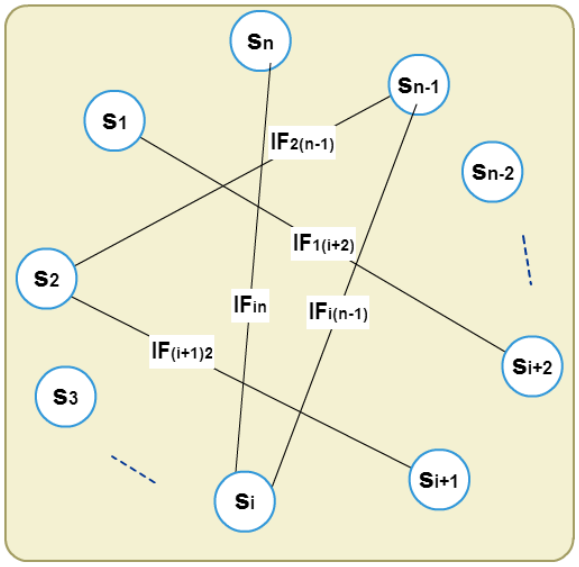


Figure 3.8. SoS meta-architecture as an undirected graph

3.4. META-ARCHITECTURE ASSESSMENT AND SELECTION

In the previous section a methodology for generating the solution was explained. Now to determine the quality of the solution (SoS architecture) a technique is needed to assess it. The technique should be generic enough to be applied to many independent domains. For this the objective function is converted to fitness functions for population based algorithms. Architecture assessment is based on KPAs which are selected based on the domain of the problem. Multiple objectives produce a non-linear hypersurface. The optimization algorithm has to trace the surface to find the global minima or maxima. This process is very computationally expensive and tedious. Fuzzy associate memories can be used as a way combining multiple objectives in to one non-linear surface with many dimensions (Agarwal, Pape, & Dagli, 2014).

The first problem is dealing with ambiguity in calculating the values of various objectives. This situation is dealt by using type-1 fuzzy systems.

Secondly, a method is needed to manage the preferences between KPAs in the fitness function. A tradeoff exists between the KPAs. This tradeoff is often non-linear and depends on a number of stakeholders of the architecture. Usually the tradeoffs are aggregated linearly through utility functions. For example, if two KPA's are scalability and reliability. The tradeoff could be higher reliability and low scalability. Besides the tradeoffs depend on a group of stakeholders which include system architect, project manager, customers and so on. Some methods such as fuzzy Pareto dominance (He & Yen, 2014), ranking of alternatives (Wang & Yang, 2009), fuzzy goal programming (Hu, Teng, & Li, 2007), weighting the objectives (Marler & Arora, 2010) have been used previously to combine them in to a single objective. Fuzzy associative memory helps capture the non-linearity that exists between the KPAs and can accommodate the view of multiple decision makers at the same time.

The third key factor is that the assessment techniques should be able to bring in performance attributes requirements from a lower level of abstraction. Often there is a difficulty in assigning actual numerical values to the KPA because the needs and requirements are expressed as words by the stakeholders. For example an attribute such as *net-centricity* can be broken down into *interoperability* and *command & control communication support* capability. Some of the prominent methods to assess the architectures include the use case maps (UCM) (Folmer, van Gurp, & Bosch, 2003), Architecture Tradeoff Analysis Method (ATAM) (Kazman et. al, 1998), and Scenario based Architecture Analysis Method (SAAM) (Kazman, Abowd, Bass, & Clements, 1996). There have been comparisons of architecture evaluation methods to choose the correction option effectively (Babar, Zhu, & Jeffery, 2004)

A beneficial approach would be to not only capture the tradeoffs points between as many possible KPAs in a nonlinear fashion, be able to compute with words, incorporate multiple views from stakeholders and help in value aggregation from different levels of abstraction of each KPA.

None of the methods discussed above are able to address the issues described above. The domain independent method proposed here for a domain dependent architecture value aims to fill this gap in literature. The proposed assessment model is based type-II fuzzy inference engine. Please refer to section 3.4.1.2 for more discussion on importance of type-II fuzzy sets. The values provide more realistic assessment of the SoS architecture’s quality. The attributes will be domain adjusted and selectable, using guidance from subject matter experts.

As the reader may recall the architecture is described as a chromosome. The fuzzy assessor based assessment is used to evaluate the fitness of the chromosome during the meta-architecture generation process. This assessor can be also used to evaluate the architecture after the negotiation. The concepts of fuzzy logic systems (FLS) are explained below to understand the working of the assessor.

3.4.1. Introduction – Fuzzy Logic. Crisp sets are those where an element is either a member of the set or not. Fuzzy logic (Zadeh, 1965) is an approach where a membership of the elements of a set is not true or false but is based on degrees of truth. A membership function (MF) is a curve that defines how each point in the input space is mapped to a membership value (or degree of membership) between 0 (not an element of the set) and 1(a member of the set). The input space is sometimes referred to as the universe of discourse. Let $\boldsymbol{U}$ be the universe of discourse which contains all the possible elements of concern in each particular context. Defining a fuzzy set $\boldsymbol{A\ in\ U}$: Fuzzy set $\boldsymbol{A\ in\ U}$ can be represented as a set of ordered pairs of a generic element $\boldsymbol{x}$ and its membership value,

$A = \{(x, \mu_A(x)) | x \in U\}$ such that $\mu_A(x) \rightarrow \{0,1\}$ (3.11)

where $\mu_A(x)$ is a degree of membership function of x in A and U is a universe of discourse.

Definition 1:

When U is continuous, A is commonly written as

$A = \int_{x\in U}^{U} \mu_A(x))|x$ (3.12)

where the integral sign does not denote integration, it denotes the collection of all points $x \in U$ with the associated membership function $\mu_A(x)$.

Definition 2:

Support: the support of a fuzzy set A in the universe of discourse U is a crisp set that contains all the elements of U that have nonzero membership values in A, that is,

$Supp\ (A) = \{x \in U | \mu_A(x) > 0\}$ (3.13)

Definition 3:

An $\alpha - cut$ of a fuzzy set A is a crisp set A_α that contains all the elements in U that have membership values in A greater than or equal to α.

Definition 4:

Fuzzy sets A and B are equal if and only if

$$A_\alpha = \{x \in U | \mu_A(x) > \alpha\} \quad \forall\, \mu_A(x) = \mu_B(x) \tag{3.14}$$

Example 1

Continuous Example: Let U be the interval [0,100] representing the reliability of a system-of-systems. Then we may define fuzzy sets “Poor” and “Excellent” as membership functions shown in Figure 3.9.

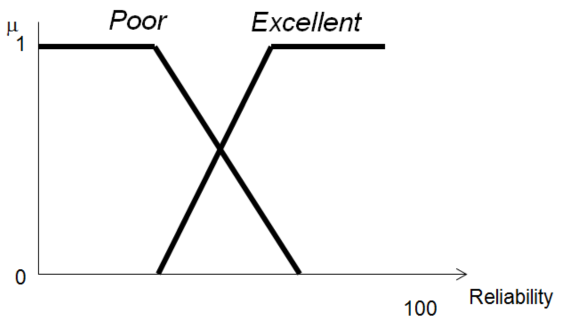


Figure 3.9. Membership functions for reliability

Definition 5:

The union of A and B is a fuzzy set in U, denoted by $A \cup B$ whose membership function is defined as $\mu_{A\cup B}(x) = \max[\mu_A(x), \mu_B(x)]$ (3.15)

Definition 6:

The intersection of A and B is a fuzzy set $A \cap B$ in with membership function

$$\mu_{A\cap B}(x) = \max[\mu_A(x), \mu_B(x)] \tag{3.16}$$

3.4.1.1 Type-I fuzzy logic system. Type-1 fuzzy set (T1 FS) theory was originally introduced by Zadeh (1965). Some of the applications include control theory

(Tzafestas, 1994), artificial intelligence (Hüllermeier,2005), and forecasting (Song, & Chissom, 1993). A typical Type-1 FLS has a fuzzifier, a rule section, fuzzy inference engine (FIS) and a defuzzifier or output processor. Figure 3.10. depicts the illustration of a type 1-FLS.

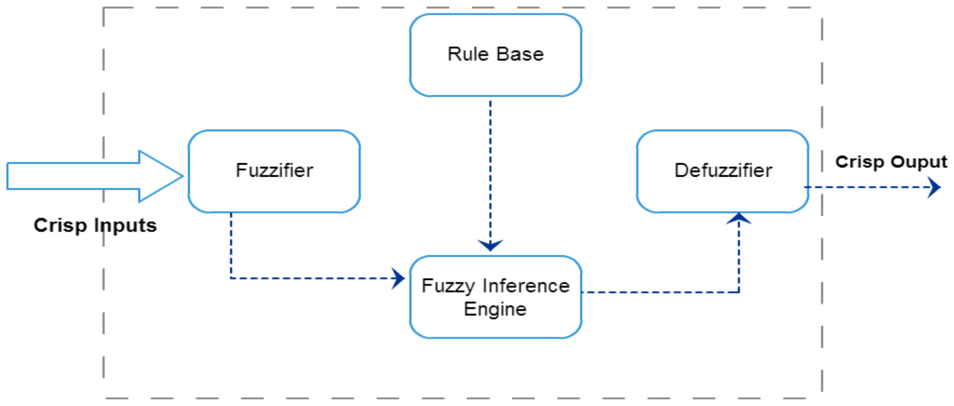


Figure 3.10. Overview of type-1 FLS

Fuzzy sets can be described as points in the unit hypercube $I^n = [0,1]^n$ (Kosko, 1992). A crisp value lies on the corner of the unit hypercube. A fuzzy system is a transformation S: $I^n \rightarrow I^m$ that maps fuzzy sets in I^n to fuzzy sets in I^m. These continuous fuzzy systems behave as associative memories. A fuzzy associative memory (FAM) contains a matrix of fuzzy values which can map an input fuzzy set into an output fuzzy set followed by an appropriate superimposition operator (Chung & lee, 1996). The rules are able to express a non-linear relationship between the variables. The process is explained through a simple example.

Example 3

The problem is to calculate the *architecture quality* of a system. For the sake of ease two inputs, *reliability* and *cost* are considered. The linguistic values for *reliability* are '*low*', '*medium*' and '*high*'. The linguistic values for *cost* are '*cheap* and '*expensive*'. The choice of membership function is up to the user based on the domain of the problem, experience and computational difficulty. The membership function for *reliability* and *cost* in the universe of discourse, U, is given below in Figure 3.11. and 3.12.

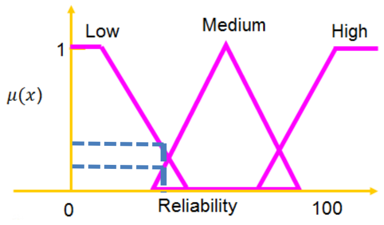


Figure 3.11. The membership functions for reliability

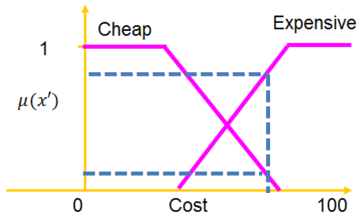


Figure 3.12. The membership functions for cost

The linguistic values for *architecture quality* are '*risky*, '*modest'*, and '*excellent'*. The membership function for *architecture quality* in the universe of discourse, U, is shown below in Figure 3.13.

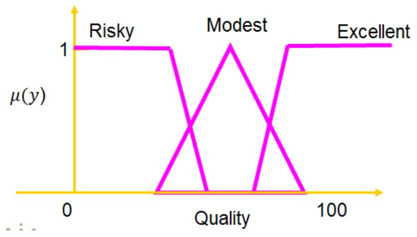


Figure 3.13. The membership function for architecture quality

Step 1

The first process involves converting the crisp inputs into fuzzy sets. This is called the fuzzification process. The inputs are reliability = 35 and cost = 80. The fuzzy values for these crisp values by using the membership functions of reliability as shown in the figure by dotted lines are:

$$\mu_{\text{reliability=low}}(35) = 0.3$$
$$\mu_{\text{reliability=medium}}(35) = 0.2$$
$$\mu_{\text{reliability=high}}(35) = 0$$

The fuzzy values for crisp values of cost are obtained by membership functions of cost in Figure 3.10 as

$$\mu_{\text{cost=cheap}}(80) = 0.1$$
$$\mu_{\text{cost=expensive}}(80) = 0.8$$

Step 2

After obtaining the fuzzy values from crisp inputs rules are needed to arrive at the final fuzzy output value. This is called the rules evaluation process. The rules for this problem are as follows:

"If the reliability is *low* or cost is *expensive*, then the quality is *risky*."

"If the reliability is *medium* and cost is *cheap*, then the quality is *modest*."

"If the reliability is *high* or cost is *cheap*, then the quality is *excellent*."

Definitions 5 and 6 are used in the rules containing disjunctions, OR and AND using the max and min operator. Each rule is evaluated below for explanation of the concept:

Rule 1

$$\mu_{quality=risky}(y) = \max[\mu_{low}(35), \mu_{expensive}(80)]$$
$$\mu_{quality=risky}(y) = \max[0.3,0.8] = 0.8$$

Rule 2

$$\mu_{quality=modest}(y) = \min[\mu_{medium}(35), \mu_{cheap}(80)]$$
$$\mu_{quality=modest}(y) = \max[0.1, 0.2] = 0.1$$

Rule 3

$$\mu_{quality=excellent}(y) = \max[\mu_{high}(35), \mu_{cheap}(80)]$$
$$\mu_{quality=excellent}(y) = \max[0, 0.1] = 0.1$$

To get the fuzzy values of the outputs, the FLS has to use fuzzy inference engine. Mamdani (1977) presented a method to synthesize the rules in fuzzy logic control. The Mamdani operator can be expressed as:

$\varphi(\mu_A(x), \mu_B(y)) = \mu_A(x) AND\ \mu_B(y)$=min$[0.8, \mu_{quality=risky}(y)]$

To defuzzify the outputs we use the center of gravity method. This process is called the defiuzzification. The center of gravity of the areas defined by the rules is the final defuzzified answer. There are many other methods such as BOA (bisector of area), CDD (constraint decision defuzzification), COA (center of area) and so on. In center of gravity method we take the output from each contributing rule, and then we add them. The centroid of the region is calculated as:

$$COG = \frac{\sum_{x=a}^{b} \mu_A(x) * x}{\sum_{x=a}^{b} \mu_A(x)}$$

The calculation for COG is shown as follows:

$$\frac{(0 + 10 + 20) * 0.8 + (30 + 40 + 50 + 60) * 0.2 + (70 + 80 + 90 + 100) * 0.5}{0.8 * 3 + 0.1 * 4 + 0.1 * 4}$$
$$= 71.8$$

It means there is 71.8 % of chance of systems quality.

In relation to this model architecture evaluation methods have been developed (Pape & Dagli, 2013) to assess robustness of SoS architectures. In addition, type-1 fuzzy associative memory has been developed to evaluate SoS architectures (Pape et al., 2013). The attributes used for evaluation were Performance, Affordability, Developmental Flexibility, and Operational Robustness. Type-1 fuzzy sets are able to model the ambiguity in the input and output variables. But type-1 fuzzy sets are insufficient in characterizing the uncertainty present in the data. Type-2 fuzzy sets proposed by Zadeh can model uncertainty and minimize its effects in FLS (Mendel & John, 2002). The next section gives a brief overview of type-2 and interval type-2 fuzzy sets.

3.4.1.2 Type-2 fuzzy sets. The cause of uncertainties in type-1 FLS includes the following:

1. Different people might interpret different meanings to the same words being used in antecedent and consequent rules
2. There is often uncertainty present in the input data which is not a single crisp value but has a given distribution if a group of decision makers are involved
3. Similarly the outputs may not have a singleton value but a distribution over which the outputs range due to multiple experts

These gaps are not addressed by type-1 fuzzy because their membership functions are totally crisp. Whereas, type-2 fuzzy sets are able to model such uncertainties due to the fact that their membership function are fuzzy themselves and are three-dimensional in nature. The structure of rules in a type-1 FLS and a type-2 FLS is the same, but in type-II the antecedents and the consequents are represented by type-2 fuzzy sets. A type-2 FLS contains a fuzzifier, a rule base, a fuzzy inference engine, and an output processor. The output processor includes type-reducer and defuzzifier. The type reducer reduces the type-2 FS to a type-1 FS whereas the defuzzifier converts the type-1 FS to a crisp number. The structure of the type-2 fuzzy associative memory maps inputs to type-2 fuzzy terms. Rules are made to describe the relationship between inputs and output using the linguistic terms of each input’s membership functions.

Type-2 FLSs are computationally demanding because of type-reduction. Interval type-2 (IT2) FSs (Liang & Mendel, 2000) are a special case of type-2 FSs extensively used for their less computational cost. IT2 FSs are often useful when there is an uncertainty involved in determining the exact membership functions, or when there are multiple stakeholders’ opinions on the same fuzzy variable (Wu, 2013). A general procedure for IT2FS is illustrated in the Figure 3.14. It is similar to type-1 FS, except fuzzifier converts the crisp inputs to IT2 FS, the outputs of the inference engine are IT2 FSs, there is another element called the type-reducer which converts the IT2FS values to type-1 FS before passing them to the defuzzifier.

An example of an IT2 FS, $\tilde{Y}$, is shown in Figure 3.15. A type-2 FS has two membership functions hence for each value of the linguistic variable the membership degree is not a number but an interval. This is because a straight line parallel to membership axis will cut the membership functions at two places. One of them will be lower forming the lower interval and the other one will form the higher interval of the degree. The two membership functions are denoted by $\bar{Y}$ (upper MF) and $\underline{Y}$ (lower MF). The area between them is the footprint of uncertainty (FOU).

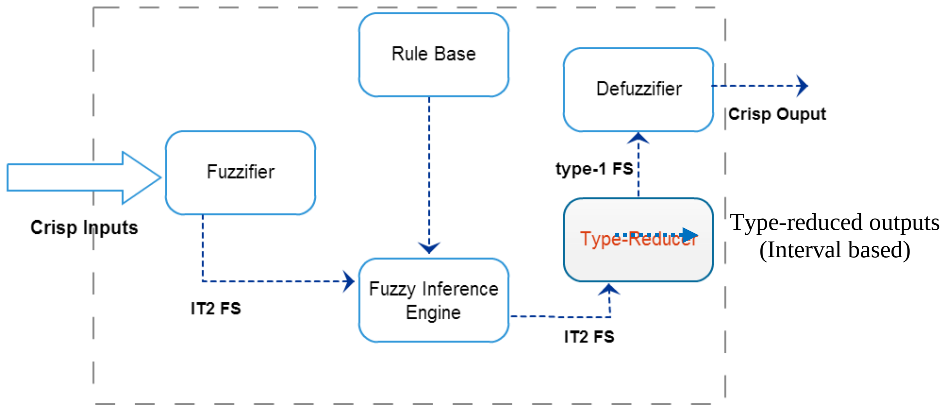


Figure 3.14. Overview of type-2 FLS

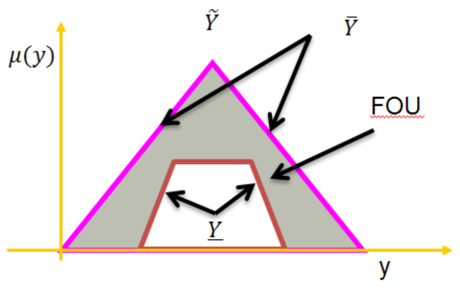


Figure 3.15. Membership function for a type-2 FLS

Given $\tilde{Y}_1^n$ are IT2 FSs antecedents or inputs, and $Z^n = [\underline{z}^n, \overline{z}^n]$ interval of a consequent output where $n = 1,2,\dots,N$ and $k = 1,2,\dots,K$

The steps in an IT2 FLS are demonstrated as follows:

1. Consider the rule base of an IT2 FLS comprising of N rules assuming that the nth rule is :
    a. IF y_1 is $\tilde{Y}_1^n$ and….. and y_K is $\tilde{Y}_K^n$, THEN z is Z^n
2. Calculate the membership of all inputs in the vector $\boldsymbol{y'} = (y'_1, y'_2, \dots y'_K)$ on each $\tilde{Y}_1^n$ for $n = 1,2,\dots,N$ and $k = 1,2,\dots,K$
    a. Membership is $[\mu_{\underline{Y}_k^n}(y'_k), \mu_{\overline{Y}_k^n}(y'_k)]$
3. When the nth rule $H^n(\boldsymbol{y'})$ for the input vector, fires the output interval can be computed as:

$$[\underline{h}^n, \overline{h}^n] = [\mu_{\underline{Y}_1^n}(y'_1) \times \dots \times \mu_{\underline{Y}_K^n}(y'_K), \mu_{\overline{Y}_1^n}(y'_1) \times \dots . \times \mu_{\overline{Y}_K^n}(y'_K)]$$

There are methods other than taking the product (Liang & Mendel, 2000) Center-of-sets (CoS) method for acting as a type-reducer (Mendel & John, 2002) has been used here for type-reduction from type-2 to type-1 fuzzy sets (Mendel & Wu, 2010).

$$Z_{CoS}(y') = \bigcup_{\substack{h^n \in H^n(y') \\ z^n \in Z^n}} \frac{\sum_{n=1}^{N} h^n z^n}{\sum_{n=1}^{N} h^n} = [z_l, z_r]$$

The lower z_l and upper limits z_r of the outputs can be calculated as follows.

4. $$z_l = \underset{x \in [1, N-1]}{min} \frac{\sum_{n=1}^{x} \overline{h}^n \underline{z}^n + \sum_{n=x+1}^{N} \underline{z}^n \underline{h}^n}{\sum_{n=1}^{x} \overline{h}^n + \sum_{n=x+1}^{N} \underline{h}^n} \equiv \frac{\sum_{n=1}^{L} \overline{h}^n \underline{z}^n + \sum_{n=L+1}^{N} \underline{z}^n \underline{h}^n}{\sum_{n=1}^{L} \overline{h}^n + \sum_{n=L+1}^{N} \underline{h}^n}$$

5. $$z_r = \underset{x \in [1, N-1]}{max} \frac{\sum_{n=1}^{x} \underline{h}^n \overline{z}^n + \sum_{n=x+1}^{N} \overline{z}^n \overline{h}^n}{\sum_{n=1}^{x} \underline{h}^n + \sum_{n=x+1}^{N} \overline{h}^n} \equiv \frac{\sum_{n=1}^{R} \underline{h}^n \overline{z}^n + \sum_{n=R+1}^{N} \overline{z}^n \overline{h}^n}{\sum_{n=1}^{R} \underline{h}^n + \sum_{n=R+1}^{N} \overline{h}^n}$$

given that $\{\underline{z}^n\}$ and $\overline{z}^n$ are first sorted in an ascending order respectively. Then points L and R are determined by

$$\underline{z}^L \leq z_l \leq \underline{z}^{L+1}$$

$$\overline{z}^{R} \leq z_r \leq \overline{z}^{R+1}$$

Wheras z_l and z_r are computed using the Karnik-Mendel (KM) algorithms (Mendel & John, 2002).

6. Finally the defuzzified output is computed as $z = \frac{[z_l + z_r]}{2}$

Although many ways exist for type-reduction and defuzzification in type-2 fuzzy sets but the KM method is the most extensively used approach.

The assessment of an architecture is based on the key performance attributes (KPA) selected by the stakeholders. Each KPA (KP_r) of SoS has a certain range of values within which it is considered meaningful. This range is derived from the stakeholder’s needs and interviews with all component systems owners. The KPA’s act as objective functions in the multi-objective meta-architecture generation problem. The KPA properties as shown in Figure 3.16. include:

1. Range of Values of KPA for evaluating SoS capability C can be provided with different levels of linguistic granularization as shown in the example above.
2. Depending on the problem the type of member ship function is required the represent the ambiguity in each KPA.
3. The crisp value of each KPA is hard to determine. Hence they are aggregated using the parts that account for each KPA. For example it is difficult to find an absolute value of net-centricity of a SoS. Since it can be viewed as a composition of interoperability and communication with ground control system, both these values are computed and aggregated using type-1 fuzzy inference.
4. Later all KPAs are aggregated using type-II inference since there is more inherent ambiguity amongst them that can be taken into account.
5. This way the crisp values are first fuzzified and fed into fuzzy inference system for type-1. This is later defuzzified to obtain values for each KPA. This is fain fuzzified using type-2 inference and later defuzzified to obtain SoS architecture quality. Based on the assessment scheme of the architecture a compromised solution is selected. The implementation of a meta-architecture through a negotiation process is explained in the next section 3.5.

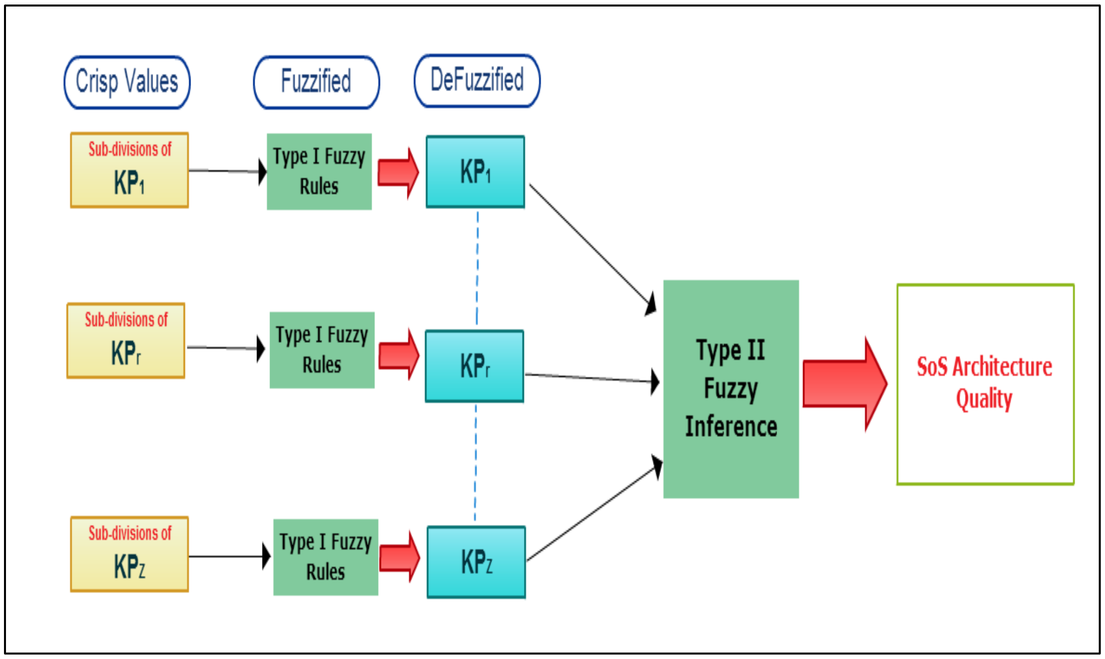


Figure 3.16. General structure of architecture assessment function

3.5. SOS NEGOTIATION APPROACH

The Acknowledged SoS manager negotiates with systems that are selected as part of the meta-architecture during the meta-architecture generation process. A negotiation procedure is necessary for the actualization or implementation of the meta-architecture generated. Since the SoS manager cannot force his demands on participating systems, negotiation helps in achieving an architecture that is implementable. The SoS manager negotiation mechanism consist of three phases of

i. modeling the opponent
ii. making a decision based on the previous offer
iii. Finally generating a counteroffer.

A bilateral counteroffer based negotiation mechanism is chosen between an SoS manager and an individual system under multiple attributes as depicted in Figure 3.17. The attributes or issues are assumed to be independent of each other and are bargained simultaneously. Modeling the opponent involves characterizing the opponent’s negotiation behavior; which could be considered cooperative, semi-cooperative or non-cooperative etc. A decision mechanism is needed to reject the offer for no further negotiation, or accept the offer as it is currently or negotiate for another round to bargain further. In case of further negotiation rounds a counter-offer generation mechanism is needed. Counter offers in automated negotiation are classified on the bases of constraints used to bargain such as time taken to negotiate, value of the overall utility achieved by a party over a set of issues, or constraints based on available resources. Section 3.5.1 gives an overview of the negotiation mechanism and variables used to explain the problem. Section 3.5.2 describes the strategy to model the opponent. Section 3.5.3 illustrates the strategy for making a decision on the negotiation offer of the opponent. Finally, in Section 3.5.4 several utility based concession curves are proposed for the SoS manager to make counteroffer. Figure 3.18. gives an overview of the three salient features of automated negotiation used in this work.

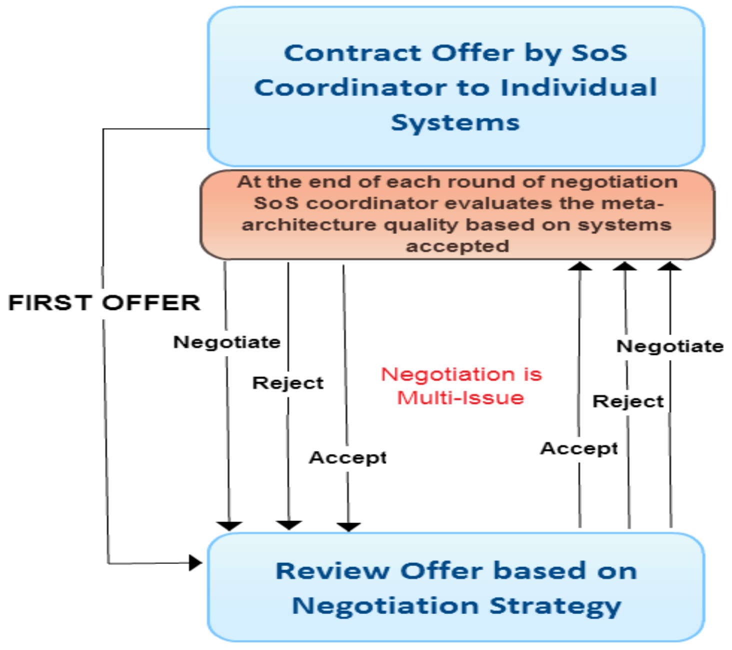

Contract Offer by SoS Coordinator to Individual Systems
At the end of each round of negotiation SoS coordinator evaluates the meta-architecture quality based on systems accepted
FIRST OFFER
Negotiate
Reject
Accept
Negotiation is Multi-Issue
Negotiate
Reject
Accept
Review Offer based on Negotiation Strategy

Figure 3.17. Bilateral negotiation mechanism

3.5.1. General Negotiation Protocol. In this section the variables used in the describing the protocol are listed for the user. The negotiation strategy is designed for a one to many participants and is not mediated by any coordinator. The structure consists of a SoS manager and multiple systems selected as part of the solution in the meta-architecture. Let us define:

V_p : *p= {1, 2,…, P}*: Attributes for bilateral negotiation

t_{max}: Total round of negotiations possible

$t = \{0,1, \dots t_{max}\}$

$V_p^{SoS}(t)$: The value of the attribute V_p for SoS manager at time t

$V_p^{S}(t)$: The value of the attribute V_p for system owner at time t

A number of negotiation rounds with different system types and SoS coordinator are conducted. Negotiation offers made by systems reveal incomplete information about their preference of issues and their strategy.

The following figure describes the methodology of modeling the opponents behavior through clustering, making a decision on the negotiation offer based on fuzzy 2-tuple linguistic multi-criteria decision making and finally generating a counteroffer based on utility concession curves.

The figure explains the processes involved in succession such as the hierarchical clustering followed by the k-means clustering. The labeled data obtained after clustering is then trained using a supervised learning algorithm. Two techniques such as learning vector quantization and radial basis function network were tried. Following which the trained network is able to predict the class of the incoming new offer. The SoS can make a final decision on the offer using linguistic fuzzy terms. This method is also known as the computing with words. Finally if the SoS feels that it needs to negotiate more it can use time dependent equations to make a counteroffer to the individual systems.

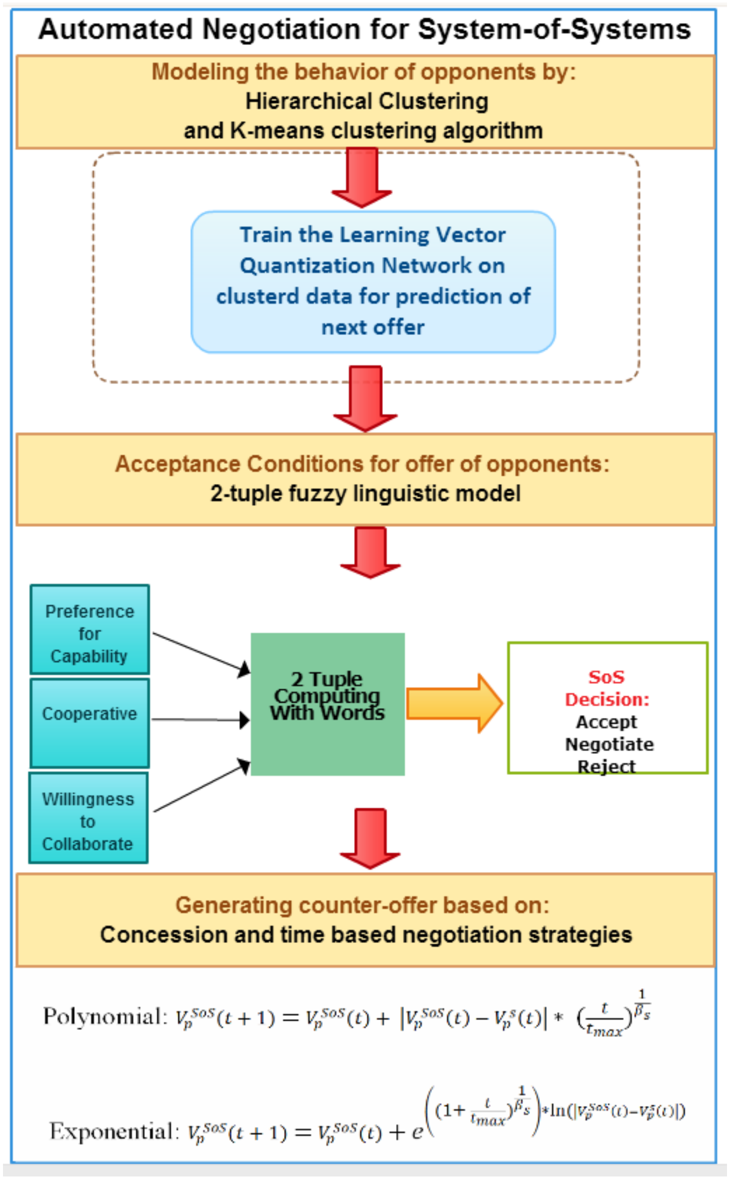

Automated Negotiation for System-of-Systems
Modeling the behavior of opponents by:
Hierarchical Clustering
and K-means clustering algorithm
Train the Learning Vector Quantization Network on clusterd data for prediction of next offer
Acceptance Conditions for offer of opponents:
2-tuple fuzzy linguistic model
Preference for Capability
Cooperative
Willingness to Collaborate
2 Tuple Computing With Words
SoS Decision:
Accept
Negotiate
Reject
Generating counter-offer based on:
Concession and time based negotiation strategies
Polynomial: $V_p^{SoS}(t+1) = V_p^{SoS}(t) + |V_p^{SoS}(t) - V_p^{s}(t)| * (\frac{t}{t_{max}})^{\frac{1}{\beta_s}}$
Exponential: $V_p^{SoS}(t+1) = V_p^{SoS}(t) + e^{\left((1+\frac{t}{t_{max}})^{\frac{1}{\beta_s}}\right)*\ln(|V_p^{SoS}(t)-V_p^{s}(t)|)}$

Figure 3.18. Three Salient Features of Automated Negotiation

3.5.2 Modeling the Opponent. Information regarding the opponent is extremely important to improve automated negotiation strategies for multi-issue multi-party negotiation (Hindriks, Jonker, & Tykhonov, 2009) and (Hindriks & Tykhonov, 2008). To have a better negotiation strategy each party requires information regarding the preferences of issues of the opponent. This information can be used to negotiate effectively. In other words it is imperative to model the behavior of the opponent by his previous offers or some other method. This helps in increasing the efficiency of agreements and is a superior method than concession-based negotiation strategy of (Baarslag, Hindriks, & Jonker, 2011).

In a concession-based negotiation size of next concession is mainly decided on the basis of the utility gap between the preceding bids of the opponent and the party. Another method of using fuzzy similarity to estimate the preference structure of the opponent and then uses a hill-climbing technique to explore the space of possible trade-offs has also been successfully implemented (Faratin, Sierra, & Jennings, 2002). In this work single round of negotiations are used to model the opponent's behavior. This is because it has been observed that usually opponent avoids any chance of revealing their preferences over issues to be exploited.

An SoS coordinator uses initial estimates of the problem's complexity and size, combined with the amount of resources currently possessed to propose a first offer to individual systems. These systems then respond to the first offer according to their negotiation behaviors. The SoS coordinator was not informed of a participating system's behavioral strategies and desires to adapt its negotiation policy accordingly. The following paragraphs outline the process of analyzing the negotiation data.

The SoS coordinator records both the offer and the counteroffer for each system. It calculates the amount of concession in each issue for each system. Concessions in all issues are calculated for each system i (see Table 2). After recording this data it is used for clustering which can reveal any behavioral groupings in counter-offers. For example, a cooperative system would agree to work for less money than a non-cooperating system would. Similarly, a non-cooperating system would ask for more money in lieu of time taken to prepare for participation. The clustering is done in multi-dimensional space of the number of negotiation attributes P. The following notation describes the clustering operation:

o_g: $g \in \boldsymbol{G}$, $\boldsymbol{G}= \{1, 2,\ldots, NoB\}$: - the number of observations made

P - the number of issues or attributes of negotiation present

L- the number of clusters the user either predicted or defined

C_h - the h^{th} cluster, a subset of $\mathrm{h} = \{1, 2, \ldots, L\}$

The values in the Table 3.1. form a P dimensional data that can be clustered to model the opponent behavior.

Table 3.1. Concession calculated by SoS manager for each system

System j	NA_1	NA_2	NA_p
	$\Delta_1 = NA_{1\text{SoS}} - NA_{1\text{S}}$	$\Delta_2 = NA_{2\text{SoS}} - NA_{2\text{S}}$	$\Delta_\text{p} = NA_{p\text{SoS}} - NA_{p\text{S}}$

3.5.2.1 Hierarchical clustering. Hierarchical clustering is a type of agglomerative clustering (Freeman, 1994). It builds a hierarchy of clusters such that clusters at one level are combined as clusters at the next level. It does not require the number of clusters in advance to proceed with. This process creates a cluster tree which is known as the *dendogram*. Hierarchical clustering algorithms require very little a priori knowledge of the data and are a non-parametric method of auto-classification (Johnson, 1967). Multi-level clustering assists the user in deciding at how many clusters are appropriate for his problem. It is often used as precursor to many other clustering algorithms to give an overview of how many clusters might be present in the data. The basic methodology of this clustering method is explained as follows:

1. Given *N* data points are to be clustered.
2. Assign a cluster based on each data appoint, which results in *N* clusters
3. A similarity metric (distance) is chosen to quantify the separation between the clusters. Similarity metric parameter defines how the distance between clusters is calculated. Some common options are:
    a. *Average Linkage*: The distance between any two clusters is estimated as the average of the distances between all the points in those clusters.
    b. *Complete Linkage*: The distance between any two clusters is the distance between the farthest points in those clusters.
    c. *Single Linkage:* The distance between two clusters is the shortest distance between any member of one cluster to anyone in the other cluster.
4. Calculate all pair-wise distances between clusters making a $N\ X\ N$ matrix
5. The most similar pair of clusters is merged into a single cluster and then all distances from this new cluster to all other clusters are evaluated to update the matrix. Each time two closest data points are merged until there is a single large

cluster containing all the original data points.

The *dendogram* helps in visualizing clustering of the data at different levels. To determine the best level for a given set of data is based on experience and type of problem being modeled. The Figure 3.19. shows three different levels represented by Line 1, Line 2 and Line 3 respectively. Each horizontal line cuts the *dendogram* at a number of places which is equal to the number of clusters present at that level. The y-axis is a measure of closeness of either individual data points or clusters. The data points are listed along the x-axis to see they belong to which cluster structure.

The decision maker can choose an appropriate level by looking at the dendrogram and hence arrive at the number of clusters that can be used as input for the clustering algorithms. Clustering through k-means is explained in the next section.

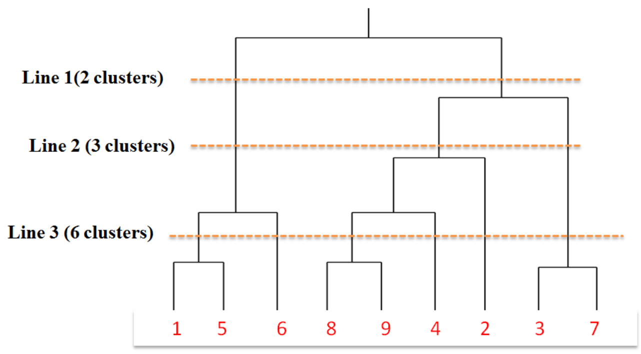


Figure 3.19. Three Salient Features of Automated Negotiation

Algorithms for hierarchical clustering are generally either agglomerative, in which one starts at the leaves and successively merges clusters together; or divisive, in which one starts at the root and recursively splits the clusters. Agglomerative algorithms begin with each element as a separate cluster and merge them into successively larger clusters. Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters. It depends on the problem to use either an Agglomerative or Divisive approach.

In my work since we are trying to model the behavior of the opponent which is unknown, we expect to have many behaviors. The numbers of behaviors will increase with the number of issues involved in negotiation. It would make more sense to start with assuming each data point (or offer) obtained from the systems is a different behavior. Therefore we used the Agglomerative approach. Although even if we start with the divisive approach I think we should arrive at the same number of optimal clusters.

3.5.2.2 K-means clustering algorithm. K-means clustering is one of the many unsupervised learning techniques (Grira, Crucianu, & Boujemaa, 2004) currently used to mine the underlying features of a dataset. Some of the popular techniques include partition around mediods (Kaufman, & Rousseeuw, 1990) where the major difference between k-means is that the algorithm uses mediods instead of centroids and the cluster centers may or may not be necessarily one of the data points, Fuzzy c-means (Pal & Bezdek, 1995) is based on k-means and on the concept that each data point has degree of being a member of a particular cluster, Expectation-Maximization (EM) algorithm (Moon, 1996), and Grid-Based Methods (Ilango & Mohan, 2010).

K-Means is useful in the cases where the user can gauge the count of clusters actually present. It is also computationally very less expensive as compared to other algorithms. K-means attempts to divide the data set into a predefined number of clusters such that the total distance between the members of each cluster and its respective centroid is minimized. Let us explain the major tenets of the algorithm.
Suppose there are N sample feature vectors o_1, o_2, ..., o_N and it is known they can be divided in L clusters where $L < N$. Let m_k be the mean of the vectors in cluster k. This suggests the following procedure for finding the k means:

- Make initial guesses for the means $\boldsymbol{c_1}, \boldsymbol{c_2}, \ldots, \boldsymbol{c_L}$
- Until the means do not change

    1. Use the estimated means to classify the samples into clusters by allocating each data point to the group that has the closest mean.
    2. For *i* from *1* to *L*
        - Replace $\boldsymbol{c_i}$ with the mean of all of the samples for cluster *i*

3. End for

- End until

The similarity metric often chosen for k-means is the distance measure $\left\|x_a^{(j)} - c_j\right\|$between a data point $x_a^{(j)}$ and the cluster center c_j.

K-means minimizes the sum of distances from each object to its cluster centroid, over all clusters which is represented as a cost function J.

$$J = \sum_{j=1}^{L} \sum_{a=1}^{n} \left\|x_a^{(j)} - c_j\right\|^2$$

J is the sum of all distances of n data points from their corresponding clusters. Hierarchical clustering alone might not be enough to determine the number of clusters required to give as input to the clustering algorithms. A number of inputs are used as clusters for k-means. Then as a method of validation a naive procedure called the elbow method is used to finally verify the approach.

3.5.2.3 Elbow method. To further confirm the enquiry on number of possible clusters present in the data a popular method known as the 'elbow method' "(Ketchen & Shook, 1996) is used based on results of k-means. The sum of squared error (SSE) is computed for some possible values of number of clusters values of ***k*** (for example 2, 3, 4, 5etc.). The SSE is calculated as the sum of the squared distance between each member of the cluster and its centroid.

$$SSE = \sum_{j=1}^{L} \sum_{a=1}^{n} \left\|x_a^{(j)} - m_j\right\|^2$$

SSE aims to represent the global error in clustering. With SSE on the vertical axis and the number of clusters on x-axis are plotted to help visualize the drop in SSE with change in number of clusters. Although with increasing number of clusters the SSE begins to drop yet there usually exists a point where there is not much change in SSE as clusters increase (Salvador & Chan, 2004). This point looks like an elbow and the number of clusters at that point is usually the best choice.

However, it may be said that the best value for number of clusters is a combination of human judgment and algorithm outputs. After clustering the data is now labeled with a mapping of inputs or the data points and the target or classes of the cluster. This labeled data can further be used for training a supervised learning algorithm for future prediction. Thus the behavior of new systems initially not present in the negotiation can be ascertained based on the through an incoming offer. The next section highlights two supervised learning algorithms called the learning vector quantization and radial basis function networks for training and prediction of classified data.

3.5.2.4 Training a LVQ network. Idea is to create an efficient mapping using supervised learning techniques between the data points and their centers. This will help us center of predicting the next incoming sample. Usually, supervised learning has inputs and preferred outputs provided by the user. Ensuing a period of training, the algorithm is capable of generalizing from the provided set of data to new sets of data. One such technique is called the learning vector quantization (LVQ) (Sato & Yamada, 1996). It is a precursor to self-organizing maps (SOM) (Somervuo & Kohonen, 1999) and supervised version of vector quantization (VQ). Sato & Yamada (1998) also provide an analysis of convergence in generalized LVQ. In other words LVQ is a neural net that combines competitive and supervised learning. It is useful for training classified data and prediction. It does not have any topological structure unlike its unsupervised counterpart SOM.

The LVQ Algorithm starts with a training set consisting of a training vector $\boldsymbol{x} = \{x_1, x_2, x_3, .. x_n\}$ and target output pairs are assumed to be given. The inputs form the input layer of the LVQ network. The numbers of neurons in the network are same as the number of classes present in the data. Let there be J classes present in the data where $k = \{1,2, .., J\}$. So there are J neurons in the output layer. All input vectors are connected to all the neurons in the network as shown in the Figure 3.20. The weights are also called the codebook vectors. The weight vector joining the inputs to the neuron k can be expressed as $\boldsymbol{w_k} = \{w_{1k}, x_{2k}, x_{3k}, .. x_{nk}\}$. Basically the codebook vectors act as piece wise linear functions to classify the data.

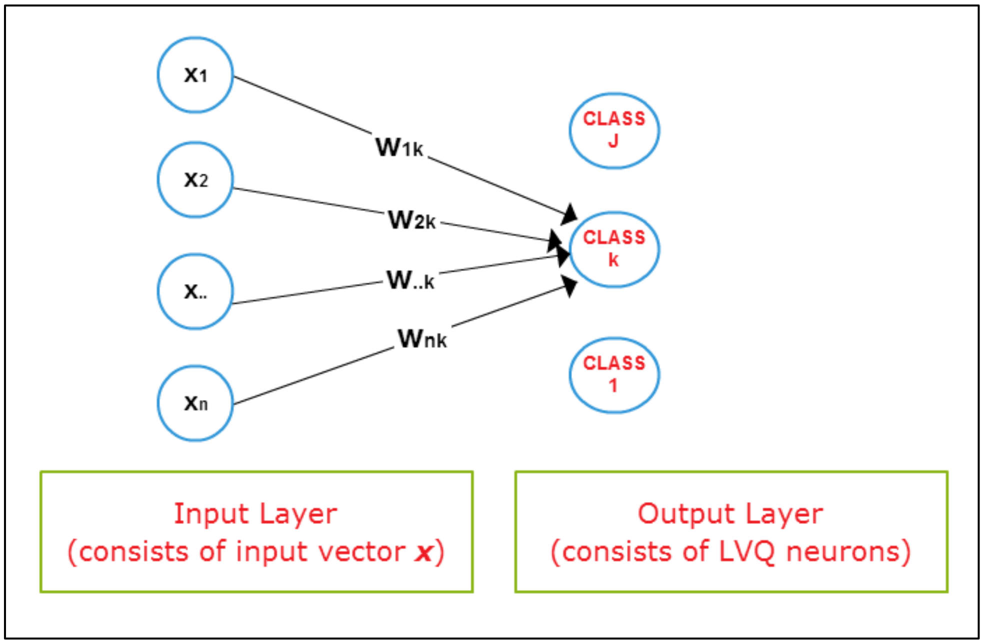

X1
X2
X..
Xn
W1k
W2k
W..k
Wnk
CLASS J
CLASS k
CLASS 1
Input Layer (consists of input vector x)
Output Layer (consists of LVQ neurons)

Figure 3.20. Structure of learning vector quantization network

The training process can be explained using the following rules:

Rule 1:

Initialize first J inputs as J weight vectors, given J classes are present in the data. Other techniques include randomly selecting J inputs from the data for initializing weights.

Rule 2:

```
While termination criterion ≠ true
    For each input vector
        Calculate the distance metric $D(k)$ from the all the weight vectors.
```

$$D(k) = \sum_{i=1}^{n} \|\mathrm{x}_i - w_{ik}\|^2$$

Choose the k that makes $D(k)$ minimum since that is to minimized. Check whether k or predicted class of the input vector is same as the target class. If the input $\boldsymbol{x}$ and the associated weight vector $\boldsymbol{w_k}$ have the identical class tag, then update the weight vector by the attraction rule (bring it closer to the input)

$$\boldsymbol{w_k}(new) = \boldsymbol{w_k}(\boldsymbol{old}) + \boldsymbol{\eta}\,(\boldsymbol{x} - \boldsymbol{w_k}(\boldsymbol{old}))$$

If the input $\boldsymbol{x}$ and the associated weight vector $\boldsymbol{w_k}$ have different class tags, then move them apart by repulsion rule:

$$\boldsymbol{w_k}(new) = \boldsymbol{w_k}(\boldsymbol{old}) - \boldsymbol{\eta}\,(\boldsymbol{x} - \boldsymbol{w_k}(\boldsymbol{old}))$$

End For

Reduce the learning rate parameter

End While

Termination of training may depend upon a fixed number of iterations or setting the minimum threshold of the learning rate.

3.5.2.5 Radial basis function network. The clustered data then can be viewed as a mapping of inputs to target (classes) and is used to train a radial basis function network (RBFN) (Buhmann, 2003) for prediction. The RBF network is a substitute to multilayer perceptron (MLP). The two differences between a MLP and a RBFN are that a RBFN trains a single layer of weights unlike MLP where all layers are trained. Also the usual activation function used in RBFN is a Gaussian as a replacement for a sigmoid.

The training phase can be done using gradient descend of the error loss function, so it is relatively simple to implement. The RBFN is three layered feed-forward neural network. The first layer is called the input layer, the second layer is called the hidden layer, and finally the last later is called the output later.

Different kernel functions such as Gaussian, polynomial, and exponential can be used for hidden layer transfer functions. The network training is divided into two stages: first determine the weights from the input to hidden layer and subsequently calculate the weights from the hidden to output layer (Schwenker, Kestler, & Palm, 2001). Weights between the hidden layer and the output layer are adapted during training.

The cluster centers become the centers of the RBF units. The number of clusters L, is a design parameter and also determines the number of nodes in the hidden layer. The centers (acquired by k-means algorithm) are used to compute the centers and widths for each basis function in the hidden neurons (Haykin, 2009). RBFN centers for hidden nodes activation functions is same a k-means centers. Now the width of each RBF unit can be calculated using the K-nearest neighbor's algorithm. A suitable number k is chosen, and the root-mean squared distance between the current cluster center and its k nearest neighbors is calculated, and this is the value chosen for kernel width σ .The formula used to fix kernel width is

$$\sigma = \sqrt{\frac{\sum_{i=1}^{k}(m_j - m_i)^2}{k}}$$

Other methods include choosing the width as a ratio of d_{max} the maximum distance between the chosen centers, and m the number of centers (Deshmukh & Gholap, 2012). A training algorithm developed (Chen, Hong, Luk, & Harris, 2009) uses evolutionary algorithms to construct tunable radial basis function networks and decide the optimal center points along with the width for each kernel in the hidden neuron.

3.5.3. Making a Decision Based on Current Round of Negotiation. It is important to make a decision on an offer in every negotiation. To decide this point of break-off (Baarslag Hindriks, & Jonker, 2014) the SoS manager must decide the conditions under which an offer will be accepted or rejected. The decision has to be made to overcome the dilemma of making a sub-optimal offer. Some of the most effective strategies applied in literature are Bayesian learning agent (Hindriks & Tykhonov, 2008), agent architecture for multi-attribute negotiation (Jonker, Robu, & Treur, 2007). Some of the methods in the past that employ fuzzy logic for making acceptance decisions in negotiations include a fuzzy e-negotiation agents system (FeNAs) (Kowalczyk & Bui, 2000), and fuzzy logic based intelligent negotiation agent (fina) in eCommerce (Wang, Shen, & Georganas, 2006).

The decision on a particular offer is based on the cooperativeness of the systems measured (Baarslag, Hindriks, & Jonker, 2013), system's willingness to collaborate, and the SoS's preference for acquiring that capability. After identifying the class of behavior the SoS coordinator can use a fuzzy inference engine to decide whether he wishes to accept the systems offer, reject the offer or further negotiate (See Figure 10). The model used is Multi criteria decision making (MCDM) Dodgson, Spackman, Pearman, & Phillips, 2009) with 2-tuple fuzzy linguistics (Carlsson & Fullér, 1996).

The decision to accept, reject or negotiate further with a system is based on the cooperative behavior of the system, willingness to collaborate, and the SoS's preference for acquiring that capability. After identifying the class of behavior the SoS coordinator can use a fuzzy inference engine to decide whether he wishes to accept the systems offer, reject the offer or further negotiate. Since all the three parameters are difficult to compute numerically the SoS coordinator has fuzzy linguistic model to aid in decision making.

The problem is handled as multi-criteria decision making using 2-tuple fuzzy linguistic model. The fuzzy linguistic approach represents qualitative variables as linguistic values by use of linguistic variables (Herrera & Martínez, 2000). The 2-tuple fuzzy linguistic representation model represents the linguistic information by means of a 2-tuple (s, α) where s is a linguistic label and α is a numerical value that represents the value of the symbolic translation.

If a variable can take words in natural languages as its values, it is called a linguistic variable, where the words are characterized by fuzzy sets defined in the universe of discourse in which the variables are defined. The linguistic variable is represented by a set of membership functions.

Definition 1. Let β be the result of aggregation of the indexes of a set of labels assessed in a linguistic term set S. Then, $\beta \in [0, g]$, where $g + 1$ is the cardinality of the set S.

Let $i = round(\beta)$ and $\alpha = \beta - i$ are two values such that, $i \in [0, g]$, and $\alpha \in [-0.5, 0.5]$, α is then called the symbolic translation.

Definition 2. The aggregation of the indexes β can be converted to s_i the closest index label to β and α the symbolic translation.

$$\Delta(\beta) = (s_i, \alpha) \qquad (3.17)$$

3.5.3.1 Ordered weighted averaging operator. Ronald R. Yager (1997) introduced an aggregation technique called the Ordered Weighted Averaging (OWA) operators, which are capable of modeling a wide range of aggregation preferences. A modified version of OWA called the Linguistic Ordered Weighted Averaging Operator (LOWA) is used here. Let S be a set of 2-tuples, $\{(\boldsymbol{s_1}, \boldsymbol{\alpha_1}), \ldots, (\boldsymbol{s_1}, \boldsymbol{\alpha_n})\}$ and $\boldsymbol{W} = (\boldsymbol{w_1}, \ldots, \boldsymbol{w_n})$ be an associated ordered weighting vector that satisfies $\boldsymbol{w_i} \in [\boldsymbol{0}, \boldsymbol{1}]$ and $\sum \boldsymbol{w_i} = \boldsymbol{1}$. Then LOWA for such a set S can be defined as:

$$\Delta\left(\sum_{i=1}^{n} w_i \cdot \beta_i\right)$$

The values in the vector $\boldsymbol{\beta}$ are first ordered such that $\beta_1 \leq \cdots \leq \beta_i \ldots \leq \beta_n$ such that w_1 is always linked to the lowest value in the vector $\boldsymbol{\beta}$ and the w_n is always linked to the highest. β for a linguistic 2 tuple set is previously defined as $\Delta(\beta) = (s_i, \alpha)$. The process of arriving at the rank of alternatives is done using two processes, aggregation and comparison. A wide range of 2-tuple aggregation operators have been developed such as the weighted average operator, the ordered weighted average (OWA) operator (Wei, 2010). After aggregation a new value of β is obtained. This value of β is converted to its 2-tuple representation as explained in the example below.

Example: Let us suppose a symbolic aggregation operation over labels assessed in S= $\{s_0, s_1, s_2, s_3\}$ is such that $\beta_1 = 2.1$. The 2-tuple representation of value is:

$i = round(2.1) = 2;\ \alpha = \beta - i = 0.1;\ \Delta(2.1) = (s_2, 0.1)$

For comparing or ranking the alternatives is done using the 2-Tuple Comparison Operators. The comparison of linguistic information represented by 2-tuples is carried out according to an ordinary lexicographic order. Let (s_x, α_1) and (s_y, α_1)be two 2-tuples, then they are compared using the following rules:

• if $x < y$ then (s_x, α_1) is smaller than (s_y, α_1)

• if $x = y$ then

1. if $\alpha_1 = \alpha_2$ then , (s_x, α_1) are (s_y, α_1) same
2. if $\alpha_1 < \alpha_2$ then , (s_x, α_1) is smaller than (s_y, α_1)
3. if $\alpha_1 > \alpha_2$ then , (s_x, α_1) is greater than (s_y, α_1)

For example a set S composed of four terms could be where S= $\{s_0 = VL, s_1 = L, s_2 = M, s_3 = H\}$shown in Figure 3.21. The first step is to assign a 2-tuple value for each alternative based on each attribute by the SoS manager as shown in Table 3.2. Subsequently calculate an aggregated value for each alternative over all attributes using 2-tuple Linguistic Aggregation. Finally all the alternatives are ranked based on this output. Some definitions and concepts are presented below to clarify the approach.

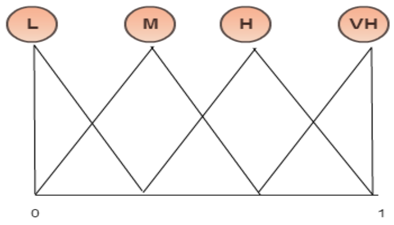


Figure 3.21. A set of four linguistic terms with their semantics

Table 3.2. General 2-tuple Linguistic Problem

Attributes/Alternatives	A1	A2
P1	(s_1, α_1)	(s_3, α_4)
P2	(s_2, α_2)	(s_1, α_1)
P3	(s_0, α_5)	(s_3, α_3)
2-tuple Linguistic Aggregation	$\beta_{A1} = (s_3, \alpha_{12})$	$\beta_{A2} = (s_2, \alpha_6)$

For the sake of ease we assume all α the symbolic translation as zero. Then alternative A1 has an aggregated value for all attributes (P1, P2, P3) as

$$\Delta\, \beta_{11} = (M, 0) => (i + \alpha_1) = 1, \beta_{11} = 1.$$

$$\Delta\, \beta_{12} = (H, 0) => (i + \alpha_2) = 2, \beta_{11} = 2.$$

$$\Delta\, \beta_{13} = (L, 0) => (i + \alpha_5) = 0, \beta_{11} = 0.$$

$$\beta_{A1} = \frac{1+2+0}{3} = 1; i = round(1) = 1;\ \alpha = 0; hence\ (s_{A1}, \alpha_{A1}) = (M, 0)$$

$$\Delta\,\beta_{21} = (VH, 0) => (i + \alpha_1) = 3, \beta_{11} = 3.$$

$$\Delta\,\beta_{11} = (M, 0) => (i + \alpha_1) = 1, \beta_{11} = 1.$$

$$\Delta\,\beta_{11} = (VH, 0) => (i + \alpha_1) = 3, \beta_{11} = 3.$$

$$\beta_{A2} = \frac{3+1+3}{3} = 3.33; i = round(3.33) = 3;\ \alpha = 0.33; hence\ (s_{A1}, \alpha_{A1}) = (VH, 0.33)$$

The aggregation is based on LOWA for a set of 2-tuples. Comparing or ranking the alternatives is done using the 2-Tuple Comparison Operators and alternative A2 is higher w.r.t to the rules given. The decision maker would choose alternative A2 over A1.

On the same note when this approach is applied to the SoS manager it can divide linguistic terms in classes for making a decision on choosing the alternatives. For example if the aggregated value of the alternative lies within the set of $\{s_0 = VL, s_1 = L\}$ the alternative is rejected. The SoS manager has a choice of making 3 kinds of decisions based on the aggregated linguistic terms of the alternatives namely: Decision of SoS{ Negotiate, Accept, or Reject}.

3.5.4. Proposing an offer. A counteroffer is made to move closer to an agreement in the multi-attribute offer space. It involves deciding the amount of concession to be made, taking into account effect of time elapsed so far and the behavior both the offer proposer and the opponent party. In all this makes quite a challenge to design offer generating strategy. An SoS coordinator can employ different time dependent and behavior dependent strategies to generate the next offer once he/she has arrived at a decision to negotiate further. An alternating protocol of offers and counteroffers is employed to reach a final decision agreeable to both parties. The convergence of a negotiation strategy (Yu, Ren, & Zhang, 2013) indicates that the negotiating agents are certain to come to an agreement if the space of available solutions within the problem is not an empty set. The following sections give an outline for three kinds of tactics based on resources, behavior and time (Matos, Sierra & Jennings, 1998).

3.5.4.1 Resource dependent tactics. Resource dependent tactics depend on the quantity of resource available (Faratin, Sierra, and Jennings, 1998). The tactic aims to become conciliatory with reduction in amount of resources. Resources could be time, number of systems interested in a particular negotiation or funding availability.

$$\boldsymbol{U(t) = \rho + (1 - \rho)e^{-resource(t)}}$$

where $\boldsymbol{resource(t)}$ is the resource available at time $\boldsymbol{t}$.

3.5.4.2 Behavior dependent tactics. Behavior dependent tactics are induced from the actions of the negotiation opponent (Axelrod, 1984). The tactics include Relative Tit-For-Tat (Relative-TFT) which accounts for in percentage the behavior exhibited by the opponent over a certain time period. On the contrary, Random Absolute Tit-For-Tat (Random-TFT) accounts for the behavior in absolute terms. These tactics work well under no time restrictions or deadlines.

3.5.4.3 Time dependent tactics. These tactics model the fact that the agent is likely to concede more rapidly as the negotiation deadline approaches. Two functions are generally employed for this purpose: the polynomial function and the exponential function (Faratin, Sierra, & Jennings, 1998). These functions represent an infinite number of possible tactics, one for each value of $\boldsymbol{\beta}$ (Coehoon and Jennings, 2004). The parameter $\boldsymbol{\beta}$ needs to be selected to ensure the convexity (or concavity) of the utility curve. The $\boldsymbol{\beta}$ however must be classified into one of the following three forms to change the behavior of the equations (Faratin, Sierra, and Jennings, 1998):

$\beta >> 1$: This choice is made if the opponent is Conceder (reluctant) (SoS starts losing ground fairly quickly) and function is concave

$\beta = 1$: This choice is made if the opponent is Linear (SoS concedes equal amount in each round of negotiation)

$0 < \beta < 1$: This choice is made if the opponent is Boulware (SoS concedes slowly till the deadline is nearly up) and function is convex

For the exact same value (big) of strategy parameter β the polynomial function is supposed to concede quicker at the start than the exponential one after which they behave similarly (Sierra, Faratin, & Jennings, 1999). β can be used in both the equations listed below to generate the new offer by the SoS coordinator. According to the assigned class of the systems offer the SoS coordinator can choose to have different values for the strategy parameter β. For non-cooperative systems the value of β is high and for a very cooperative system its value should be kept low. Faratin has suggested exponential functions besides with the polynomial function shown below in equations. The common characteristic among the two functions is that both exhibit convexity w.r.t. t, and their degree of convexity is determined through the parameter β.

$$\text{Polynomial: } V_p^{SoS}(t+1) = V_p^{SoS}(t) + \left|V_p^{SoS}(t) - V_p^{s}(t)\right| * \left(\frac{t}{t_{max}}\right)^{\frac{1}{\beta_S}}$$

$$\text{Exponential: } V_p^{SoS}(t+1) = V_p^{SoS}(t) + e^{\left((1+\frac{t}{t_{max}})^{\frac{1}{\beta_S}}\right) * \ln(\left|V_p^{SoS}(t) - V_p^{s}(t)\right|)}$$

Here $0 \leq \beta_S \leq 1$ is the system's strategy parameter and t is current round of negotiation s.t. $t > 1$, $V_i^{SoS}(t)$ is the SoS's offer to the system at current negotiation round t, $V_i^{s}(t)$ is the system's offer to the SoS at time t, $V_i^{SoS}(t+1)$ is the SoS's new offer to the system (using the equations) and t_{max} is the maximum number of negotiations possible (Bahrammirzaee, Chohra, & Madani, 2013).

It is expected that by the use of these equation based offer generations the SoS manger can respond to a system on each issue. Figure 3.22. gives examples of concession curves for the polynomial time-dependent family of tactics. The concession curves assume that the offers range between 5 units and 100 units. So the SoS coordinator can choose amongst a family of curves to cover the difference. The boulware curve occurs at $\beta = 5$, the linerar curve is at $\beta = 1$ and conceder curve corresponds to $\beta = 0.1$.

Nevertheless a negotiator might not just respond aggressively to an aggressive opponent or quickly conceding to as conceding opponent. There are can be number of behaviors theta are possible as shown in Figure 3.23. based on the negotiator's attitude (Baarslag, 2014).

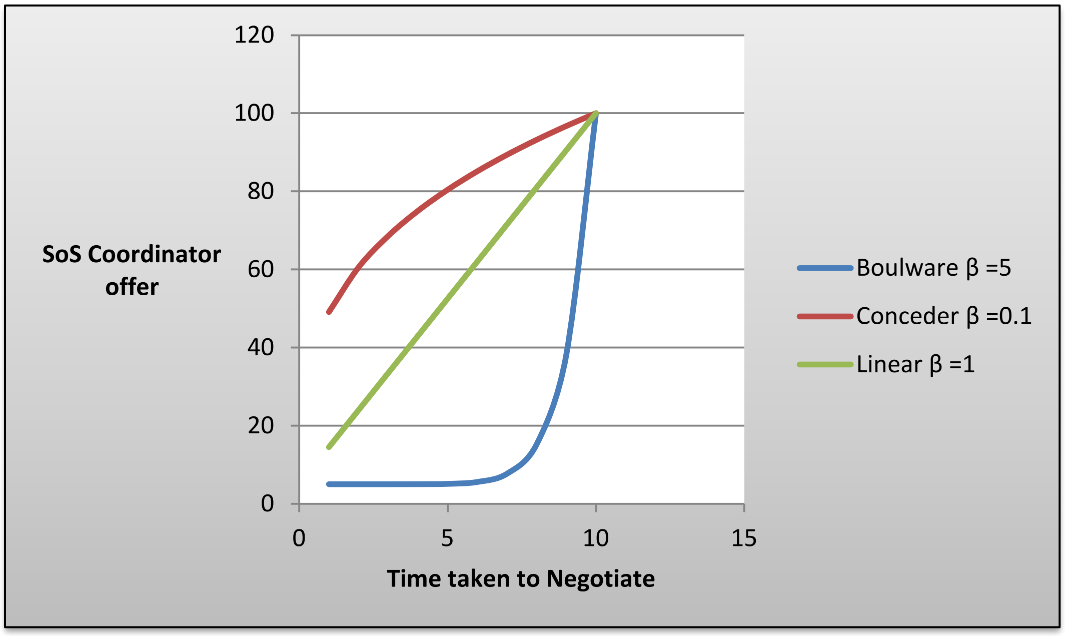


Figure 3.22. Concession Curves for the Polynomial Time-dependent Family

For example, the first tactic can be described as matching the exact style of negotiation of the opponent. Where a negotiator may cooperate (or conceding) when up against a cooperative opponent, on the other hand negotiator may behave competitively (not yielding easily) with a competing system (aggressive). This negotiator can be termed as a *matcher*.

The other contrary tactic is for a negotiator to behave in complete contrast to the opponent. In this tactic negotiator is cooperative towards a non-cooperative (competing) opponent. The negotiator also adopts non-yielding strategy (aggressive) to its cooperative opponents. Such a negotiator can also be called an *inverter*. In literature four types of behaviors are considered prominent, namely, Inverter, Conceder, Competitor, and Matcher.

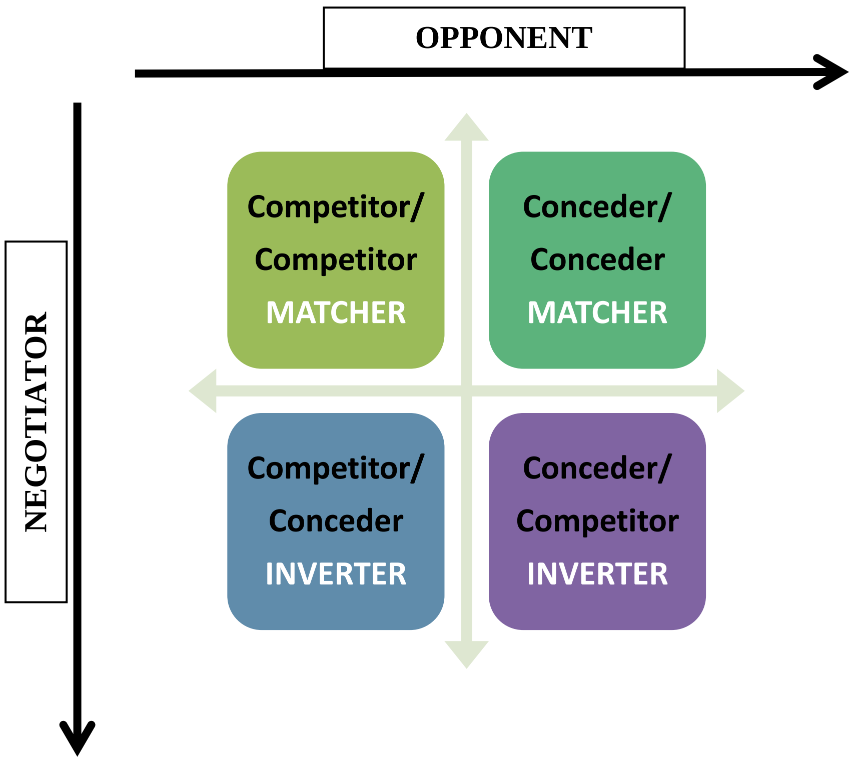

OPPONENT
NEGOTIATOR
Competitor/ Competitor MATCHER
Conceder/ Conceder MATCHER
Competitor/ Conceder INVERTER
Conceder/ Competitor INVERTER

Figure 3.23. Four styles of negotiation coordination

3.6. OVERALL NEGOTIATION PROTOCOL

The overall negotiation protocol can be illustrated as a set of statements as follows:

1) Send an offer to all systems simultaneously
2) Receive a counter-offer from all systems
3) Model the opponent behaviour-(clustering)
4) First make decision on set of systems with capability $i \text{ } where \text{ } i = 1 \text{ } to \text{ } M$
5) Need to select at least one system from each capability i
    a. Select a system with the best offer amongst them for the same capability if no system within a particular capability class is accepted
    b. Do so for each capability i to be acquired
    c. Form the architecture using the selecting systems and the interfaces

6) Evaluate the overall architecture quality based on the systems selected in one epoch

7) If the architecture is not of a predefined quality then go for a second epoch for systems not yet selected

The model described here is a decision making aid for the SoS manager. It does not so much find the best solution to designing a SoS, as help the manager explore the influence of the various constraints on the shape of a reasonable solution. The models described can be used in conjunction with others to explore the SoS context and goals.

This will help in developing SoS architectures including the full range of candidate systems and their interfaces. Our attempt has been to produce a holistic architecting methodology that is reconfigurable and has models that are adaptive to the environment.

The next sections describe the implementation of the model on search and rescue (SAR) SoS scenario.

4. COMPUTATIONAL INTELLIGENCE MODEL IMPLEMENTATION

To implement the architecture a Coast Guard Search and Rescue (SAR) problem serving the Alaskan coast region was selected. A brief introduction of SAR is given in section 4.1. For further details please refer to the integrated model structure report in volume 1 and 2 of FILA-SoS version 1.0.

In a SAR scenario a whenever a vessel in distress, the regulation of the sea requires mariners to reach out for help. This help comes in the form of a large number of disparate systems joining in an ad hoc congregation thus forming a SoS. The concept graphic or OV-1 is shown in Figure 4.1.

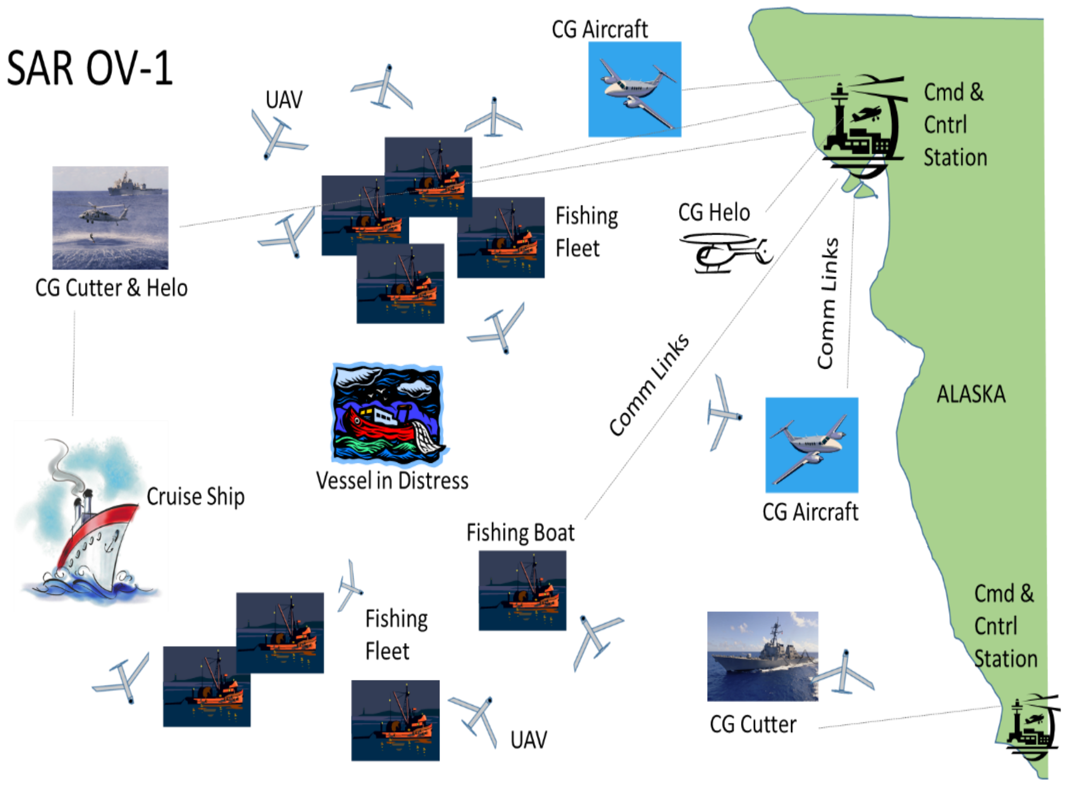


Figure 4.1. Operational View 1 for Search and Rescue scenario

4.1. PROBLEM STATEMENT

The overarching purpose of SoS in this case is a Coast Guard SAR capability within the Sea of Alaska was selected as the problem. The Coast Guard SoS has numerous systems with multiple capabilities such as cutters, aircraft, helicopters, communication systems, and control centers each different form the other and available from a number of stations in the area. In addition, the SoS comprises of systems such as fishing vessels, unmanned aerial vehicles (UAVs), civilian craft, and commercial to provide support in the event of a disaster strike (Breivik, Allen, Maisondieu, & Olagnon, 2013). The communication systems enable coordination of the sensing and rescuing capabilities of each vehicle.

The data and contextual information was collected from various Coast Guard documents and news stories about maritime rescues (Ullman, O'Donnell, Edwards, Fake, & Morschauser, 2003). A sample SAR SoS with 22 systems, with 5 capabilities is formed as shown in Table 4.1 and Figure 4.1.

This section explains the variables defined in context of the mission. Information required for architecture generation of a Search and Rescue (SAR) operation used to solve the Acknowledged SoS architectural evolution problem involves the overarching capability *C:* A Coast Guard SAR capability within the Sea of Alaska. The five sub-capabilities of the systems selected include c_j: $j \in \boldsymbol{J}, \boldsymbol{J}= \{1, 2, \ldots, 5\}$. For details refer to Table 4.1 for constituent system capabilities. The systems selected to participate in the SoS, s_i: $i \in \boldsymbol{I}, \boldsymbol{I}= \{1, 2, \ldots, 22\}$ can be referred to in Table 4.1. and Table 4.2. The information for variables such as performance of each system P_i, funding allocated to each system F_i, deadline for preparation D_i, interface cost IC_i, and operations cost OC_i can be referred from Table 4.2. There are five key performance attributes selected for the SoS such as KP_r : $r \in \boldsymbol{R}, R= \{1, 2, \ldots, 5\}$ which are listed as :

- $KP_1 = P^{SoS}$: Performance of SoS
- $KP_2 = A^{SoS}$: Affordability of SoS
- $KP_3 = R^{SoS}$: Robustness of SoS
- $KP_4 = M^{SoS}$: Modularity of SoS
- $KP_5 = NC^{SoS}$: Net-Centricity of SoS

Other related information required for SoS architecture generation is LV_i: the systems performance among participating systems based on ability to search and provide assistance and SP_i: the systems' speeds in air or water can also be inferred form Table 4.2. The three negotiation attributes $p= \{1, 2, 3\}$ for bilateral negotiation are: NA_1 =Funding, NA_2 =Deadline, and NA_3 =Performance.

The following sections describe the implementation of the meta-architecture generation, architecture assessment and implementation. The Figure 3.4. describes the other modules used in conjunction. The individual models presented in the next sections are part of the version 1.0 of FILA-SoS.

4.2. META-ARCHITECTURE GENERATION

This section describes the meta-architecture generation problem in terms of key performance attributes:

Optimize $\boldsymbol{F} = \{f_{KP_1}(\boldsymbol{s},\boldsymbol{IF}), f_{KP_2}(\boldsymbol{s},\boldsymbol{IF}), f_{KP_3}(\boldsymbol{s},\boldsymbol{IF}), f_{KP_4}(\boldsymbol{s},\boldsymbol{IF}), f_{KP_5}(\boldsymbol{s},\boldsymbol{IF})\ \}$

where $f_{KP_r}(\boldsymbol{s},\boldsymbol{IF})$ is the value of the key performance attribute r for decision variables $\boldsymbol{s}$ and $\boldsymbol{IF}$. A meta-architecture has to be selected from the systems in Table 4.1. The table gives the name of participating systems and the capabilities possessed by them.

Table 4.1. Types of the systems and capabilities present in the SoS

SysNo	Type	No of cap	No	Capability Name
s_1 and s_2	Cutter	2,5	c_2	High Speed
s_3 and s_4	Helicopter	2,5	c_2	High Speed
s_5 and s_6	Aircraft	2,5	c_2	High Speed
s_7 to s_{12}	UAV	1,5	c_1	IR & Night Vision
s_{13} to s_{16}	Ship or Vessel	3,5	c_3	Deliver Medical Aid
s_{17} and s_{18}	Coordination Control	4,5	c_4	RF Direction Finding
s_{19} and s_{22}	Communication	5	c_5	Communication Systems

Table 4.2. Input variables required for SAR meta-architecture generation

SysNo	Type	Capability	IC_i	OC_i	P_i	D_i	LV_i	SP_i
1	Cutter	2	0.03	0.2	12	1	8.3	6
2	Cutter	2	0.03	0.2	12	1	8.3	6
3	Helicopter	2	0.1	0.2	20	1	10.0	8
4	Helicopter	2	0.1	0.2	20	1	10.0	8
5	Aircraft	2	0.1	0.5	10	1	10.0	10
6	Aircraft	2	0.1	0.5	10	1	10.0	10
7	UAV	1	0.1	0.1	7	1	1.7	2
8	UAV	1	0.1	0.1	7	1	1.7	2
9	UAV	1	0.1	0.1	7	1	1.7	2
10	UAV	1	0.1	0.1	7	1	1.7	2
11	UAV	1	0.1	0.1	7	1	1.7	2
12	UAV	1	0.1	0.1	7	1	1.7	2
13	Fish Vessel	3	0.03	0.5	10	1	5.0	4
14	Fish Vessel	3	0.03	0.5	10	1	5.0	4
15	Fish Vessel	3	0.03	0.5	10	1	5.0	4
16	Civ Ship	3	0.05	2	8	1	6.7	4
17	Coord Ctr	4	0.05	0.5	5	1	0.5	0
18	Coord Ctr	4	0.05	0.5	5	1	0.5	0
19	Comm	5	0.02	0.03	1	0	0.5	0
20	Comm	5	0.02	0.03	1	0	0.5	0
21	Comm	5	0.02	0.03	1	0	0.5	0
22	Comm	5	0.02	0.03	1	0	0.5	0

In this work, two evolutionary algorithms were used, genetic algorithm and binary particle swarm optimization.

4.2.1. Genetic Algorithm. A genetic algorithm (GA) mimics the biological evolution process to solve constrained optimization problems. The GA is a good optimizer for large scale optimization problems with many decision variables. The basic idea is to improve the solution based on the objective function at each iteration through crossover and mutation of parent solutions. Over successive generations, the population converges toward a near optimal solution.

The important parameters of GA include crossover type (recombination operator $r\boldsymbol{\theta}_{\boldsymbol{r}}$), crossover rate, mutation type (mutation operator $m\boldsymbol{\theta}_{\boldsymbol{m}}$), mutation rate and method to choose (selection process $s\boldsymbol{\theta}_{\boldsymbol{s}}$) parents to crossover. Crossover rate defines how often will be crossover performed. In case of no crossover, offspring is an exact copy of its parents. The crossover type used in this work is two-point crossover. Crossover occurs between two individuals or chromosomes. Here two crossover points are selected on a binary chromosome.

Mutation rate defines the frequency of bits of chromosome to be mutated. Mutation type used here is bit flip. Crossover can be thought of as a global search parameter whereas mutation can be referred to as a local search parameter. The parents are selected using the elitism method. There are many selection procedures (Sivaraj & Ravichandran, 2011); the one used here is elitism. In elitism method, a predefined percentage (usually 50%) of new population is constructed using the best chromosomes ranked by fitness value. This prevents loss of any good solutions, which might have been lost due to crossover and mutation. Rest of the population is generated by mutating the parent chromosomes of the current population.

The pseudo code of the genetic algorithm as applied to the SAR is illustrated below:

Step 1: Generate a random population $P(0)$ of size μ individuals (chromosomes) with a chromosome size of $N + N * (N - 1)/2$ where $N = 29$

Step 2: Evaluate the fitness of each chromosome in the population through a fitness function $\boldsymbol{\phi}$ to calculate the fitness value.

Step 3: Create a new population $P(t)$ by iterating following steps until the new population is complete

Selection: Select two parent chromosomes from a population according to their fitness (selection process$\boldsymbol{\theta_s} = elitism$**)**

Crossover: With a crossover probability cross over the parents to form a new offspring $(P'(t) = r\boldsymbol{\theta_r}(P(t)))$

Mutation: With a mutation probability mutate new offspring at each position in chromosome $(P''(t) = m\boldsymbol{\theta_m}(P'(t))$

Accepting: Place new offspring in a new population $P''(t)$

Replace: Use new generated population for a further run of algorithm

Test: If the best fitness value does not change after certain iterations, stop, and return the best solution (highest fitness valued chromosome) in current population

$$P''(t) = \{\boldsymbol{\phi}(\overline{a}''_1(0)), \ldots., \boldsymbol{\phi}(\overline{a}''_\mu(0))\ \};$$

Loop Go to step 2

4.2.2. Binary Particle Swarm Optimization. In the original particle swarm optimization (PSO) the solutions are represented as a swarm of particles moving through the search space. A PSO algorithm preserves a swarm of individuals (called particles). Each individual (particle) represents an architecture solution. Particles try to follow the path of neighboring particles. Each particle is initialized with certain coordinates in problem space and a velocity. Each particle records its coordinates associated with the best solution (fitness) it has achieved so far. This value is called "pbest". Another "best" value that is tracked by a particle is the best value, obtained so far by any particle.

The parameters c_1 and c_2 are coefficients that regulate the relative velocity toward global and local best. Parameters r_1 and r_2 are two random numbers uniformly distributed in [0, 1]. The velocity vector V_{max}. is a bound on the velocities of particles on each dimension. In case of exceeding the velocity the particle is assigned the velocity of V_{max}. In a D-dimensional search space the position of i^{th} , where $i = \{1, .., N\}$,

particle of the swarm can be represented by a D-dimensional vector,

x_i = (x_{i1} ,…, x_{id} ,…, x_{iD}). Similarly the velocity can be expressed as

v_i = (v_{i1} ,…, v_{id} ,…, v_{iD}). The variables used in PSO are defined as follows:

c_1 : Self learning Factor

c_2 : Swarm learning Factor or social factor (It is suggested to maintain $c_1 + c_2 = 4$)

$r_{1,}\ r_2$: Random Number between 0 and 1used to maintain the diversity of the population

p_{id} :Personal Best Position of the i^{th} particle in d^{th}dimension

p_{gd} :Global Best Position of the g^{th} particle in d^{th}dimension

$v_{id}\ (t)$: The current velocity of the i^{th} particle in d^{th}dimension at time t

$v_{id}\ (t+1)$: The new velocity of the i^{th} particle in d^{th}dimension at time $t+1$

$x_{id}\ (t)$: The current position of the i^{th} particle in d^{th}dimension at time t

$x_{id}\ (t+1)$: The new or updated position of the i^{th} particle in d^{th}dimension at time $t+1$

w : Inertia Weight (The Inertia Weight determines the contribution rate of a particle's previous velocity to its velocity at the current time step)

V_{max}=-V_{min}=4

Particle position and velocity is updated using the equations given below:

$$v_{id}(t+1) = v_{id}(t) + c_1 r_1 (p_{id} - x_{id}(t)) + c_2 r_2 (p_{gd} - x_{id}(t))$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1)$$

The number of particles ranges from 20 - 100. Binary particle swarm optimization (BPSO) algorithm is a variant of the original PSO where the decision variables are binary in nature. For each particle, a binary value of 0 or 1 is allocated with a probability of 0.5 for all dimensions. The initial velocity of the particles in all dimensions is allocated using

$$v_{id} = V_{min} + (V_{max} - V_{min}) * \ rand()$$

A sigmoid function is used to scale the velocities between 0 and 1. These velocities are then used to update the position of the particles as a binary number. The concept is explained in the equations.

$sigmoid(v_{id}) = \frac{1}{1+e^{-v_{id}}}$;$x_{id} = 1 \ if \ U(0,1) < sigmoid(v_{id})$; $x_{id} =$
0 $otherwise$;

The pseudo code of the BPSO is illustrated below:

```
I) for each generation:
        Initialize particles in the population with a velocity and position
$Initialization \ \ P(0) = \{\overline{a}_1(0), \dots, \overline{a}_\mu(0)\}, \in \boldsymbol{I^\mu}$
II) For each particle in swarm:
    1.  Calculate fitness value ($evaluation \ operator \ \boldsymbol{\phi}$)
    2.  If the fitness of particle is better than the best fitness value (pBest) in its history,
        set the current value as the new pBest to $p_{id}$
    3.  Also set global best value as the new gBest to $p_{gd}$ for all particles or a
        neighborhood
    4.  Calculate particle velocity according to the velocity equation
    5.  Apply the velocity constriction
    6.  Update particle position according to the position equation
```

End While maximum iterations or minimum error criteria is not attained.

4.2.3. Fuzzy Evaluator. The fitness calculation is explained in this section. Figure 4.2. illustrates the modular fuzzy net process. It is used for to assessing the fitness of the of individual architecture instances (chromosomes). First, we calculate the values of inputs that are required for each KPA (e.g., affordability, performance, and net-centricity). Crisp values for the KPAs are then calculated using Type I fuzzy rules. These rules are based on the stakeholder's views. For example, a rule can be written that states the following: "If operations cost is high and the interfacing cost is high, then affordability is low". These fuzzy rules can be used to assign a crisp number to the affordability of the overall architecture. Each of the KPAs are then modeled as interval type II fuzzy sets (IT2FS) so that a crisp value can be obtained for the architectures overall quality.

IT2FSs have been shown to be more capable of modeling uncertainties than are T1 FSs. Each KPA with its inputs is referred to as a module. Type I FSs are used in modules to reduce computational time. The rules of the fuzzy evaluator are adjustable to allow for differences between the stakeholders' views. This adjustability makes fuzzy net usable for a larger section of perspectives that share the same domain problem. This approach can also be applied to model many other domains. The fuzzy network helps bring in uncertainties at lower levels of the KPA. KPAs of the SoS can be provided with different levels of linguistic granularization such as:

- Affordability: very costly, costly, cheap
- Modularity: little, average, good
- Performance: very low, mediocre, great
- Robustness: less, ordinary, excellent
- Net Centricity: low, medium, high

Triangular type-2 membership functions were used for all attributes. Twenty-five rules were created to link these five objectives to four fuzzy attributes. These statements help clarify stakeholders perspectives. Figure 4.3. represents the kiviat chart (visually displays a set of metrics) for the second rule to illustrate the concept. It explains architecture is *too risky* if it fails to meet more than 70% of key performance parameters.

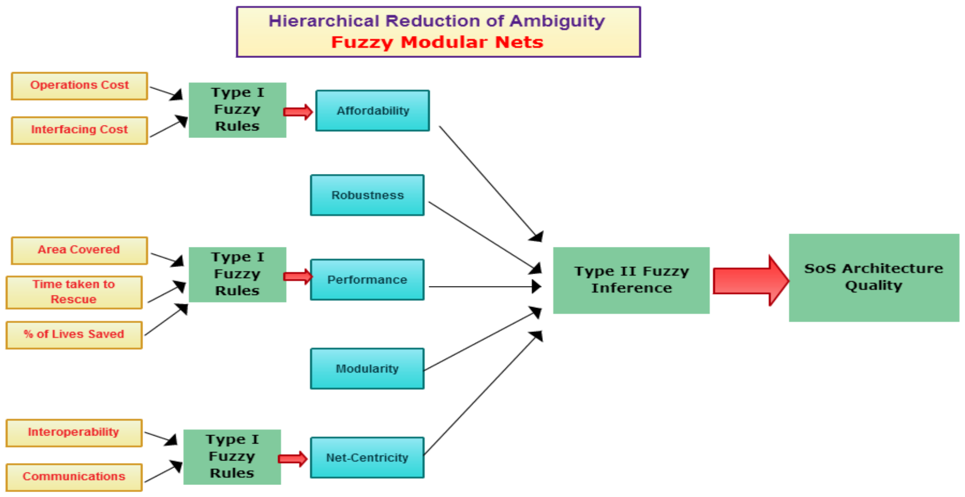


Figure 4.2. The fuzzy nets to evaluate architecture's quality

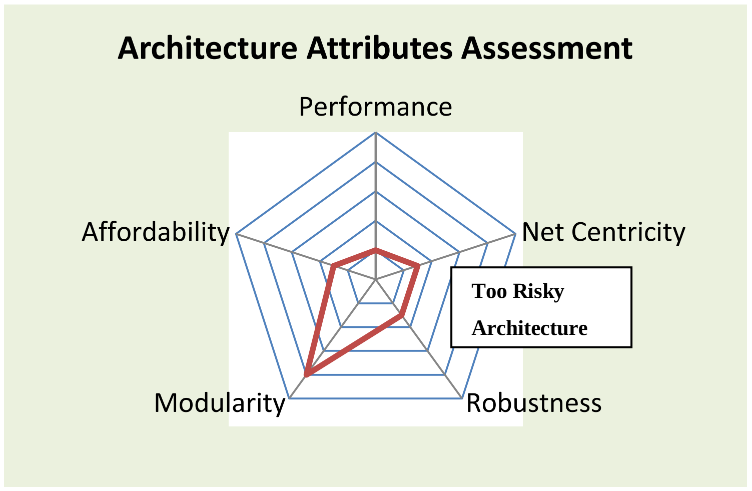


Figure 4.3. A Kiviat chart of Architecture Attributes for SoS Assessment

4.2.4. Fitness Evaluation of the Population. A fuzzy fitness evaluation is proposed, which uses a fuzzy estimator that is parallel in structure, simple in operations, and thus less in computation time to evaluate the fitness of individuals and reduces the computation time to solve a real problem. The fuzzy estimator can be adaptively trained to approximate the fitness function more accurately. The SoS manager may not want all systems to be present simultaneously when s/he is designing a mission. The correct set of systems should be chosen such that all ten capabilities are acquired, while trying to maximize overall performance and minimize the cost of acquisition.

The architectures are assessed according to a fitness function. The fitness function is a multi-objective problem. A number of independent functions need to be simultaneously addressed, to make up the fitness function. Five KPAs are the independent functions that will be used in this example for assessing the overall SoS architecture (Hilliard, Kurland, & Litvintchouk, 1997). The following sections explain each KPA in detail and how it will be calculated for each architecture.

4.2.4.1 Performance. The architecture's performance is calculated as fuzzy aggregate of $\boldsymbol{PA_i}$, $\boldsymbol{LV_i}$, and $\boldsymbol{SP_i}$. This method helps in obtaining a comprehensive view of the SoS performance in the areas that count in finding and rescuing people in distress. E.g., aircraft may be able to search more area faster, but cannot stop and render assistance; cutters are slower, but better at rendering assistance, and helicopters are good at both, but with a shorter range.

The membership functions used are Gaussian to aggregate the inputs for calculating the performance as depicted in Figure 4.4. The output is represented by triangular membership functions. The figure is generated through MATLAB. Nine rules were created to map the inputs to outputs. A rule used is written as: "If the Area covered is *less* and lives saved is *little*, and Rescue time is *small* then the Performance is *low*."

These rules are able to generate a non-linear surface when combined as shown in Figure 4.5. The tradeoffs between various inputs are captured through this surface. Using the Surface Viewer in MATLAB it presents with a three-dimensional curve. Only two inputs can be selected at a time whereas the third input remains constant. In the figure performance in terms of area covered is constant whereas rescue time and lives saved is variable.

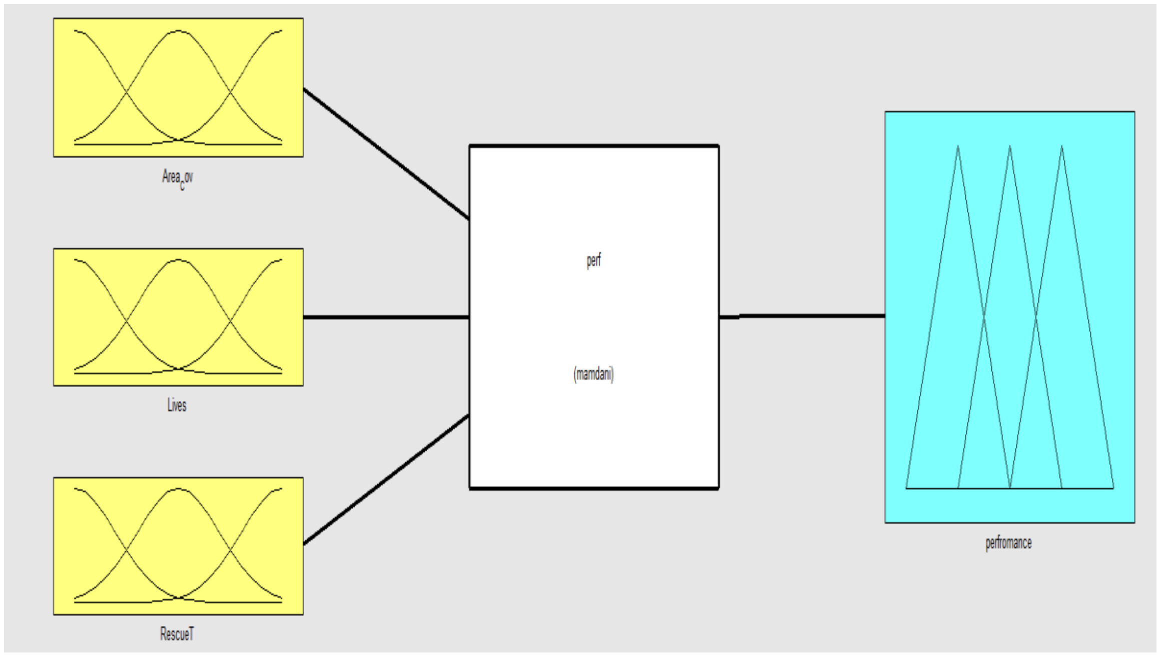
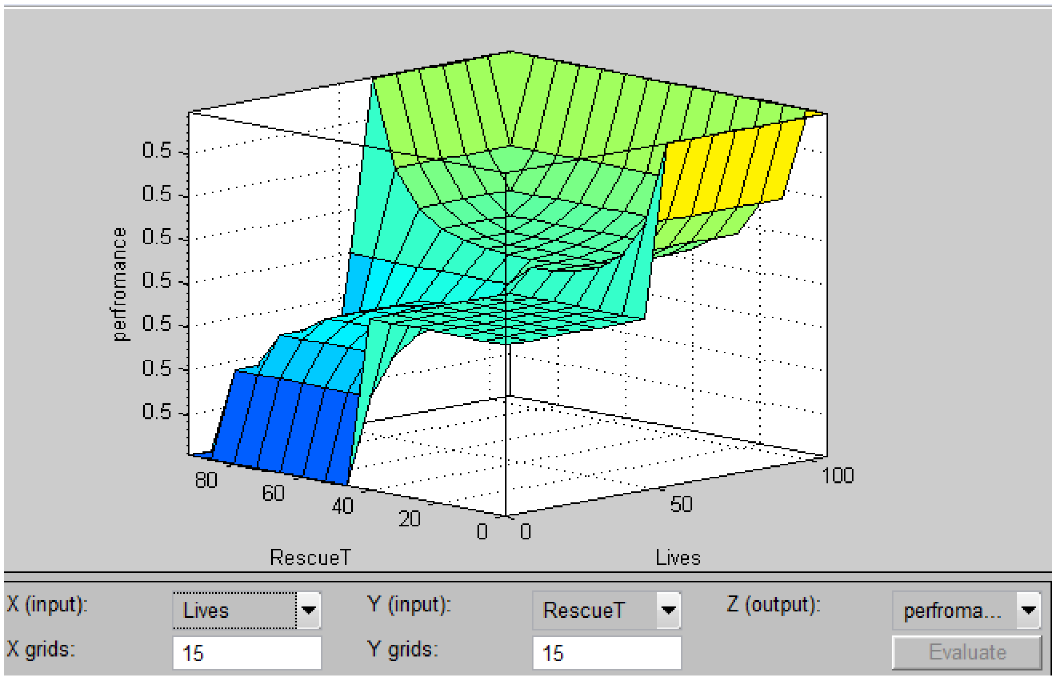


Figure 4.4. Fuzzy membership functions for Performance Attribute Assessment



X (input): Lives
Y (input): RescueT
Z (output): perfroma...
X grids: 15
Y grids: 15

Figure 4.5. Non-linear surface of tradeoffs between rescue time and lives saved

4.2.4.2 Net-centricity. Net-centricity is a property of SoS that relates to the availability ability to share of information; it is central to network-centric operations (Fry & DeLaurentis, 2011). The degree of net-centricity is a measure of the influence of net-centricity toward achieving the SoS objectives. The net-centricity of an architecture is based on interoperability of participating systems and centralized common communication for sharing information. Interoperability is defined as sharing an interface with other constituent component systems. Communication measures whether or not these systems are coordinating among themselves through a common control station or communication channel. In this problem, communications systems channels are numbered systems from 19 to 22.

$$\text{Interoperability} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \sum_{\boldsymbol{k=1}}^{\boldsymbol{N}} s_i * s_k * IF_{ik} \tag{4.1}$$

$$\text{Communication} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \sum_{\boldsymbol{k=26}}^{\boldsymbol{29}} \emptyset * IF_{ik} * IF_{kk'} \tag{4.2}$$

IF_{ik} and $IF_{kk'}$ aid in determining whether or not an interface exists between either systems i and k or systems k' and k, respectively. The metric used tries to capture the number of channels present that can transfer information within the SoS. If either of the two systems is not present, the metric is zero. If an interface exists between the two systems the net-centricity of SoS increases. Net-centricity increases further if the any two systems communicate through systems numbered 26 to 29. This is shown in Equation 4.2 where by multiplying ($1 \leq \emptyset \leq \infty$) enhances the communication capability.

Finally, if both systems are present but neither of them interface either among themselves or through the communication systems, the net-centricity is zero. The concept of interoperability presented here is simplistic. Interoperability can be viewed as having multiple dimensions from sharing and interface, to sharing data in the same format, to operational compatibility, exchanging useful information, systems' having trained together, and so on. Figure 4.6. explains a method to calculate net-centricity form the two metrics communication and interoperability.

After calculating interoperability and communication individually, they are fuzzified using Gaussian membership functions. Later, triangular membership functions are used to calculate the overall net-centricity. A rule used is written below to explain the idea "If the communication is more and interoperability is high, then the net-centricity is excellent." Figure 4.7. depicts the non-linear surface created by such rules.

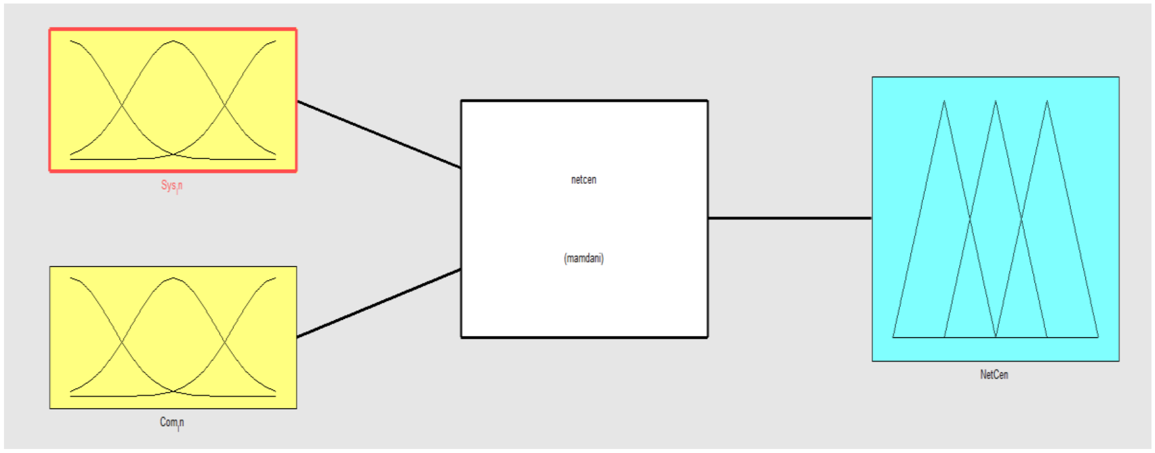
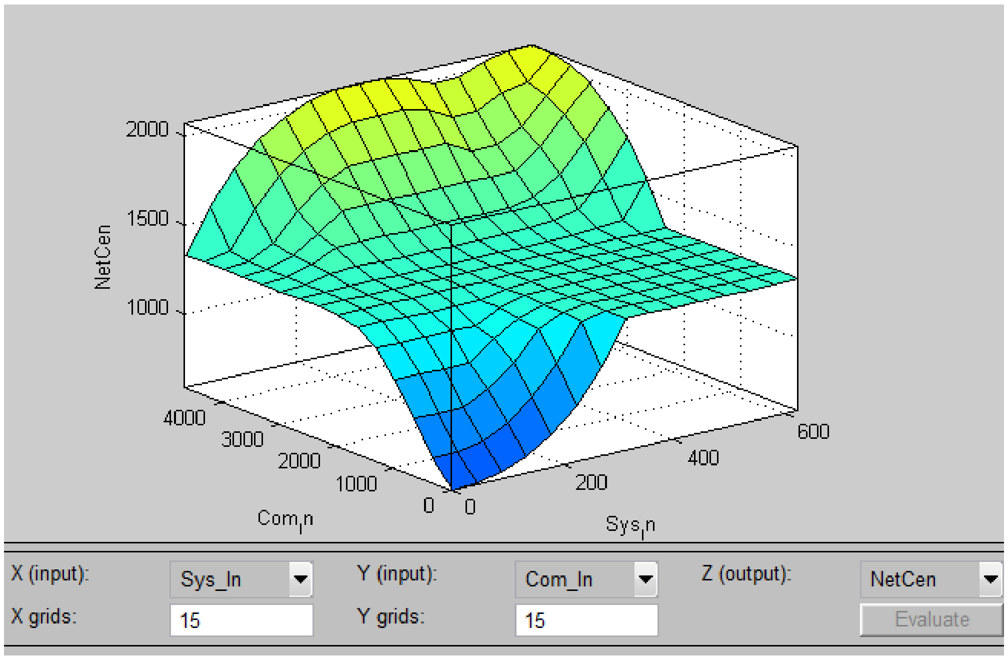


Figure 4.6. Fuzzy membership functions for net-centricity Attribute Assessment



X (input): Sys_In
X grids: 15
Y (input): Com_In
Y grids: 15
Z (output): NetCen

Figure 4.7. Non-linear tradeoffs surface between interoperability & communication

4.2.4.3 Affordability. Affordability is dependent on the sum of operation costs of the systems present times the number of capabilities possessed by that system. In addition, the interface development cost is systems present times the number of interfaces that specific system makes with other systems also present (Figure 4.8.).

Operations cost= $\sum_{i=1}^{N} OC_i * s_i * \sum_{j=1}^{M} a_{ij}$ *(4.3)*

Interfaces cost= $\sum_{s=1}^{N} IC_i * s_i * \sum_{i=1,k\neq i}^{N} IF_{ik}$ *(4.4)*

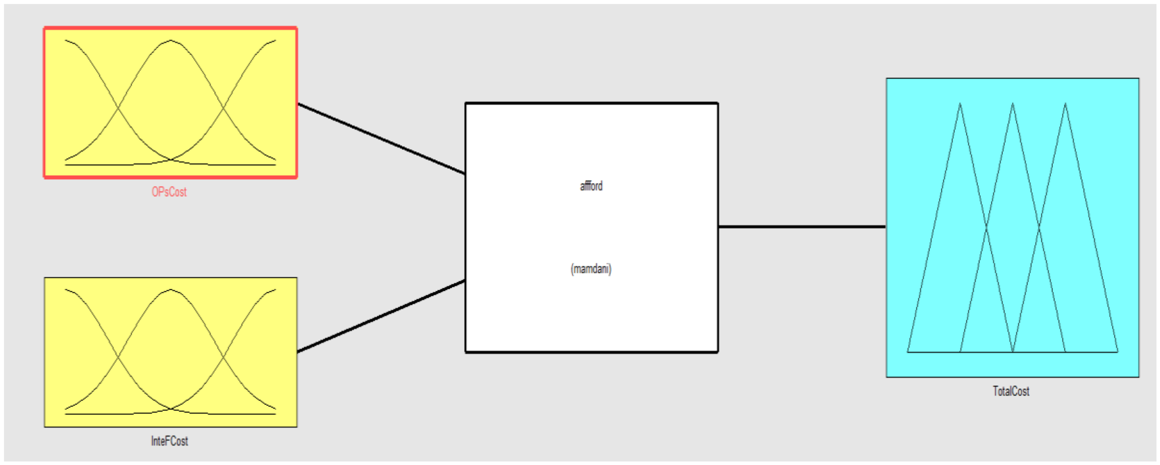


Figure 4.8. Fuzzy membership functions for Affordability Attribute Assessment

The operations cost and the interfaces cost are the two inputs and the total cost is the output. Inputs are represented by Gaussian membership functions whereas the output is a triangular membership function. A rule used is written as: "If the operation cost is *cheap* and interface cost is *more*, then the Affordability is *good*."

4.2.4.4 Robustness. One of the matrices within spectral measures of a graph is known as Laplacian (an SoS can be described as a graph that has vertices as systems and interfaces as edges.) The Laplacian (L) is calculated as the difference between the degree matrix (denoted by Δ) and the adjacency matrix (denoted by A). The second smallest eigenvalue λ_2 of the Laplacian is known as algebraic connectivity (Jamakovic & Uhlig, 2007). This value is used to assess the robustness of the graphs structure to external perturbations. The algebraic connectivity is equal to zero if and only if the graph is unconnected. (Fiedler, 1973) proved that the range of the value of λ_2 is $\mathbf{0 \leq \lambda_2 \leq \frac{N}{N-1} D_{min}}$, where N is the number of vertices and $\boldsymbol{D_{min}}$ is the minimum degree of the graph. A MATLAB toolbox was used to calculate the metrics (Bounova & de Weck, 2012).

4.2.4.5 Modularity. Modularity measures the structure of networks and graphs. It is used to compute the maximum possible indivisible graphs (either groups, clusters or communities) within a network. Here, Q (modularity metric) = the number of edges within groups subtracted from expected number of edges within group for a random graph with same node degree distribution as the given network. The Newman Girvan algorithm (Newman, 2006) is used to calculate it. The value of modularity is between '-1' and '1'. The networks modularity increases as this value increases.

4.3. META-ARCHITECTURE RESULTS

This study generated two models: a binary genetic algorithm (GA) that was combined with a fuzzy modular net fitness evaluator (Huang & Xie, 1998) and a binary particle swarm optimization (BPSO) (Kennedy & Eberhart, 1997) that was combined with the same fitness evaluator. These models were compared to one another in an attempt to generate better architectures. A fuzzy assessor to evaluate the fitness of individual architectures as compared to other techniques is flexible and reduces the computational time.

4.3.1. Genetic Algorithm Application. The process of natural selection inspired the creation of GA. The GA employed here utilizes a roulette wheel-type of selection to generate offspring's and an elitist approach for forming the new population (Konak, Coit, & Smith, 2006). The parameters used are described in Table 4.3.
Each model was run for 100 generations and 50 times to obtain a better assessment of the stochastic techniques used. The model with the highest architecture value in 50 iterations is presented here in each case. Increasing the generations to 300 did not affect the maximum architecture quality. Hence, it was reasonable to keep the same architecture's quality that was obtained in smaller simulation time. The population size was kept as 50, probability of mutation is 0.1, size of dormant selection for next

population is kept as 2, and lastly the population fraction maintained at the end of each epoch was 0.5.

The results presented in Figure 4.9. are architecture values over 100 generations using the GA. The results did not improve with increasing the generations to 200. The set of systems selected and the interfaces is presented as circular graph in Figure 4.9. The systems not selected are marked as red asterisks. The paramters used in GA are listed in Table 4.3. Systems selected are named in Table 4.4.

Table 4.3. The parameters used in GA

Generations	100,200
Population Size	50
Probability of Mutation	0.1
Tournament Selection Size	2
Population fraction kept for next generation	0.5

The best architecture obtained by GA is illustrated in Figure 4.9. A total number of 11 systems were selected and edges show the interfaces that exist amongst them. Each system and its capabilities are listed for comparison in Table 4.4. The architecture quality history obver many generations is shown in Figure 4.10.

Table 4.4. Systems and capabilities selected in best architecture by GA

Systems Selected by GA	**Capabilities Provided**
Systems 1,2 -Cutter	2
Systems 3-Helicopter	2
Systems 5-Aircraft	2
Systems 7,8,9,11,12-UAV	1
Systems 15-Fish Vessel	4
Systems 17 –Coordination Control	4
Systems 22-Communication	5

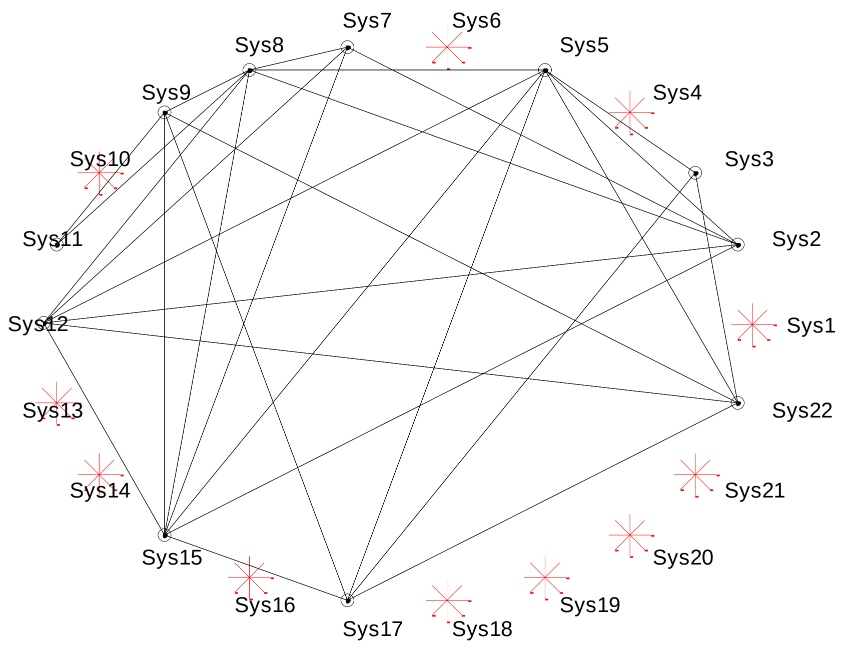


Figure 4.9. Systems selected in the SAR-22 SoS architecture through GA

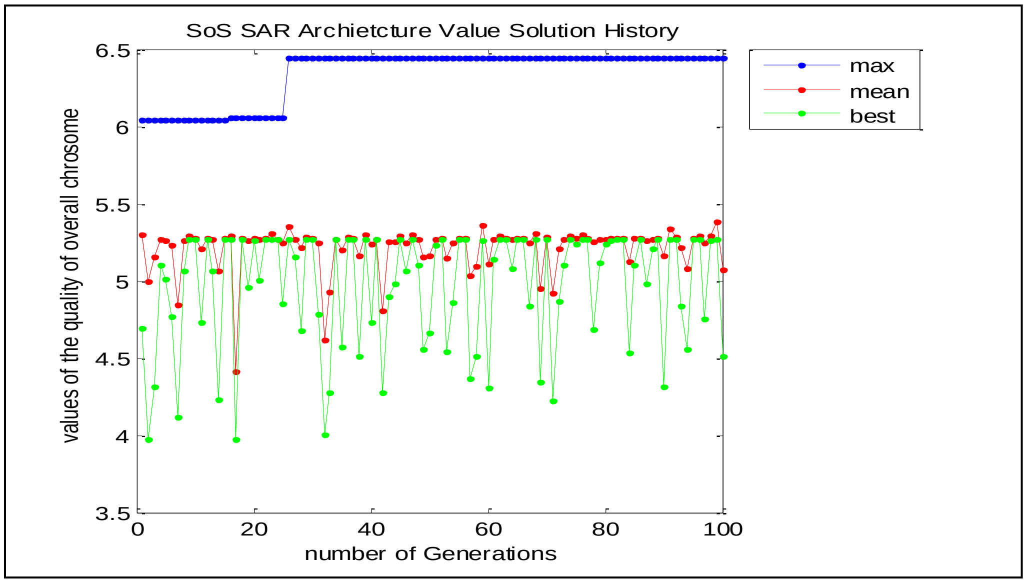


Figure 4.10. SoS architectural quality over generations through GA

4.3.2. BPSO Application. PSO was inspired by the social behavior of bird flocks and fish schools (Coello, 1999). PSO algorithms start with a group of a randomly generated population (particles in PSO). Population individuals are evaluated by a fitness function. Both update the population and search based on the best value achieved. PSO does not have genetic operators (e.g., crossover and mutation). Particles update is based on individual position, velocity and on the best position and velocity of the swarm leader. All the above procedures are valid for PSO and BPSO.

The major difference between BPSO with real-valued version is that velocities of the particles are defined in terms of probabilities that a bit will change to one or zero. Usually a sigmoid function is used to map all real valued velocities to the range of [0, 1]. The number of iterations was usually 100, population size was kept at 50, cognitive and social parameters were both equal to 2, and constriction factor was 1.

The maximum and minimum velocity was maintained between -4 and 4, and inertia weight decreased linearly based on number of iterations. These are all standard parameters in PSO. The parameters used for BPSO are listed in Table 4.5. The best architecture obtained is depicted in Figures 4.11. and 4.12. Each system and its capabilities are listed for comparison in Table 4.6.

Table 4.5. The parameters used in BPSO

Iterations	100,200
Population Size	40
Cognitive Parameter	2
Social Parameter	2
Constriction Factor	1
[velocity min, velocity max]	[-4, 4]
Inertia Weight	(Maximum iterations-Current iteration)/ Maximum iterations

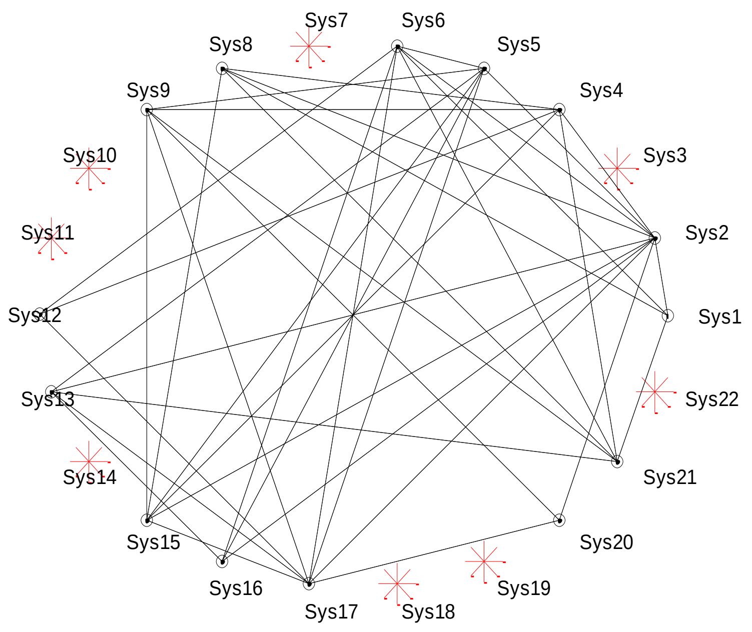


Figure 4.11. Systems selected in the SAR-22 SoS architecture through BPSO

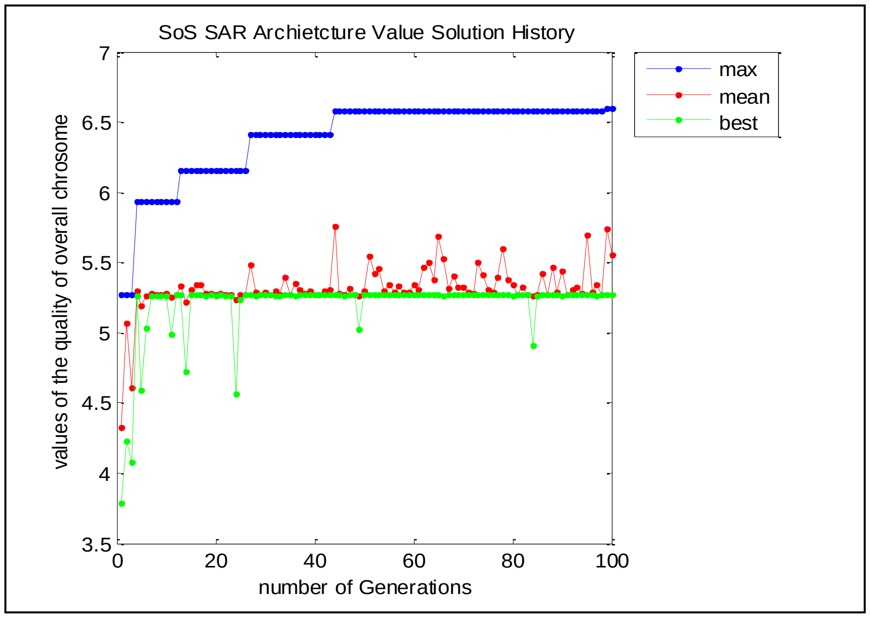


Figure 4.12. SoS architectural quality over generations through PSO

Table 4.6. Systems and the capabilities in the best architecture by the PSO

Systems Selected by BPSO	**Capabilities Provided**
Systems 1,2-Cutter	2
4-Helicopter	2
5,6-Aircraft	2
Systems 8, 9,12-UAV	1
Systems 13,15,16 -Fish Vessel	3
Systems 17 –Coordination Control	4
Systems 20, 21-Communication	5

The results from the BPSO can be considered as the first wave of meta-architecture. The following sections will be based on the results obtained from the BPSO.

4.4. NEGOTIATION APPLICATION

This section describes the techniques used by SoS manager to model the opponent for negotiations. A database of previous offers and counteroffers is required to implement the techniques utilized. In the current settings this approach does not involve collecting of data and the model structure is not universal. Based on the data involved the techniques might need to be modified to adapt to the scenarios.

Three system negotiation models namely; cooperative system negotiation model, non-cooperative system negotiation model, and semi-cooperative system negotiation model, have been previously used to generate the first round of negotiations. Volumes 5, 6, and 7 SERC describe the systems negotiation models.

The Cooperative System Negotiation Model is described in Volume 5 of the SERC report. The systems following this model behave cooperatively while negotiating with the SoS manager. The model of cooperative behavior is based on agent preferences and the negotiation length. Each system agent has two inherent behaviors of cooperativeness: Purposive (normal behavior) and Contingent (behavior driven by unforeseen circumstances). The approach models the tradeoff between the two behaviors for the systems. A fuzzy weighted average approach is used to arrive at the final proposed value.

Non-Cooperative System Negotiation Model is illustrated in Volume 6 of SERC report. In this model systems behave in their self-interest while negotiating with the SoS coordinator. A mathematical model of individual system’s participation capability and self-interest negotiation behavior is created. This methodology is an optimization-based generator of alternatives for strategically negotiating multiple items with multiple criteria. Besides, a conflict evaluation function that estimates prospective outcome for identified alternative is proposed.

The third and last system negotiation model is described in Volume 7, which illustrates the Semi-Cooperative System Negotiation Model. It exhibits the capability of being flexible or opportunistic: i.e., extremely cooperative or uncooperative based on different parameter values settings. A Markov-chain based model designed for handling uncertainty in negotiation modeling in an SoS. A model based on Markov chains is used for estimating the outputs. The work assigned by the SoS to the system is assumed to be a ``project'' that takes a random amount of time and a random amount of resources (funding) to complete.

4.4.1. Hierarchical Clustering. The scale function in R was used to both compute and standardize the difference between the offer made by the SoS and the counteroffers of the systems. This standardized data is then used for unsupervised clustering operations. Hierarchical clustering (Langfelder, Zhang, & Horvath, 2008) was conducted on the data to clarify the number of clusters that may be present. Algorithms for hierarchical clustering are generally either agglomerative, in which one starts at the leaves and successively merges clusters together; or divisive, in which one starts at the root and recursively splits the clusters. Agglomerative algorithms begin with each element as a separate cluster and merge them into successively larger clusters. Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters. It depends on the problem to use either an Agglomerative or Divisive approach.

In my work since we are trying to model the behavior of the opponent which is unknown, we expect to have many behaviors. The numbers of behaviors will increase with the number of issues involved in negotiation. It would make more sense to start with assuming each data point (or offer) obtained from the systems is a different behavior. Therefore we used the Agglomerative approach. Although even if we start with the divisive approach I think we should arrive at the same number of optimal clusters.

The method used here Ward's method began with n clusters of size 1 and iterated until all of the observations were incorporated into one of the clusters. Ward's method uses variance as a substitute of distance metrics or measures of association.

Figures 4.13. list all of the samples in the clustering dataset. It also indicates at what level of similarity any two clusters were joined. Horizontal lines indicate the distance at which clusters were joined. The first option would be a line just above a height of thirty, creating four clusters. Other options include any horizontal line below that level will increase the cluster size to more than seven. Seven negotiation behaviors are not expected to be present in the data. Therefore, the first option is chosen.
R command *cutree(fit, k=5)* can draw red borders around the k clusters in the dendrogram displayed in Figure 4.14.

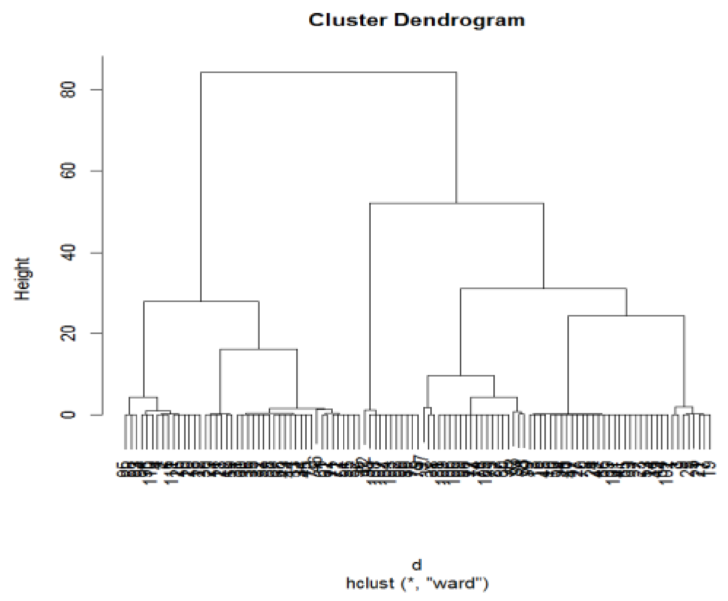


Figure 4.13. A dendrogram by Ward method showing four major clusters

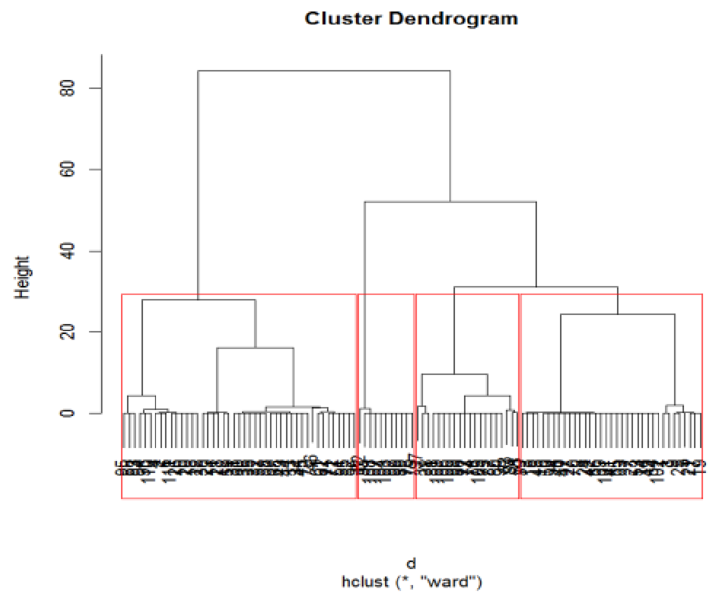


Figure 4.14. Four red boxes over the major clusters in the dendrogram

4.4.2. K-means clustering. All K-means clustering (MacQueen, 1967) required was the number of clusters (k) to be given as input. The results gathered from the dendrogram indicated to attempt k-means clustering with an input of 4 clusters in R. Another confirmation comes from Figure 4.15. which plots the within groups sum of squares by number of clusters extracted (Elbow method). It helped determine the appropriate number of clusters for k means by looking for a bend in the plot, which in this case seems to fall around the mark of 4 clusters (Everrit & Hothorn, 2009).

The number of points used in k means was 110, the number of dimensions was 3, and the number of expected classes is 4. The result had 41 points in class 1, 19 in class 2, 40 in class 3 and finally 11 in class 4. All of the pairwise dissimilarities (distances) between observations in the data set were computed to generate a silhouette plot. A silhouette represents each cluster. The entire clustering was displayed by combining the silhouettes into a single plot (Figure 4.16.). This technique assisted in selecting the number of clusters that maximized the silhouette coefficient.

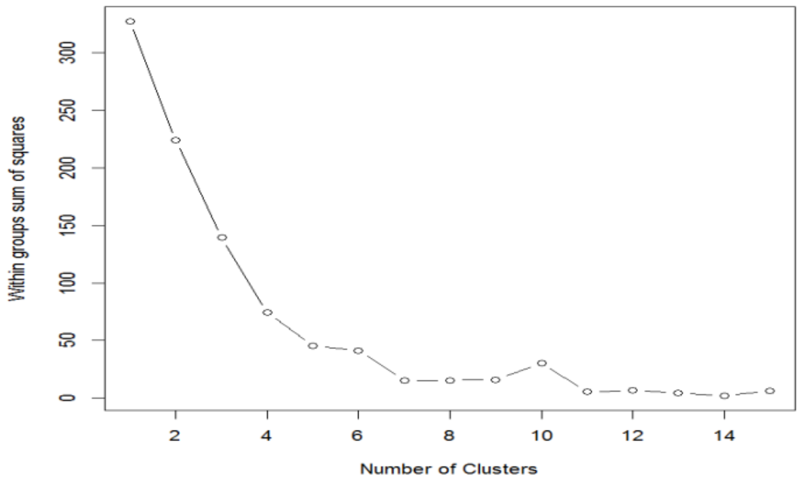


Figure 4.15. A plot of the within groups sum of squares by number of clusters

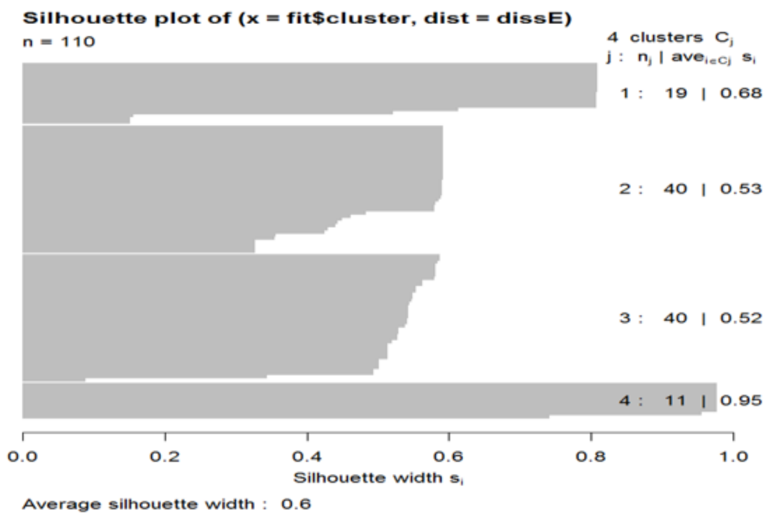


Figure 4.16. Silhouette plot based on the dissimilarity matrix of clustered data

The silhouette width is measured how close each point was from other points in the same cluster as compared to points in other clusters. If one point was closer to points in neighboring clusters than it was to points in its own cluster, then the clustering was inefficient and the value of the average silhouette width decreases (Kaufman & Rousseeuw, 2009). Figure 4.16. illustrates the following characteristics for each cluster:

- the number of plots per cluster is equal to the number of horizontal lines
- the number of points in each cluster, and their average silhouette width
- plots with an average silhouette width (0.6 in this case) value indicate stronger clustering

The average silhouette width value can be interpreted as follows:

1. 0.71-1.0—Good clustering has been achieved
2. 0.51-0.70--A reasonable clustering has been achieved
3. 0.26-0.50—clustering formed is a poor fit to the data
4. < 0.25--No clustering was identified

The systems' behaviors as reflected in the four clusters can be expressed as follows:

- Class 1-ready for participation in lesser time, asks for less funding and provides a stronger performance

- Class 2- Request for more time to participate, asks for less funding, provides a stronger performance
- Class 3- ready for participation in lesser time, asks for less funding and provides a weaker performance than any of the other clusters
- Class 4- Request for more time to participate, asks for more funding, provides a stronger performance

Systems in Class 1 can be referred to very cooperative. Class 2 and 3 systems can be referred to as are semi-cooperative. Class 4 can be referred to as selfish.

The coordinates of the four centers are given below as:

	V1	V2	V3
1	0.11403324	-0.3300645	-0.8465794
2	-1.39725071	-0.3527665	-1.5071888
3	1.32306041	2.7686001	0.4945384
4	0.03577032	-0.3396422	0.7059110

4.4.3. Network Architecture Based on Clustered Data. The clustered data then can be viewed as a mapping of inputs (3 variables) to target (classes) and is used to train a supervised learning network for prediction. The networks presented here are radial basis function network (RBFN) and linear vector quantization network (LVQN). RBFN and LVQN are utilized as a supervised method for prediction the class of the new incoming sample in this study. The labelled data is used to determine a set of prototype vectors that best represent each class. Further, these prototype vectors can be used to predict the class of a new incoming sample data as it is depicted in Figure 4.17.

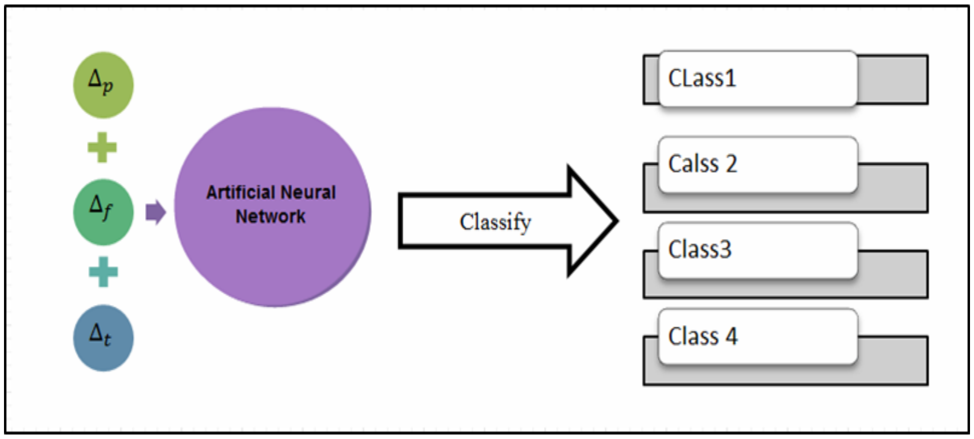
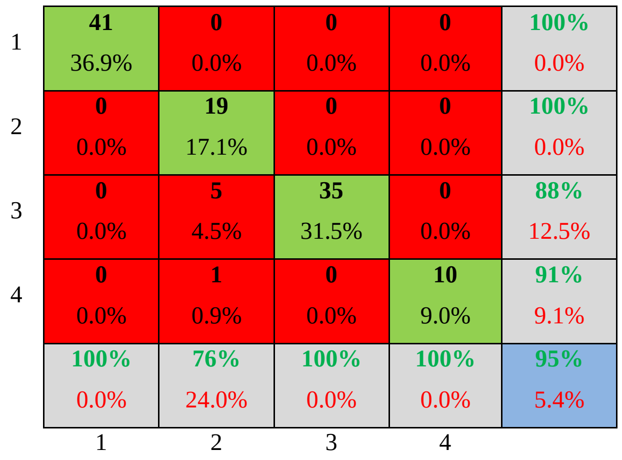


Figure 4.17. Neural networks architecture for supervised classification

4.4.3.1. RBFN architecture for the problem. The major difference between RBF networks and multi-layer perceptron is that the hidden units in RBF networks have a basis kernel function. Each hidden unit computes a value for the similarity between the input vector and its connection weights or centers. Based on these values the weights are updated.

RBFN has three layers: Step 1: Input layer – There is one neuron in the input layer for each predictor variable. There are three neurons in this case. The input neurons then feed the values to each of the neurons in the hidden layer. Step 2: Hidden layer – This layer has a variable number of neurons that starts with four neurons based on four centers. Each neuron consists of a radial basis function centered on a point with three dimensions. The resultant value from each neuron then is multiplied by the weight connection from hidden to the output layer and summed. Step 3: Output layer – The output layer has one output. The center and width for each basis function is computed using the centers obtained by k-means.

In performing this experiment, a separate data set for training containing 80 samples, and another separate data set for testing containing 31 samples was used. The samples were in randomly sampled order. The process repeated 20 times from the start to end at the best possible network. The network achieved a performance of 0.023 after adding neurons on each step to 40 neurons. The performance on prediction was good. Figure 4.18. displays the confusion matrix of RBFN.



1	**41** 36.9%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**100%** 0.0%
2	**0** 0.0%	**19** 17.1%	**0** 0.0%	**0** 0.0%	**100%** 0.0%
3	**0** 0.0%	**5** 4.5%	**35** 31.5%	**0** 0.0%	**88%** 12.5%
4	**0** 0.0%	**1** 0.9%	**0** 0.0%	**10** 9.0%	**91%** 9.1%
	100% 0.0%	**76%** 24.0%	**100%** 0.0%	**100%** 0.0%	**95%** 5.4%
	1	2	3	4	

Figure 4.18. Confusion Matrix for both Training and Testing by RBFN

4.4.3.2 LVQN Architecture for the problem. This is a supervised version of vector quantization. The labelled data is utilized to determine a set of prototype vectors that best represent each class. Further, these prototype vectors can be used of predict the class of a new incoming sample data. Learning Vector Quantization (LVQ) neural networks suggests abundant amount of robustness in clustering complex datasets.

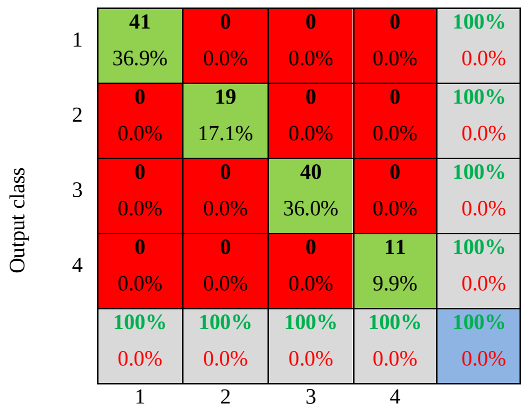
An LVQ consists of two layers, the first is competitive layer and the second is linear layer. The competitive layer learns to classify input vectors. The linear layer converts the competitive layer's classes into target classifications identified by the user. The classes learned by the competitive layer are denoted as subclasses and the classes of the linear layer as target classes. Both the competitive and linear layers have one neuron per (sub or target) class.

In conducting this experiment the same data set which was exploited for RBFN is utilized for training and testing. The samples were in randomly sampled order. We use the classes obtained by k-means as the primary prototypes to start LVQ. In LVQ, the learning rate $\varepsilon = 0.1$is considered. The network is trained to determine the weights given to connections between the hidden neurons and the output neuron. This network incorporates a random order incremental training algorithm for training the weights.

The network achieved a MSE of zero after adding neurons on each step to 110 neurons. The performance on prediction for classification was very robust. This is a supervised version of vector quantization. The labelled data is utilized to determine a set of prototype vectors that best represent each class.

Further, these prototype vectors can be used of predict the class of a new incoming sample data. We use the classes obtained by k-means as the primary prototypes to start LVQ. In LVQ, set the learning rate ε = 0.1.

Figure 4.17 depicts the confusion matrix of LVQN for both training and testing. The result shows that the performance of LVQN is way better than RBFN. The number of misclassification for RBFN is 6 out of 111 sample inputs; however, the misclassification for LVQN is zero which is shown in Figures 4.18. and 4.19.



Output class	1	2	3	4	
1	**41** 36.9%	**0** 0.0%	**0** 0.0%	**0** 0.0%	100% 0.0%
2	**0** 0.0%	**19** 17.1%	**0** 0.0%	**0** 0.0%	100% 0.0%
3	**0** 0.0%	**0** 0.0%	**40** 36.0%	**0** 0.0%	100% 0.0%
4	**0** 0.0%	**0** 0.0%	**0** 0.0%	**11** 9.9%	100% 0.0%
	100% 0.0%	100% 0.0%	100% 0.0%	100% 0.0%	100% 0.0%

Figure 4.19. Confusion Matrix for both Training and Testing by LVQN

Based on the network the classes (behavior type) of the system selected in the meta-architecture were predicted using the values of the respective systems first offers. Tables 4.7. and Table 4.8. provide a list of all systems selected in the meta-architecture through GA and BPSO and their corresponding behavior types.

Table 4.7. Systems and capabilities selected in best architecture by PSO

Systems Selected by BPSO	**Behavior Predicted**	**Capabilities Provided**
Systems 1, 2-Cutter	Class2, Class4	2
Systems 4-Helicopter	Class 1	2
Systems 5, 6-Aircraft	Class 2, Class 1	
Systems 8, 9,12-UAV	Class3, Class 1, Class 4	1
Systems 13,15,16-Fish Vessel	Class 3 , Class 4, Class 2	3
Systems 17 –Coordination Control	Class 1	4
Systems 20, 21-Communication	Class 3, Class 2	5

4.5. ESTIMATING UTILITY OF THE CURRENT OFFER

In a multi-attribute negotiation the solution space is n-dimensional (n>1) rather than a single dimensional line as in a single-attribute negotiation. This makes the negotiation strategy in multi-attribute negotiations complex: because the space is n-dimensional, every time an agent plans to concede, she needs to first decide the direction of concession.

Apparently there are many choices on the concession direction she can take: to concede on issue 1, …, n or different combinations of the issues. Specifically, the decision on the concession direction may also depend on the opponent’s preference because conceding on the issue more important to the opponent can make the offer more acceptable. Finally, to decide how much to concede is now more complicated because the direction can impact the amount as well. So the burden of computation and reasoning for the negotiation strategy is higher in a multi-attribute negotiation than in a single-attribute negotiation. The decision about how much concession has to be made in a particular issue is based on the negotiation strategy for the next round (Baarslag, Hindriks, & Jonker, 2013), willingness to collaborate, and the SoS's preference for acquiring that capability. After identifying the class of behavior the SoS coordinator can use a fuzzy inference engine to decide whether he wishes to accept the systems offer, reject the offer or further negotiate (Figure 4.20.). The model used is shown here as

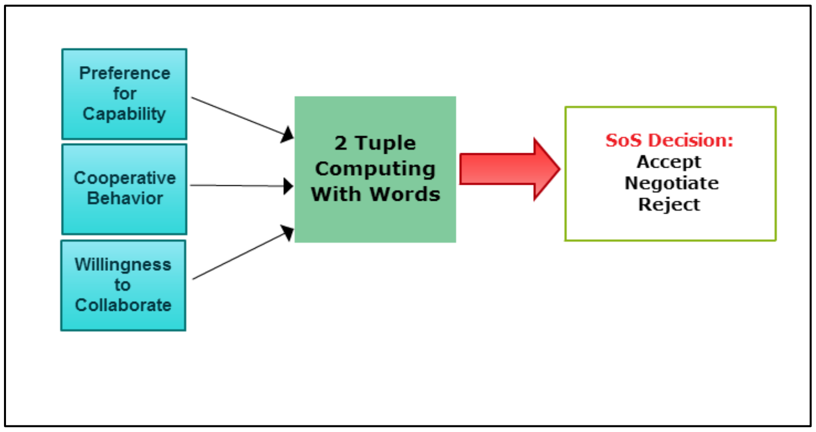

Preference for Capability
Cooperative Behavior
Willingness to Collaborate
2 Tuple Computing With Words
SoS Decision:
Accept
Negotiate
Reject

Figure 4.20. Fuzzy Network Architecture for decision making

This decision making problem is dealt as computing with words (CWW) whose linguistic term set is {low, medium, high, very high} as shown in Figure 4.21. If the aggregation is a tuple with terms *low* it is rejected, *medium* are negotiated further whereas, *very high and high* aggregation values are accepted immediately.The results with all systems selected from each class are shown in the Table 4.8.

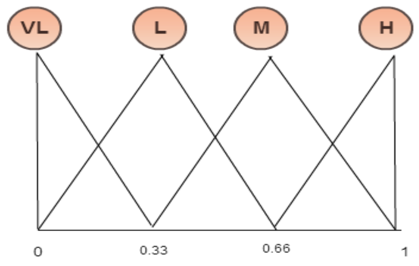


Figure 4.21. A set of four linguistic terms with their semantics

These results of meta-architecture generated by the BPSO indicate that the SoS manager after making a decision on the offers has been able to acquire capabilities 1, 2, 3, 4, and 5 (Table 4.8.). The SoS manager now needs to negotiate with systems to acquire more systems with the capabilities. The capabilities need to be acquired to complete the quorum of full 5 capabilities as shown in Figure 4.22.

System 17 is the only system selected in the meta-architecture that has capability 4. Hence although the estimated utility of the offer is low still it would be beneficial for the SoS manager to accept the system. System 5, 8 and 13 were rejected for further negotiation due its low aggregated value. The SoS manager has to negotiate with System 2, 12 and 16 to acquire an extra set of capabilities. The results of further negotiation with these systems still led to system 16 being accepted and systems 2 and 12 being rejected. The preference to acquire the capability was reduced hence the aggregate values using CWW for selection were lower and affected systems 2 and 12 more than system 16.

Table 4.8. Decision by SoS manager for each system in meta-architecture

Sys No	Cooperativeness	Willingness	Preference Capability	Aggregated Value	Selection
1	H	H	VH	2.33 or(H,+0.33)	Yes
2	L	H	M	1 or (M)	Neg
4	VH	L	VH	2 or (H)	Yes
5	H	L	L	0.66 or (M,-0.33)	No
6	VH	H	VH	2.66 or (VH,-	Yes
8	M	M	M	1 or (M)	No
9	VH	VH	M	2.33 or (H, 0.34)	Yes
12	L	H	M	1 or (M)	Neg
13	M	L	L	0.33 or (L,0.34)	No
15	L	VH	VH	2 or (H)	Yes
16	H	M	H	1.66 or (H,-0.33)	Neg
17	VH	L	H	1.66 or (H,-0.33)	Accepted
20	M	VH	VH	2.33 or (H, 0.34)	Yes
21	H	M	VH	2 or (H)	Yes

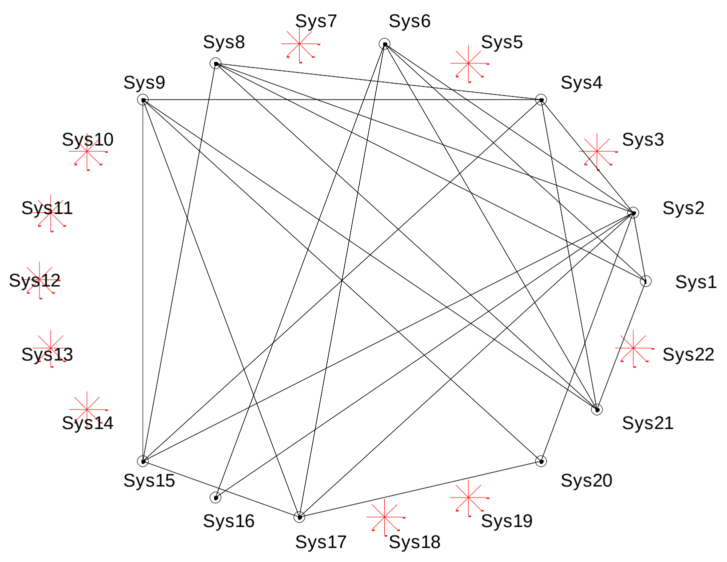


Figure 4.22. SoS negotiated architecture for wave 1 through BPSO

4.6. ILLUSTRATION OF THE SECOND WAVE

This section presents the second wave in SAR through FILA-SoS. The systems highlighted in yellow were selected at the end of negotiation process in the previous wave as shown in Table 4.9. Hence, they are preserved or maintained in the next wave meta-architecture. New systems replace the other systems with different values for the key attributes. To make things simple, we have not changed the order of the systems from one wave to the next, although this is possible.

Figure 4.23. gives the meta-architecture based on the domain inputs. Table 4.10. gives the list of system selected in the second wave. Table 4.11. gives the decision by SoS manager for each system in meta-architecture for wave 2. Table 4.12. gives the domain inputs for the generation of meta-architecture for the third wave. Table 4.13. gives a list of systems selected in wave 3 of the meta-architecture.

Table 4.9. Domain specific inputs for the second wave in SAR

SysNo	Type	Capability	IC_i	OC_i	P_i	D_i	LV_i	SP_i
1	Cutter	2	0.03	0.2	12	1	8.3	6
2	Cutter	2	0.03	0.2	12	1	8.3	6
3	Helicopter	2	0.5	0.6	10	0	10.0	8
4	Helicopter	2	0.1	0.2	20	1	10.0	8
5	Aircraft	2	0.6	1	15	1	10.0	10
6	Aircraft	2	0.1	0.5	10	1	10.0	10
7	UAV	1	0.3	0.3	5	2	1.7	2
8	UAV	1	0.1	0.1	7	1	1.7	2
9	UAV	1	0.1	0.1	7	1	1.7	2
10	UAV	1	0.3	0.3	5	1	1.7	2
11	UAV	1	0.3	0.3	5	1	1.7	2
12	UAV	1	0.3	0.3	5	1	1.7	2
13	Fish Vessel	3	0.02	1.5	20	1	5.0	4
14	Fish Vessel	3	0.02	1.5	20	1	5.0	4
15	Fish Vessel	3	0.03	0.5	10	1	5.0	4
16	Civ Ship	3	0.05	2	8	1	6.7	4
17	Coord Ctr	4	0.05	0.5	5	1	0.5	0
18	Coord Ctr	4	0.04	0.2	7	1	0.5	0
19	Comm	5	0.04	0.02	2	0	0.5	0
20	Comm	5	0.02	0.03	1	0	0.5	0
21	Comm	5	0.02	0.03	1	0	0.5	0
22	Comm	5	0.02	0.03	1	0	0.5	0

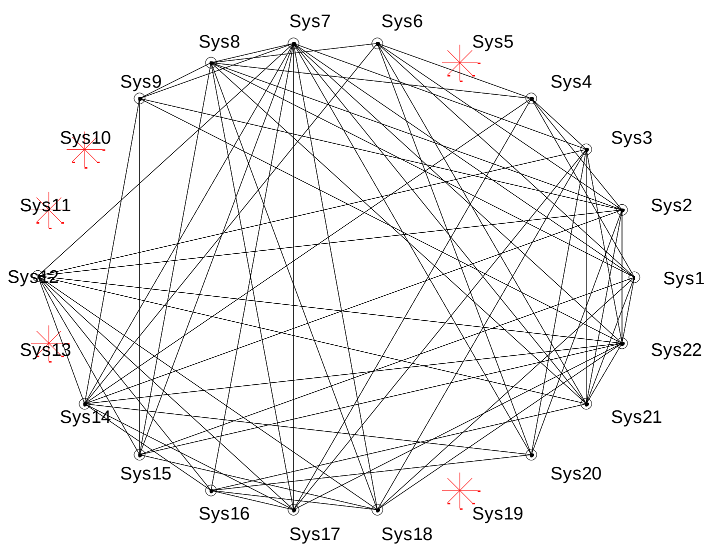


Figure 4.23. SoS meta-architecture for wave 2 through BPSO

Table 4.10. Systems selected in wave 2 meta-architecture

Systems Selected by PSO	**Behavior Predicted**	**Capabilities Provided**
Systems 1, 2-Cutter	Class 2, Class1	2
Systems 3, 4-Helicopter	Class 4, Class 1	2
Systems 6-Aircraft	Class 1	
Systems 7, 8, 9,12-UAV	Class 2, Class 3, Class 1, Class 2	1
Systems 14,15,16-Fish Vessel	Class 1 , Class 4, Class 2	3
Systems 17,18 – Coordination Control	Class 1, Class 3	4
Systems 20, 21,22-Communication	Class 3, Class 2, Class 4	5

The behavior of new systems may be different or same depending on their negotiation tactic. Whereas the older systems have the same behavior since they are preserved in the architecture form previous wave. The yellow highlighted classes of behavior belong to systems that were not pre-selected for this meta-architecture.

These results of meta-architecture indicate that the SoS manager after making a decision on the offers has been able to select the systems such that all capabilities required are bagged. Figure 4.23. shows the systems selected in the meta-architecture during wave 2 of SoS evolution and Figure 4.24. shows the negotiated architecture.

Table 4.11. Decision by SoS manager for each system in meta-architecture

Sys No	Cooperativeness	Willingness	Preference Capability	Aggregated Value	Selection
1	H	H	VH	2.33 or(H,+0.33)	Yes
2	VH	H	M	2 or (H)	Yes
3	L	L	M	0.33 or (L,0.33)	No
4	VH	L	VH	2 or (H)	Yes
6	VH	H	VH	2.66 or (VH,-.34)	Yes
7	H	VH	H	2.33 or(H,+0.33)	Yes
8	M	M	M	1 or (M)	No
9	VH	VH	M	2.33 or (H, 0.34)	Yes
12	H	L	M	1 or (M)	No
14	M	L	L	0.33 or (L,0.34)	No
15	L	VH	VH	2 or (H)	Yes
16	H	M	H	1.66 or (H,-0.33)	Neg
17	VH	L	H	1.66 or (H,-0.33)	Neg
18	M	H	VH	2 or (H)	Yes
20	M	VH	VH	2.33 or (H, 0.34)	Yes
21	H	M	VH	2 or (H)	Yes
22	L	L	VH	1 or (M)	No

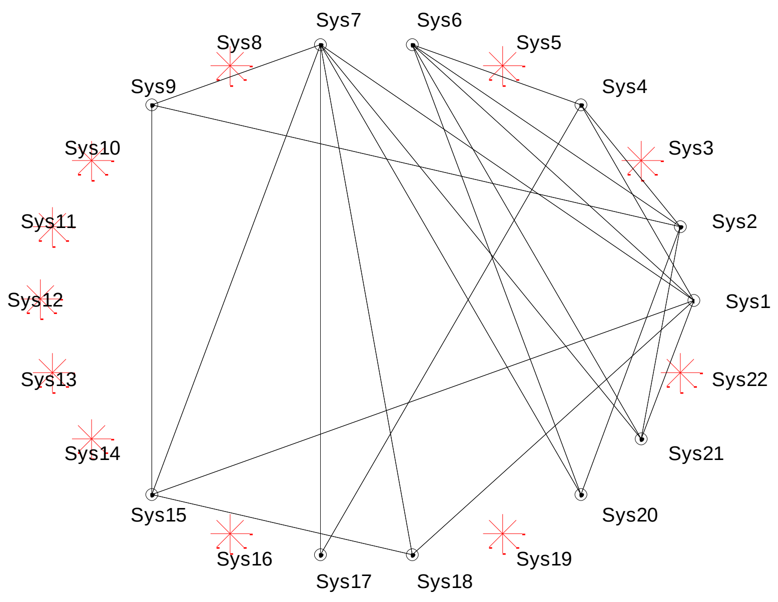


Figure 4.24. SoS negotiated architecture for wave 2 through BPSO

Figure 4.24. shows the final architecture that is agreeable between the SoS manager and the systems after negotiation during wave 2. Systems selected after negotiation are 1, 2, 4, 6, 7, 9, 15, 18, 20, and 21. System 3, 8, 12, 14 and 22 were rejected for further negotiation due its low aggregated value. The SoS manager has to negotiate with System 16 and 17 to acquire an extra set of capabilities. The results of further negotiation with these systems still led to system 16 and 17 being rejected. The preference to acquire the capability was reduced hence the aggregate values using CWW for selection were lower.

4.7. ILLUSTRATION OF THE THIRD WAVE

This section further extends the process of wave model through the third wave for SAR scenario. The behavior of new systems may be different or same depending on their negotiation tactic. Whereas the older systems have the same behavior since they are preserved in the architecture form previous wave. Figure 4.25. and Figure 4.26. show the meta-architecture and the negotiated architecture for wave 3 respectively.

Table 4.12. Domain specific inputs for the third wave in SAR.

SysNo	Type	Capability	IC_i	OC_i	P_i	D_i	LV_i	SP_i
1	Cutter	2	0.03	0.2	12	1	8.3	6
2	Cutter	2	0.03	0.2	12	1	8.3	6
3	Helicopter	2	0.02	0.9	10	0	10.0	8
4	Helicopter	2	0.1	0.2	20	1	10.0	8
5	Aircraft	2	0.5	1	15	0	10.0	10
6	Aircraft	2	0.1	0.5	10	1	10.0	10
7	UAV	1	0.3	0.3	5	2	1.7	2
8	UAV	1	0.4	0.1	5	1	1.7	2
9	UAV	1	0.1	0.1	7	1	1.7	2
10	UAV	1	0.4	0.1	5	1	1.7	2
11	UAV	1	0.4	0.1	7	1	1.7	2
12	UAV	1	0.3	0.3	7	1	1.7	2
13	Fish Vessel	3	0.02	1.5	20	1	5.0	4
14	Fish Vessel	3	0.02	1.5	20	1	5.0	4
15	Fish Vessel	3	0.03	0.5	10	1	5.0	4
16	Civ Ship	3	0.05	1	12	0	6.7	4
17	Coord Ctr	4	0.03	0.4	5	1	0.5	0
18	Coord Ctr	4	0.04	0.2	7	1	0.5	0
19	Comm	5	0.03	0.05	2	0	0.5	0
20	Comm	5	0.02	0.03	1	0	0.5	0
21	Comm	5	0.02	0.03	1	0	0.5	0
22	Comm	5	0.04	0.05	2	0	0.5	0

Table 4.13. Systems selected in wave 3 meta-architecture

Systems Selected by BPSO	**Behavior Predicted**	**Capabilities Provided**
Systems 1, 2-Cutter	Class 2, Class1	2
Systems 4-Helicopter	Class 1	2
Systems 5, 6-Aircraft	Class 1, Class 1	
Systems 7, 9,11-UAV	Class 2, Class 1, Class 3	1
Systems 13, 14,15-Fish Vessel	Class 3, Class 1 , Class 4,	3
Systems 17,18 – Coordination Control	Class 1, Class 3	4
Systems 19, 20, 21,22-Communication	Class 1, Class 3, Class 2, Class 4	5

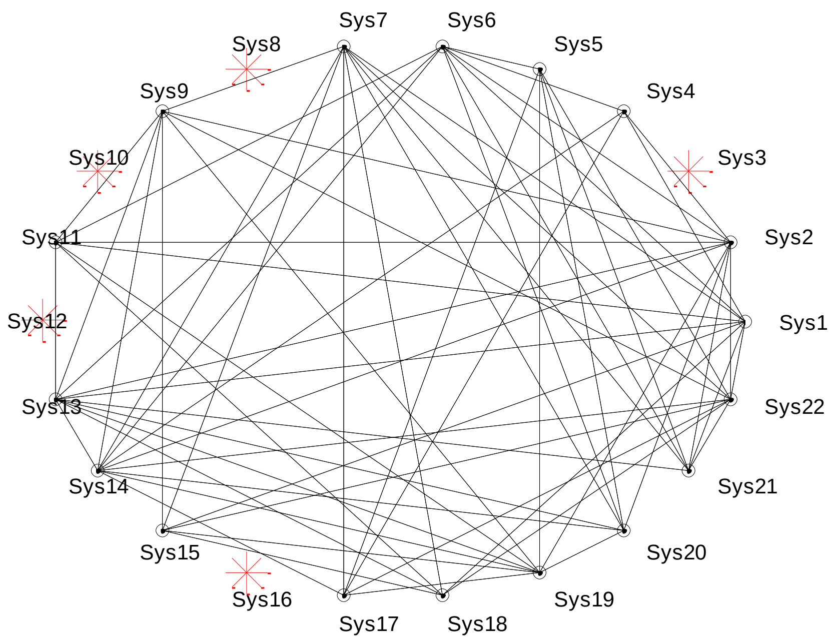


Figure 4.25. SoS meta-architecture for wave 3 through BPSO

These results of meta-architecture indicate that the SoS manager after making a decision on the offers has been able to select the systems such that all capabilities required are bagged. Table 4.14. gives the decision by SoS manager on systems.

Table 4.14. Decision by SoS for each system in meta-architecture

Sys No	Cooperativeness	Willingness	Preference Capability	Aggregated Value	Selection
1	H	H	VH	2.33 or(H,+0.33)	Yes
2	VH	H	M	2 or (H)	Yes
4	VH	L	L	1 or (M)	No
5	VH	M	VH	2.33 or(H,+0.33)	Yes
6	VH	H	VH	2.66 or (VH,-.34)	Yes
7	H	VH	H	2.33 or(H,+0.33)	Yes
9	VH	VH	M	2.33 or (H, 0.34)	Yes
11	M	L	M	.66 or (M,-0.33)	No

Table 4.14. Decision by SoS for each system in meta-architecture (cont.)

13	M	M	H	1.33 or (M, 0.33)	No
14	VH	L	L	1 or (M)	No
15	L	VH	VH	2 or (H)	Yes
17	VH	L	H	1.66 or (H,-0.33)	Neg
18	M	H	VH	2 or (H)	Yes
19	VH	H	VH	2.66 or (VH,-.34)	Yes
20	M	VH	VH	2.33 or (H, 0.34)	Yes
21	H	M	VH	2 or (H)	Yes
22	L	VH	H	1.66 or (H,-0.33)	Neg

Systems selected after negotiation are 1, 2, 5, 6, 7, 9, 15, 18, 19, 20, and 21. Systems 4, 11, 12, 13, were rejected for further negotiation due its low aggregated value as shown in Figure 4.26. The SoS manager has to negotiate with System 22 and 17 to acquire an extra set of capabilities.

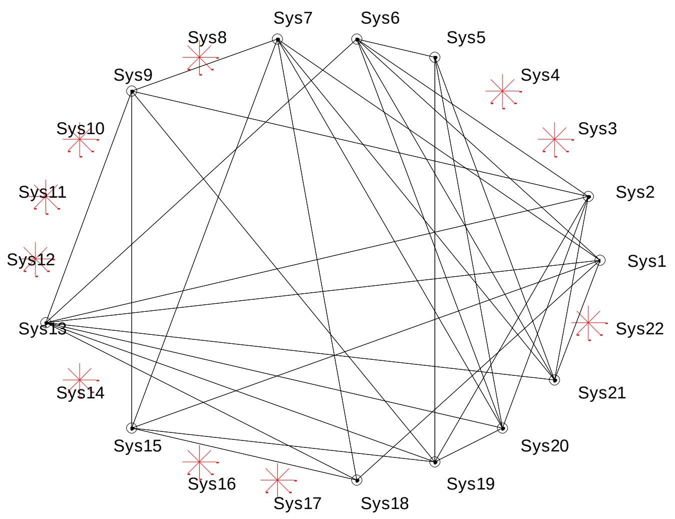


Figure 4.26. SoS negotiated architecture for wave 3 through BPSO

4.8. WAVE MODEL RESULTS

The architecture quality and values of various key performance attributes on scale of 1 to 4, of various SoS architectures is listed below in Figure 4.27. The discussion of the results obtained by the approach is presented in the forthcoming sections. A number of scenarios are developed to illustrate the meta-architecture generation and negotiation models.

Meta-Architecture wave 1

Quality	**3.11**
Performance	3.36
Affordability	3.01
Net-Centricity	2.55
Robustness	2.74

Negotiated-Architecture wave 1

Quality	**1.75**
Performance	2.8
Affordability	3.7
Net-Centricity	1.55
Robustness	1.74

Meta-Architecture wave 2

Quality	**3.29**
Performance	3.21
Affordability	2.98
Net-Centricity	3.64
Robustness	3.74

Negotiated-Architecture wave 2

Quality	**2.12**
Performance	1.8
Affordability	2.58
Net-Centricity	2.07
Robustness	1.33

Meta-Architecture wave 3

Quality	**3.21**
Performance	3.09
Affordability	3.78
Net-Centricity	3
Robustness	2.79

Negotiated-Architecture wave 3

Quality	**1.82**
Performance	2.8
Affordability	3.7
Net-Centricity	1.55
Robustness	1.74

Figure 4.27. Architecture assessment results for three waves

5. WHAT-IF ANALYSIS & DISCUSSION OF RESULTS

The model answers the research question “*What is the impact of different constituent system perspectives regarding participating in the SoS on the overall mission effectiveness of the SoS?”*with what-if analysis to assist the decision maker in preparing different alternatives and testing various scenarios. In this chapter the alternatives are generated from the implementation of the model to the SAR.

5.1. META-ARCHITECTURE GENERATION ANALYSIS

In this section we have developed what-if analysis based on the fact that different algorithms can be used to assess the impact of changes in system parameters, constitution, and configuration of the overall functionality and capability of the SoS. Each algorithm searches the solution space differently and can help in attaining better solutions.

This analysis will assist the SoS manager in future decision making by providing flexibility of technique. The meta-architecture generation technique helps in capturing the varied differences in the resources required by systems to prepare for participation. An architectural search methodology was applied to a generic SAR problem, and a set of architectures each with a high fitness, was obtained. The architectures generated via computational intelligence reduced both complexity and time. The architectures generated were the best combinations possible for the given domain problem. The stochastic heuristic techniques can assist in the systems architecting process by providing the systems architects with a set of feasible designs that can be developed into a near optimal architecture.

Although the best architecture obtained by the two techniques is slightly different for the same set of constraints, it means much good architecture exist in the modeled design space as shown in Table 5.1. Both GA and BPSO try to model the fitness function surface to reach the global maxima. The architecture value obtained by BPSO is higher than GA as shown in Table 5.2. This signifies the PSO was better able to map the surface of the fitness function generated by the fuzzy rules.

Each solution is architecture and has a fitness value. The higher the fitness value the higher the quality of the architecture. Solutions with higher fitness values are preserved over many generations in an evolutionary algorithm and finally the algorithm is terminated if the fitness value does not change over many generations. In the last generation the solution which has the highest architecture quality or fitness value is the best solution.

Table 5.1. Types of systems, capabilities and behaviors present in SoS

Systems Selected by GA	**Capabilities Provided**	**Behavior Predicted**
Systems 1,2 -Cutter	2	Class 2, Class 4
Systems 3-Helicopter	2	Class 1
Systems 5-Aircraft	2	Class 3
Systems 7,8,9,11,12-UAV	1	Class 3, Class 1,Class 2, Class 3,Class 4
Systems 15-Fish Vessel	4	Class 1
Systems 17 –Coordination Control	4	Class 3
Systems 22-Communication	5	Class 4

Table 5.2. Architecture Quality of the SoS for GA

Quality	**2.98**
Performance	3.16
Affordability	3.53
Net-Centricity	2.18
Robustness	2.39

The results of the selected systems in the meta-architected and the architecture quality are different from architecture generated by BPSO. Although the initial inputs are the same yet we may see different solutions. The quality of this meta-architecture by GA is slightly lower than that of BPSO. It is possible that a different algorithm might improve the architecture equality or on a different set of inputs the GA performs better than the BPSO.

5.2. ARCHITECTURE ASSESSMENT ANALYSIS

The architecture assessment model can be adjusted for different domains and stakeholders, changes in the environment, and relative priorities of the attributes can also be accommodated by ordering assessment rules. A what-if analysis based on the above criterion is presented to highlight its effects. This analysis displays the non-linearity in key performance attribute (KPA) tradeoffs, is able to accommodate any number of attributes for a selected SoS capability, and incorporate multiple stakeholder's understanding of KPA's.

Architecture assessment is completed through rules based on fuzzy assessor (Pape et al., 2013). These rules capture non-linearity in key performance parameters tradeoffs. Furthermore, fuzzy rules are able to comprehend multiple stakeholders' understanding of key performance attributes. Comparative significances of the attributes can also be accommodated by prioritizing assessment rules. The output is the value of a given architecture based on the assessment of the attributes. The architecture quality of the negotiated architecture is always less than or equal to the meta-architecture (Agarwal, Wang, & Dagli, 2015).

The solutions are initially represented as a vector of random numbers and using a sigmoid function is converted to binary value (Agarwal, Wang, & Dagli, 2015). Each solution is assessed by a fuzzy assessor which helps in reducing the complexity and computational time. Out of 20, some rules created to define the trade-offs between the many objectives are stated:

- If (Performance is high) and (Affordability is low) and Net-Centricity is high) and (Robustness is low) then (SoS_Arch_Fitness is medium)
- If (Performance is medium) and (Affordability is high) and Net-Centricity is

high) and (Robustness is high) then (SoS_Arch_Fitness is high)

- If (Performance is low) and (Affordability is medium) and Net-Centricity is high) and (Robustness is high) then (SoS_Arch_Fitness is high)
- If (Performance is medium) and (Affordability is medium) and Net-Centricity is low) and (Robustness is low) then (SoS_Arch_Fitness is low)

The rules are created in the fuzzy assessor to evaluate architectures in Section 4 seem to support affordability and performance as compared to robustness and net-centricity. Different set of rules for the same assessor may give different values of attributes and hence might also result different set of architectures. The rules above might represent different stakeholders in the SoS. Changing the rules might give a different assessment to the same architecture. This phenomenon is presented below where a SoS architecture is chosen in Figure 5.1. and then evaluated by two different fuzzy assessors.

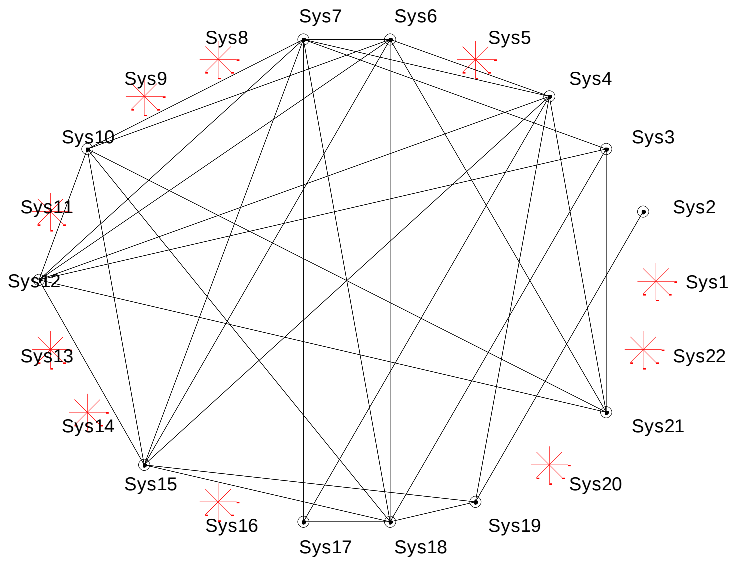


Figure 5.1. Meta-architecture selected for evaluation

For the same value of the performance as 3.16, affordability as 2.5, net-centricity as 2.12 and robustness as 3.27 different architecture qualities are reached. The fuzzy assessor with no change in rules gives an architecture quality of 3.24 whereas the new fuzzy assessor gives a value of 2.98.

Since only two systems 19 and 21 were selected provided the inter-communication capability and net-centricity was more dominant in providing a higher architecture quality hence the second quality is a little lower than the first (Hassan et al., 2005). Besides affordability affects the architecture quality as it is low in this architecture. This analysis can enable the decision-maker to choose the architecture that suits best based on stakeholder views.

5.3. ADAPTIVE NEGOTIATION ANALYSIS

Similarly, the behavioral aspect of systems is tackled through an adaptive SoS negotiation strategy. Different behaviors of the systems for the same architecture can help us generate possible negotiated architecture qualities. This is a very quick and effective approach to adapt communication strategies in SoS environment. This section entails what-if analysis based on simulating rules of engagement & behavior settings such as: all systems are selfish, all systems are opportunistic, and all systems are cooperative or a combination. It provides answers to questions such as whether an individual system can be impacted by negotiation strategies of the SoS and how so.

This includes an examination of architecture quality obtained under different behavioral settings including such as when does non-cooperative behavior dominates the acquisition environment or when does semi-cooperative behavior dominate or when does cooperative behavior dominate. Various incentive mechanisms can be analyzed when there is uncertainty in individual system performance outcomes. Table 5.3. gives the setting of negotiation decision making in case of random behavior of systems where ‘VH’ corresponds to extremely coo-operative, ‘H’ and ‘M’ relate to semi-cooperative, and ‘L’ denotes the non-cooperative behavior.

Table 5.3. Decision by SoS manager for each system in meta-architecture for BPSO

Sys No	Cooperativeness	Willingness	Preference Capability	Aggregated Value	Selection
1	H	H	VH	2.33 or(H,+0.33)	Yes
2	L	H	M	1 or (M)	Neg
4	VH	L	VH	2 or (H)	Yes
5	H	L	L	0.66 or (M,-0.33)	No
6	VH	H	VH	2.66 or (VH,-0.34)	Yes
8	M	M	M	1 or (M)	No
9	VH	VH	M	2.33 or (H, 0.34)	Yes
12	L	H	M	1 or (M)	Neg
13	M	L	L	0.33 or (L,0.34)	No
15	L	VH	VH	2 or (H)	Yes
16	H	M	H	1.66 or (H,-0.33)	Neg
17	VH	L	H	1.66 or (H,-0.33)	Accepted
20	M	VH	VH	2.33 or (H, 0.34)	Yes
21	H	M	VH	2 or (H)	Yes

Scenario 1. To visualize a condition if all selected systems in the meta-architecture at any stage behaved cooperatively, the wave 1 meta-architecture was selected. The behavior of systems 1, 2, 5, 8, 12, 13, 15, 16, 20, and 21 were updated to 'VH' as represented in Table 5.4. Rests of the selected systems were already cooperative. Table 5.3. can be compared with Table 5.4. for easier understanding. This resulted in the following changes:

1. Systems 2, 12, 16, which were earlier in the negotiated category were now accepted
2. Systems 8 which was earlier in rejected category was now being negotiated

Besides the architecture quality had a small improvement due to more systems with capabilities added. This reinforced the robustness and the net-centricity of the systems. The attributes of negotiated architecture are given in Table 5.5.

Table 5.4. Decision for all cooperative systems in meta-architecture

Sys No	Cooperativeness	Willingness	Preference Capability	Aggregated Value	Selection
1	VH	H	VH	2.66 or(H,+0.66)	Yes
2	VH	H	M	2 or (M)	Yes
4	VH	L	VH	2 or (H)	Yes
5	VH	L	L	1 or (M)	No
6	VH	H	VH	2.66 or (VH,-0.34)	Yes
8	VH	M	M	1.66 or (H,-0.33)	Neg
9	VH	VH	M	2.33 or (H, 0.34)	Yes
12	VH	H	M	2 or (M)	Yes
13	VH	L	L	1 or (M)	No
15	VH	VH	VH	3 or (VH)	Yes
16	VH	M	H	2 or (M)	Yes
17	VH	L	H	1.66 or (H,-0.33)	Accepted
20	VH	VH	VH	3 or (VH)	Yes
21	VH	M	VH	2.33 or (H, 0.34)	Yes

Table 5.5. Negotiated-Architecture of wave 1 under cooperative conditions

Negotiated-Architecture	Quality	Performance	Affordability	Net-Centricity	Robustness
Random behavior	1.75	2.8	3.7	1.55	1.74
All Cooperative	2.83	3	3.25	2.67	2.98

Based on the analysis of CWW different set of systems are finally selected for implementation of the SoS architecture. The quality and values of attributes of this negotiated architecture are given for comparison in Table 5.5. The architecture quality is higher than before as more systems are selected. The negotiated architecture can be only as good as the meta-architecture itself.

Scenario 2. In the second scenario all selected systems in the meta-architecture are designed with a behavior of non-cooperativeness. This condition will help realize the effect of behavior in such a setting. The behavior of systems is represented in Table 5.6.

Table 5.6. Decision for all non-cooperative systems in meta-architecture

Sys No	Cooperativeness	Willingness	Preference Capability	Aggregated Value	Selection
1	L	H	VH	1.66 or (H,-0.33)	Neg
2	L	H	M	2 or (M)	Yes
4	L	L	VH	1 or (M)	No
5	L	L	L	L	No
6	L	H	VH	1.66 or (H,-0.33)	Neg
8	L	M	M	0.6 or (M, 0.3)	No
9	L	VH	M	1.33 or (M, 0.34)	No
12	L	H	M	1 or (M)	No
13	L	L	L	0 or (L)	No
15	L	VH	VH	2 or (H)	Yes
16	L	M	H	1 or (M)	No
17	L	L	H	0.66 or (M,-	No
20	L	VH	VH	2 or (H)	Yes
21	L	M	VH	1.33 or (M, 0.34)	No

This resulted in the following changes:

1. Systems 8, 4, 9, 12 , 17, and 21 which was earlier in accepted category are now as rejected systems
2. Systems whose configuration remains the same after the change in behaviours are 2,5,8,13,15, and 20
3. Systems that were accepted earlier were now being negotiated are Systems 1 and 16

Besides the architecture quality decreased due to less systems selected due to non-cooperative behavior, other attributes value remaining the same. The results are listed in Table 5.7.

Table 5.7. Negotiated-Architecture of wave 1 under non-cooperative conditions

Negotiated-Architecture	Quality	Performance	Affordability	Net-Centricity	Robustness
Random behavior	1.75	2.8	3.7	1.55	1.74
All Non-Cooperative	1.22	1.2	3.89	2	1.34

Scenario 3. In the final scenario all selected systems in the meta-architecture are designated with a behavior of semi-cooperativeness. This condition will help realize the effect of behavior in such a setting. The behavior of systems is represented in Table 5.8.

Negotiated-Architecture of wave 1 under semi-cooperative conditions results are listed in Table 5.9.

Table 5.8. Decision for all semi-cooperative systems in meta-architecture

Sys No	Cooperativeness	Willingness	Preference Capability	Aggregated Value	Selection
1	M	H	VH	1.66 or (H,-0.33)	Neg
2	M	H	M	2 or (M)	Yes
4	M	L	VH	2 or (H)	Yes
5	M	L	L	1 or (M)	No
6	M	H	VH	2.66 or (VH,-0.34)	Yes
8	M	M	M	1.66 or (H,-0.33)	Neg
9	M	VH	M	2.33 or (H, 0.34)	Yes
12	M	H	M	2 or (M)	Yes
13	M	L	L	1 or (M)	No
15	M	VH	VH	3 or (VH)	Yes
16	M	M	H	2 or (M)	Yes
17	M	L	H	1.66 or (H,-0.33)	Accepted
20	M	VH	VH	3 or (VH)	Yes
21	M	M	VH	2.33 or (H, 0.34)	Yes

Table 5.9. Negotiated-Architecture of wave 1 under semi-cooperative conditions

Negotiated-Architecture	Quality	Performance	Affordability	Net-Centricity	Robustness
Random behavior	1.75	2.8	3.7	1.55	1.74
All Non-Cooperative	2.01	1.2	3.89	2	1.34

This resulted in the following changes:

1. Systems 8 which was earlier in accepted category was now being rejected systems 4, 9, 12 , 17, and 21
2. Systems whose configuration remains the same after the change in behaviors are 2,5,8,13,15, and 20
3. Systems that were accepted earlier were now being negotiated are Systems 1 and 16

Besides the architecture quality decreased due to less systems selected due to non-cooperative behavior, other attributes value remaining the same. The results are listed in Table 5.9.

These scenarios explain how after arriving at the meta-architecture, SoS manger may obtain different architecture qualities based on system behaviors. There could be three scenarios each for each wave in the SoS. Each scenario is divided in domination of cooperative, semi-cooperative and non-cooperative behaviors. Such scenarios are able to answer thw question that in the same wave if all systems were cooperative, all semi-cooperative how it will affect the architecture quality. The inferences drawn from this analysis are as follows:

1. It is quite predictable to have cooperative and semi-cooperative systems selected more often than non-cooperative systems
2. Final systems behaviour configuration changes in the architecture based on number of waves
3. The negotiated architecture quality is lower than the meta-architecture quality
4. Simulating rules of engagement & behaviour settings: all systems are selfish, all systems are opportunistic, all systems are cooperative or a combination can be beneficial for future analysis
5. The architecture quality improves with increase in cooperativeness of systems

The next section gives some scenarios to further show this approach which involves meta-architecture generation and SoS negotiation models to implement our ideas.

5.4. VARIOUS DIFFERENT SCENARIOS

This section explains different scenarios to judge how well decision-makers can instrument mentioned strategies to manage uncertainty in complex adaptive SoS.

The first scenario has the settings such that capabilities are given preferences of each other as compare to previous waves. Previously the preference of capability could be changed during the wave transition by the SoS manager. Here it is demonstrated how different inputs and preference produce different architectures.

The conditions that affect the architecture quality of SoS are also due to changes in costs for developing the interfaces are assigned to each system, as well as a cost for operating the system. The deadline for development of an interface which may be different in each wave of acquisition assigned out of three values 0 – ready now, 1 – will be ready by the end of this wave, or 2 – won't be ready this wave can affect the value of key performance attributes. Variables such as SoS funding and capability priority can be changed as the acquisition progresses though wave cycles causing different architectures to be selected.

The costs for development were rough estimates of official and informal budgetary evaluations for interfacing with communications systems and integrating the mission systems to be able to interoperate. The costs to operate aircraft or other systems were also determined in a similar fashion. The numbers usually kept small to accommodate the sensitivity in the analysis. They were within 0.1 to 20. This scenario is about giving a pre-defined preference to each capability by the SoS manager. This preference can be continued to subsequent waves or changed in each wave.

Table 5.10. starts with domain inputs for scenario1, Figure 5.2. shows the meta-architecture for scenario 1 and Figure 5.3. shows the negotiated architecture for scenario 1. Figures 5.4. and 5.5. show the meta-architecture and the negotiated architecture for scenario 2 respectively. Similarly for scenario 3 Figures 5.6. and Figure 5.7. show the meta-architecture and negotiated architecture whereas scenario 4 is depicted in Figure 5.8. and Figure 5.9. Table 5.11. and Table 5.12. give the decision matrix and systems selected for meta-architecture. Tables 5.13. , 5.16. , 5.19. give the new domain inputs, Tables 5.14., 5.18., and 5.21. the decision matrix and Tables 5.15., 5.17., 5.20. the systems selected for scenario 2, 3 and 4 respectively.

Table 5.10. Domain Inputs for Scenario 1

SysNo	Type	Capability	I/FDevCos	OpsCost/I	Perf	DevTime
1	Cutter	2	0.03	0.2	12	1
2	Cutter	2	0.03	0.2	12	1
3	Helicopter	2	0.1	0.2	20	1
4	Helicopter	2	0.1	0.2	20	1
5	Aircraft	2	0.1	0.5	10	1
6	Aircraft	2	0.1	0.5	10	1
7	UAV	1	0.2	0.2	5	1
8	UAV	1	0.2	0.2	5	1
9	UAV	1	0.2	0.2	5	1
10	UAV	1	0.1	0.1	7	1
11	UAV	1	0.1	0.1	7	1
12	UAV	1	0.1	0.1	7	1
13	Fish Vessel	3	0.03	0.5	10	1
14	Fish Vessel	3	0.03	0.5	10	1
15	Fish Vessel	3	0.03	0.5	10	1
16	Civ Ship	3	0.05	2	8	1
17	Coord Ctr	4	0.05	0.5	5	1
18	Coord Ctr	4	0.05	0.5	5	1
19	Comm	5	0.07	0.01	1	0
20	Comm	5	0.07	0.01	1	0
21	Comm	5	0.07	0.01	1	0
22	Comm	5	0.02	0.03	1	0

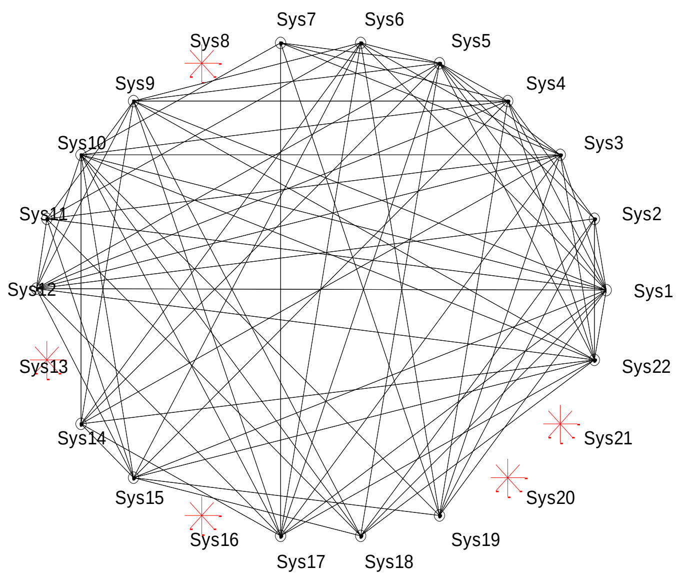


Figure 5.2. Meta-Architecture for scenario 1

Table 5.11. Decision by SoS manager for each system in meta-architecture

Sys No	Cooperativeness	Willingness of Collaboration	Preference Capability	Aggregated Value	Selection
1	H	H	L	1.33 or(M,+0.33)	Neg
2	L	H	L	0.66 or (M,-0.33)	No
3	L	L	H	0.66 or (M,-0.33)	No
4	VH	M	H	2 or (H)	Yes
5	H	M	VH	2 or (H)	Yes
6	VH	H	VH	2.66 or (VH,-0.34)	Yes
7	M	M	M	1 or (M)	No
9	VH	VH	M	2.33 or (H, 0.34)	Yes
10	VH	M	M	1.66 or (H,-0.33)	Neg
11	VH	H	M	2 or (H)	Yes
12	L	H	M	1 or (M)	Neg
14	M	L	H	1 or (M)	Neg
15	H	VH	H	2.33 or (H, 0.34)	Yes
17	H	M	VH	2 or (H)	Yes
18	VH	L	VH	2 or (H)	Yes
19	M	VH	VH	2.33 or (H, 0.34)	Yes
22	H	M	VH	2 or (H)	Yes

Table 5.12. Systems and capabilities in Scenario 1

Systems Selected in Meta-Architecture	**Capabilities Provided**	**Systems Selected in Negotiated Architecture**	**Capabilities Provided**
Systems 1,2-Cutter	2	None	2
3,4-Helicopter	2	4-Helicopter	2
5,6-Aircraft	2	5,6-Aircraft	2
Systems 7, 9,10, 11, 12-UAV	1	Systems 9, 11 UAV	1
Systems 14,15-Fish Vessel	3	Systems 15-Fish Vessel	3
Systems 17, 18 – Coordination Control	4	Systems 17, 18 – Coordination Control	4
Systems 19, 22-Communication	5	Systems 19, 22-Communication	5

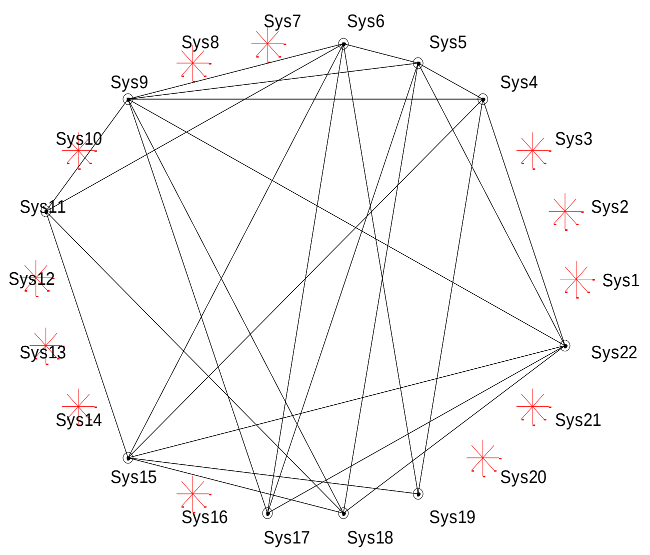


Figure 5.3. SoS negotiated architecture for scenario 1

Table 5.13. Domain Inputs for Scenario 2

SysNo	Type	Capability	I/FDevCos	OpsCost/I	Perf	DevTime
1	Cutter	2	0.04	0.5	10	1
2	Cutter	2	0.04	0.5	10	1
3	Helicopte	2	0.2	0.4	15	1
4	Helicopte	2	0.1	0.2	20	1
5	Aircraft	2	0.1	0.5	10	1
6	Aircraft	2	0.1	0.5	10	1
7	UAV	1	0.1	0.1	8	1
8	UAV	1	0.1	0.1	8	1
9	UAV	1	0.2	0.2	5	1
10	UAV	1	0.1	0.1	7	1
11	UAV	1	0.1	0.1	7	1
12	UAV	1	0.1	0.1	7	1
13	Fish Vesse	3	0.03	0.5	12	1
14	Fish Vesse	3	0.03	0.5	12	1
15	Fish Vesse	3	0.03	0.5	10	1
16	Civ Ship	3	0.05	2.5	8	1
17	Coord Ctr	4	0.05	0.5	5	1
18	Coord Ctr	4	0.05	0.5	5	1
19	Comm	5	0.07	0.01	1	0
20	Comm	5	0.09	0.1	1	0
21	Comm	5	0.09	0.1	1	0
22	Comm	5	0.02	0.03	1	0

Table 5.14. Decision by SoS manager for each system in meta-architecture

Sys No	Cooperativeness	Willingness of Collaboration	Preference Capability	Aggregated Value	Selection
1	VH	VH	L	2 or (H)	Yes
3	L	L	H	0.66 or (M,-0.33)	No
4	VH	M	H	2 or (H)	Yes
5	H	M	VH	2 or (H)	Yes
6	VH	H	VH	2.66 or (VH,-0.34)	Yes
9	VH	VH	M	2.33 or (H, 0.34)	Yes
11	VH	H	M	2 or (H)	Yes
12	L	H	M	1 or (M)	Neg
13	M	L	H	1 or (M)	Neg
15	H	VH	H	2.33 or (H, 0.34)	Yes
17	H	M	VH	2 or (H)	Yes
18	VH	L	VH	2 or (H)	Yes
19	M	VH	VH	2.33 or (H, 0.34)	Yes
21	VH	M	VH	2.33 or (H, 0.34)	Yes
22	H	M	VH	2 or (H)	Yes

Table 5.15. Systems and capabilities in Scenario 2

Systems Selected in Meta-Architecture	**Capabilities Provided**	**Systems Selected in Negotiated Architecture**	**Capabilities Provided**
Systems 1-Cutter	2	Systems 1-Cutter	2
3,4-Helicopter	2	4-Helicopter	2
5,6-Aircraft	2	5,6-Aircraft	2
Systems 9,11, 12-UAV	1	Systems 9, 11 UAV	1
Systems 13,15-Fish Vessel	3	Systems 15-Fish Vessel	3
Systems 17, 18 – Coordination Control	4	Systems 17, 18 – Coordination Control	4
Systems 19, 21, 22-Communication	5	Systems 19, 21, 22-Communication	5

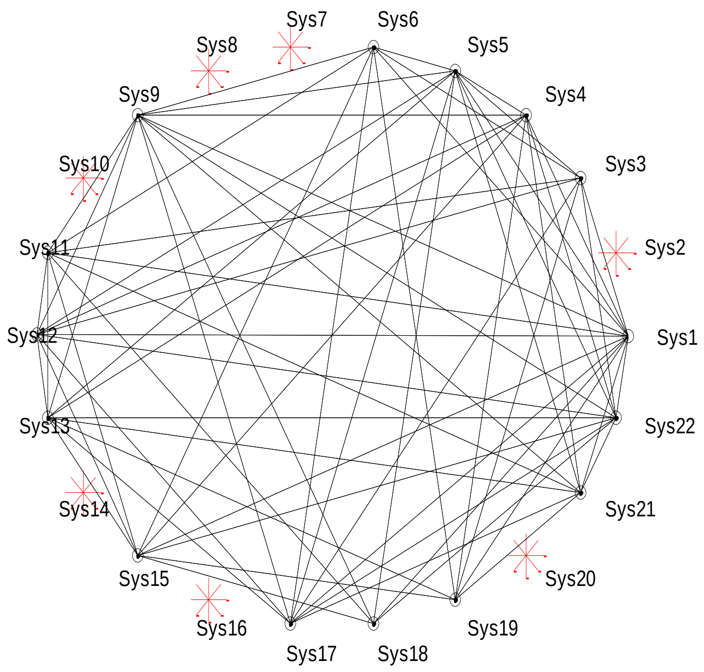


Figure 5.4. Meta-Architecture for scenario 2

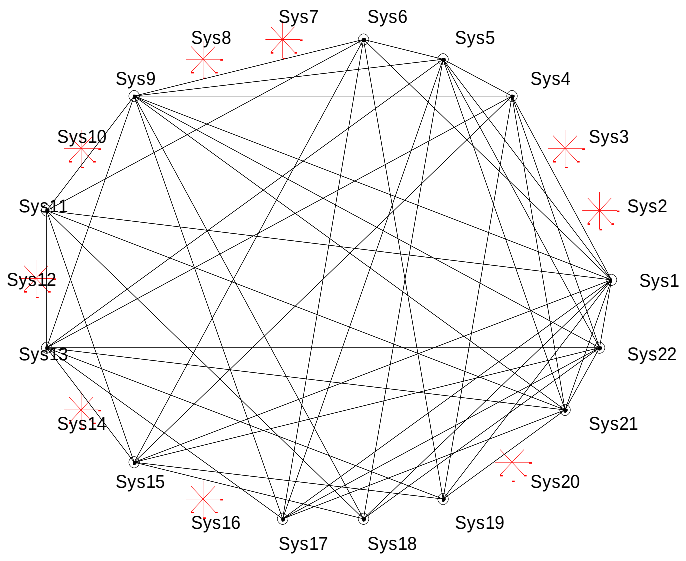


Figure 5.5. SoS negotiated architecture for scenario 2

Table 5.16. Domain Inputs for Scenario 3

SysNo	Type	Capability	I/FDevCos	OpsCost/I	Perf	DevTime
1	Cutter	2	0.04	0.5	10	1
2	Cutter	2	0.04	0.5	10	1
3	Helicopte	2	0.2	0.4	15	1
4	Helicopte	2	0.2	0.4	20	1
5	Aircraft	2	0.1	0.5	10	1
6	Aircraft	2	0.1	0.5	10	1
7	UAV	1	0.4	0.1	8	1
8	UAV	1	0.4	0.1	8	1
9	UAV	1	0.4	0.1	5	1
10	UAV	1	0.4	0.1	7	1
11	UAV	1	0.4	0.1	7	1
12	UAV	1	0.4	0.1	7	1
13	Fish Vesse	3	0.03	0.5	12	1
14	Fish Vesse	3	0.03	0.5	12	1
15	Fish Vesse	3	0.03	0.5	10	1
16	Civ Ship	3	0.05	2.5	8	1
17	Coord Ctr	4	0.05	0.5	5	1
18	Coord Ctr	4	0.05	0.5	5	1
19	Comm	5	0.09	0.1	1	0
20	Comm	5	0.09	0.1	1	0
21	Comm	5	0.09	0.1	1	0
22	Comm	5	0.09	0.1	1	0

Table 5.17. Systems and capabilities in Scenario 3

Systems Selected in Meta-Architecture	**Capabilities Provided**	**Systems Selected in Negotiated Architecture**	**Capabilities Provided**
Systems 2-Cutter	2	Systems 2-Cutter	2
3,4-Helicopter	2	4-Helicopter	2
5,6-Aircraft	2	None	2
Systems 8, 9,11, 12-UAV	1	Systems 9 UAV	1
Systems 13,14, 16-Fish Vessel	3	Systems 14-Fish Vessel	3
Systems 18 –Coordination Control	4	Systems 18 – Coordination Control	4
Systems 19, 20, 21, 22-Communication	5	Systems 19, 20-Communication	5

Table 5.18. Decision by SoS manager for each system in meta-architecture

Sys No	Cooperativeness	Willingness of Collaboration	Preference Capability	Aggregated Value	Selection
2	VH	VH	L	2 or (H)	Yes
3	L	VH	H	1.66 or (H,-0.33)	Neg
4	M	VH	H	2 or (H)	Yes
5	M	L	VH	1.33 or (M,0.33)	Neg
6	H	L	VH	1.66 or (H,-0.33)	Neg
8	H	H	M	1.66 or (H,-0.33)	Neg
9	VH	H	M	2 or (H)	Yes
11	H	H	M	1.66 or (H,-0.33)	Neg
12	H	H	M	1.66 or (H,-0.33)	Neg
13	L	M	H	1 or (M)	No
14	VH	M	H	2 or (H)	Yes
16	M	M	VH	1.66 or (H,-0.33)	Neg
18	L	VH	VH	2 or (H)	Yes
19	VH	L	VH	2 or (H)	Yes
20	VH	L	VH	2 or (H)	Yes
21	M	L	VH	1.66 or (H,-0.33)	Neg
22	M	L	VH	1.66 or (H,-0.33)	Neg

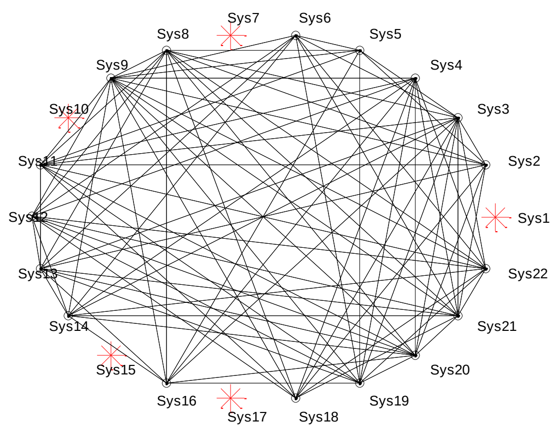


Figure 5.6. SoS Meta-Architecture for scenario 3

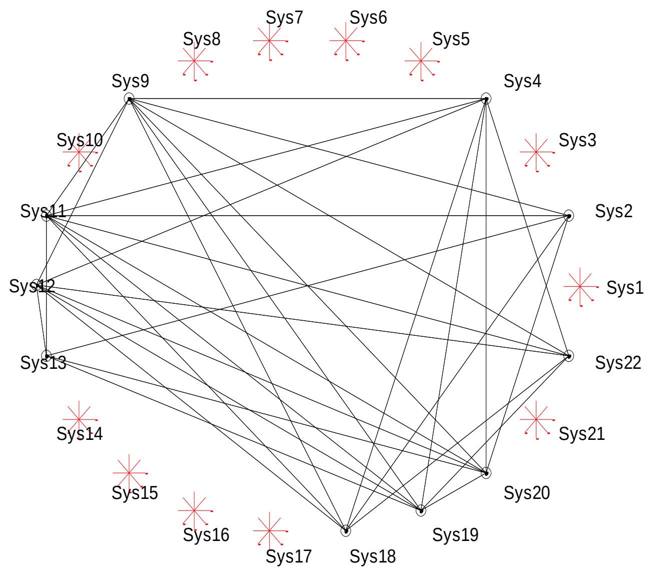


Figure 5.7. Negotiated-Architecture for scenario 3

Table 5.19. Domain Inputs for Scenario 4

SysNo	Type	Capability	I/FDevCos	OpsCost/I	Perf	DevTime
1	Cutter	2	0.1	0.05	12	1
2	Cutter	2	0.04	0.5	10	1
3	Helicopte	2	0.3	0.5	10	1
4	Helicopte	2	0.2	0.4	20	1
5	Aircraft	2	0.3	0.6	15	1
6	Aircraft	2	0.3	0.6	15	1
7	UAV	1	0.7	0.1	8	1
8	UAV	1	0.7	0.1	8	1
9	UAV	1	0.4	0.1	5	1
10	UAV	1	0.7	0.1	7	1
11	UAV	1	0.7	0.1	7	1
12	UAV	1	0.7	0.1	7	1
13	Fish Vesse	3	0.03	0.4	10	1
14	Fish Vesse	3	0.03	0.5	12	1
15	Fish Vesse	3	0.03	0.4	12	0
16	Civ Ship	3	0.05	2.5	8	0
17	Coord Ctr	4	0.05	0.2	7	0
18	Coord Ctr	4	0.05	0.5	5	1
19	Comm	5	0.09	0.1	1	0
20	Comm	5	0.09	0.1	1	0
21	Comm	5	0.09	0.1	2	1
22	Comm	5	0.09	0.1	2	1

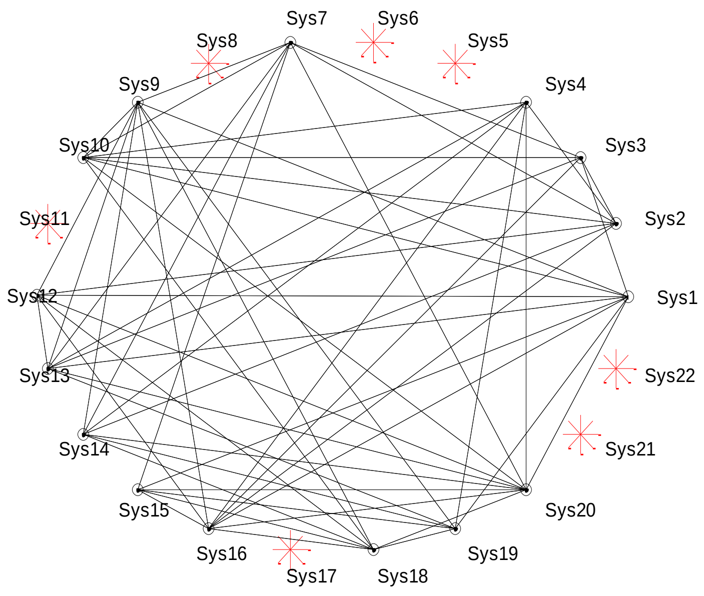


Figure 5.8. SoS Meta-Architecture for scenario 4

Table 5.20. Systems and capabilities in Scenario 4

Systems Selected in Meta-Architecture	**Capabilities Provided**	**Systems Selected in Negotiated Architecture**	**Capabilities Provided**
Systems 1, 2-Cutter	2	None	2
3,4-Helicopter	2	4-Helicopter	2
5,6-Aircraft	2	None	2
Systems 7, 9,10,12-UAV	1	Systems 7,9,12 UAV	1
Systems 13,14,15,16-Fish Vessel	3	Systems 14,15-Fish Vessel	3
Systems 18 –Coordination Control	4	Systems 18 – Coordination Control	4
Systems 19, 20-Communication	5	Systems 19, 20-Communication	5

Table 5.21. Decision by SoS manager for each system in meta-architecture

Sys No	Cooperativeness	Willingness of Collaboration	Preference Capability	Aggregated Value	Selection
1	M	L	L	0.33 or (L,0.33)	No
2	VH	L	L	1 or (M)	No
3	L	VH	H	1.66 or (H,-0.33)	Neg
4	M	VH	H	2 or (H)	Yes
7	H	H	VH	2.33 or (H,0.33)	Yes
8	H	H	M	1.66 or (H,-0.33)	Neg
9	VH	H	M	2 or (H)	Yes
10	H	H	M	1.66 or (H,-0.33)	Neg
12	VH	H	M	2 or (H)	Yes
13	L	M	H	1 or (M)	No
14	VH	M	H	2 or (H)	Yes
15	VH	M	H	2 or (H)	Yes
16	M	M	VH	1.66 or (H,-0.33)	Neg
18	L	VH	VH	2 or (H)	Yes
19	VH	L	VH	2 or (H)	Yes
20	VH	L	VH	2 or (H)	Yes

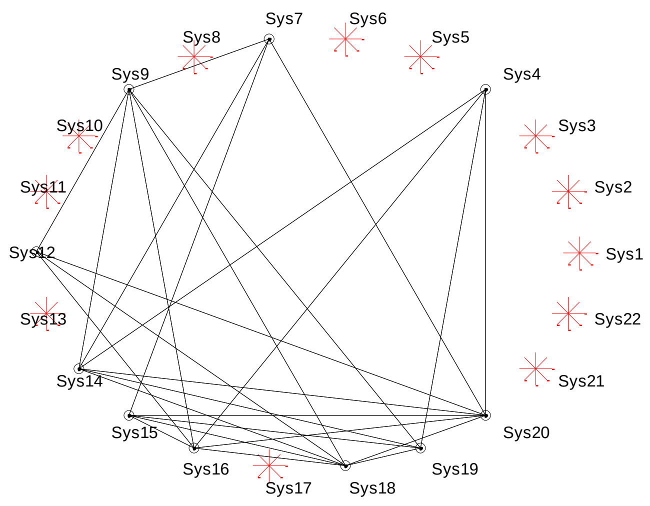


Figure 5.9. Negotiated-Architecture for scenario 4

The distribution of monetary benefits was changed at the start of a new wave when new systems were incorporated for selection in the domain specific inputs. This is because a system may spend funds on an interface that will not be ready until the next epoch, but they will get no performance increment from that interface until it is complete. Similarly a different overall ‘relative’ performance value was assigned to each system based on its key capability at the start of new wave.

In Scenario 5 some rules were created to define the trade-offs between the many objectives as stated to result in Figures 5.10. and Figures 5.11. The rules are:

- If (Performance is medium) and (Affordability is medium) and Net-Centricity is high) and (Robustness is low) then (SoS_Arch_Fitness is low)
- If (Performance is high) and (Affordability is medium) and Net-Centricity is medium) and (Robustness is medium) then (SoS_Arch_Fitness is medium)
- If (Performance is high) and (Affordability is medium) and Net-Centricity is high) and (Robustness is high) then (SoS_Arch_Fitness is high)
- If (Performance is medium) and (Affordability is medium) and Net-Centricity is medium) and (Robustness is medium) then (SoS_Arch_Fitness is medium)

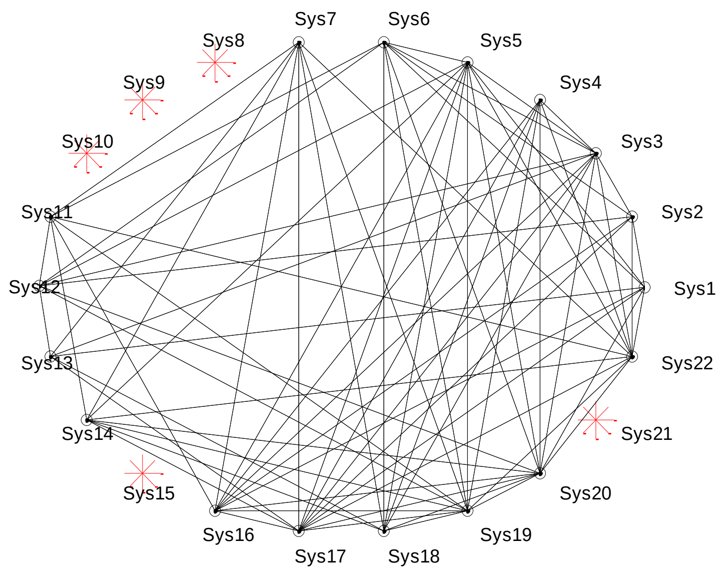


Figure 5.10. SoS Meta-Architecture for scenario 5

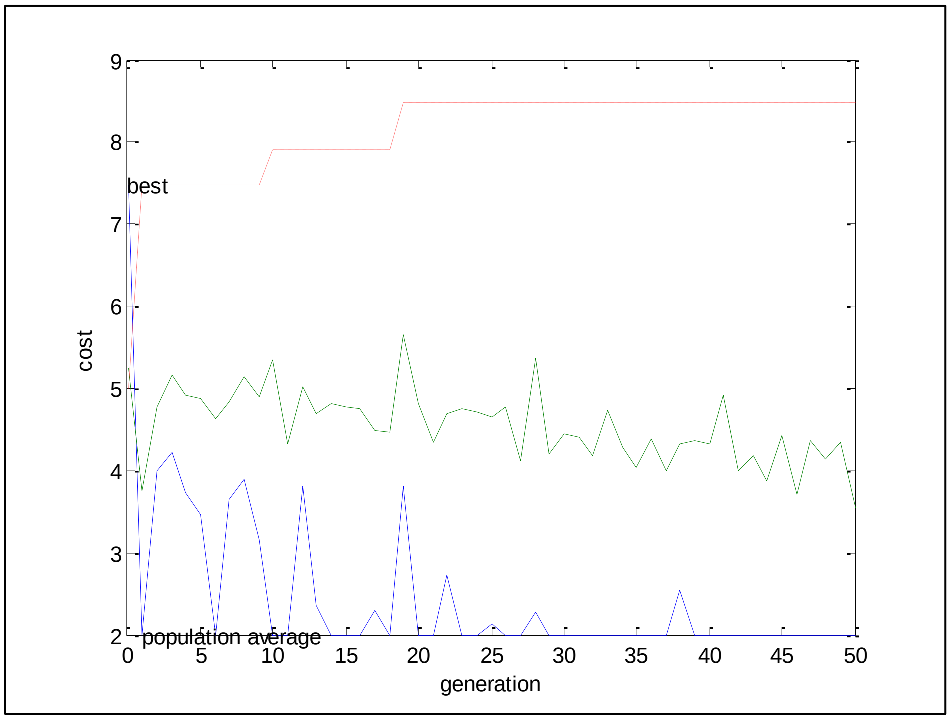


Figure 5.11. SoS architecture quality for 50 generations

Table 5.22. lists the systems and capabilities selected in scenario 5.

Table 5.22. Systems and capabilities in Scenario 5

Systems Selected in Meta-Architecture	**Capabilities Provided**	**Systems Selected in Negotiated Architecture**	**Capabilities Provided**
Systems 1, 2-Cutter	2	None	2
3,4-Helicopter	2	4-Helicopter	2
5,6-Aircraft	2	None	2
Systems 7, 11,12-UAV	1	Systems 7,9,12 UAV	1
Systems 13,14,16-Fish Vessel	3	Systems 14,15-Fish Vessel	3
Systems 17, 18 – Coordination Control	4	Systems 18 – Coordination Control	4
Systems 19, 20, 22- Communication	5	Systems 19, 20- Communication	5

The architecture quality and values of various key performance attributes on scale of 1 to 4, of various scenarios is listed below in Figures 5.12. and 5.13.

Meta-Architecture Scenario 1

Quality	**3.67**
Performance	3.66
Affordability	2.43
Net-Centricity	3
Robustness	3.74

Negotiated-Architecture Scenario 1

Quality	**1.45**
Performance	2.8
Affordability	3.76
Net-Centricity	2.55
Robustness	1.74

Meta-Architecture Scenario 2

Quality	**3.49**
Performance	3.21
Affordability	2.68
Net-Centricity	3.84
Robustness	3.24

Negotiated-Architecture Scenario 2

Quality	**2.38**
Performance	2.8
Affordability	2.58
Net-Centricity	2.87
Robustness	2.33

Meta-Architecture Scenario 3

Quality	**3.72**
Performance	3.59
Affordability	2.41
Net-Centricity	3.55
Robustness	3.36

Negotiated-Architecture Scenario 3

Quality	**1.37**
Performance	1.81
Affordability	3.82
Net-Centricity	2.95
Robustness	1.74

Figure 5.12. Architecture assessment results for Scenarios 1-3

Meta-Architecture Scenario 4

Quality	**3.19**
Performance	3.01
Affordability	2.49
Net-Centricity	3.64
Robustness	3.24

Negotiated-Architecture Scenario 4

Quality	**1.12**
Performance	1.18
Affordability	3.5
Net-Centricity	1.1
Robustness	1.13

Meta-Architecture Scenario 5

Quality	**3.21**
Performance	3.09
Affordability	3.08
Net-Centricity	3.8
Robustness	2.79

Figure 5.13. Architecture assessment results for Scenarios 4-5

Scenario 2 highlights preference of capability analysis such as the conditions when an individual system is more/less capable than the SoS expects. Scenario 3 highlights changes in willingness to collaborate and new set of domain inputs. This research proposes a different look at generating numerous underlying structures and dynamics of SoS. The inputs and rules in Scenario 5 are easily changed based on domain.

Incorporating these analyses helps the SoS decision maker to get an higher level overview of the situation. For further in-depth analysis in future, other techniques can be used to solve such problems. The next section highlights some methods that can enhance the existing model in future.

The model is a decision making aid for the SoS manager. It does not so much find the best solution to designing a SoS, as help the manager explore the influence of the various constraints on the shape of a reasonable solution. The models described can be used in conjunction with others to explore the SoS context and goals. This will help in developing SoS architectures including the full range of candidate systems and their interfaces. Our attempt has been to produce a holistic architecting methodology that is reconfigurable and has models that are adaptive to the environment.

FILA-SoS provides a capability to Acknowledged SoS manager to evaluate the impact of his sequence architecture selection and implementation decisions throughout the waves. It has been suggested by Maier (2005) to use Dynamic programming for formulating the SoS management problem. Neuro-Dynamic Programming (NDP) uses the concepts of neural networks for approximation of value functions, which are hard to calculate (Bertsekas & Tsitsiklis, 1995). Another approach for future work is to use approximate dynamic programming (Powell, 2207) which is based on post-decision state variables that avoid computing the expectation of uncertainties.

Sequential decision making by dynamic programming has been previously implemented (Dai Pra, Runggaldier, & Rudari, 1997). This structure wishes to provide SoS manager a sequence of architecture alternatives at different stages given the individual system capabilities and resource constraints.

Dynamic programming algorithms can be used to generate optimal sequence of decisions in enhancing this capability of FILA-SoS. A mathematical model formulation is provided to illustrate the concept.

Classical dynamic programming recursively computes the Bellman equation which is the essence of dynamic programming as following:

$$V_t(S_t) = \max_{x_t \in X_t}(C_t(S_t, x_t) + \gamma E\{V_{t+1}(S_{t+1})|S_t\})$$

where S_t represents state variables, x_t represents decision variables, C_t means current contribution, γ is discount factor and $V_{t+1}(S_{t+1})$ means expected value of being in state S_{t+1}.

Bitran (1970) developed theory and algorithms for multiple-criteria linear programs with binary variables. The algorithms were based on enumerative schemes and solving some auxiliary multiple objective programs. Multiple criteria integer linear programs were studied by several authors. Klein and Hannan (1982) developed an algorithm for generating the complete efficient set of such problems. This is a sequential procedure in which one of the criterion functions is optimized subject to progressively more constrained feasible sets determined by the other criteria and previously found efficient solutions.

The problem can be defined in simpler terms as:

1. States are equal to number of waves W where $j, k \in \{1,2, \dots, W]$
2. Actions $\in \{Selected, not\ selected]$
3. Transitioning to a state $\boldsymbol{S^j} = [s_{\boldsymbol{1}}^{\boldsymbol{j}}, \dots, s_{\boldsymbol{i}}^{\boldsymbol{j}}, \dots, s_{\boldsymbol{N}}^{\boldsymbol{j}}]$ with an action $a \in A$ and receive a discounted award $\boldsymbol{r_j}$
4. Probability of transitioning of system s_i^j (system *i* in state j) to s_i^k (system *i* in state *k)* is $\boldsymbol{P_{jk}^i}$

The idea is depicted in the Figure 5.14. The cost function ca be thought of as architecture quality that can be expressed as linear combination of key performance parameters or can be solved as a multi-objective optimization problem. The quality can also be assessed through a fuzzy assessor.

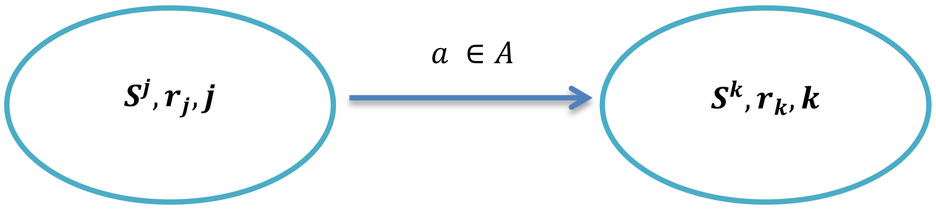


Figure 5.14. Transitioning in Dynamic Programming

First the SoS managers needs to start with a feasible solution state. Constraints are incorporated within actions taken to reach the new state. The expected discounted sum of future rewards when a system is starting in state *j* is given by $J^*(\boldsymbol{S^j}) = \boldsymbol{r_j} + \boldsymbol{\gamma}\{[\prod_{\boldsymbol{i=1}}^{\boldsymbol{N}} \boldsymbol{P_{j1}^i}].J^*(\boldsymbol{S^1}) + [\prod_{\boldsymbol{i=1}}^{\boldsymbol{N}} \boldsymbol{P_{j2}^i}].J^*(\boldsymbol{S^2}) + \dots + [\prod_{\boldsymbol{i=1}}^{\boldsymbol{N}} \boldsymbol{P_{jW}^i}].J^*(\boldsymbol{S^W})\}$.
Similarly, expected discounted sum of future rewards for a system in each possible staring state can be given as the matrix *J*, reward *as R* and probability *as P*.

$$J = \begin{pmatrix} J^*(\boldsymbol{S^1}) \\ J^*(\boldsymbol{S^2}) \\ . \\ J^*(\boldsymbol{S^W}) \end{pmatrix}$$

$$R = \begin{pmatrix} \boldsymbol{r_1} \\ \boldsymbol{r_2} \\ . \\ \boldsymbol{r_W} \end{pmatrix}$$

$$P = \begin{bmatrix} \prod_{\boldsymbol{i=1}}^{\boldsymbol{N}} \boldsymbol{P_{11}^i} & . & . \\ . & . & . \\ \prod_{\boldsymbol{i=1}}^{\boldsymbol{N}} \boldsymbol{P_{W1}^i} & . & \prod_{\boldsymbol{i=1}}^{\boldsymbol{N}} \boldsymbol{P_{WW}^i} \end{bmatrix}$$

If you have a lot of states let’s say a 100, then a 100X100 system of equations needs to be solved. This is computationally expensive hence neuro-dynamic programming can be used to solve such problems. This approach can further help the SoS manager in solving the problems at a lower level.

6. CONCLUSIONS AND FUTURE WORK

The goal of this research is to model the evolution of the architecture of an acknowledged Systems of Systems (SoS) that accounts for the ability and willingness of constituent systems to support the SoS capability development. The Wave Process Model provides a framework for modeling methodology, and this research provides different sets of modules to be integrated with the rest of them. The research is successfully able to achiev the objectives that are to develop a simulation for acknowledged SoS architecture selection and evolution, have a structured, repeatable approach for planning and modeling and study and evaluate the impact of individual system behavior on SoS capability and architecture evolution process. Results have been satisfactory and proved the model as a prototype.

In this dissertation research question "*What is the impact of different constituent system perspectives regarding participating in the SoS on the overall mission effectiveness of the SoS?"*is answered through the integrated model. This work helps in examining the impact of development approaches of different participating systems in a SoS to achieve the overarching capability. This approach involves meta-architecture generation and SoS negotiation models to implement our ideas. The meta-architecture generation technique helps in capturing the varied differences in the resources required by systems to prepare for participation. Similarly, the behavioral aspect of systems is tackled through an adaptive SoS negotiation strategy. The overall mission effectiveness is measured by effectively meeting the overarching objective

This thesis represents a first step towards addressing the tenacious problems in Acknowledged SoS such as cost estimates and cost overruns (Schwartz, 2010), which have overwhelmed the DoD. Future research on the ideas presented in this thesis could benefit the systems engineering community as demonstrated in this thesis.

This research has some limitation such as it has the data used for clustering has needs to be updated as time goes along. Besides the multiple waves depend on scenario for simulation and hence different domains may lead to different results. Other limitations include a more detailed way of defining the membership functions. This may affect the architecture quality results. Better ways to visualize this information may be helpful in the future.

When making decision on offers made by SoS stakeholders the goals of the individual decision-makers may differ on the alternatives based on attributes. This is due to the fact each individual processes the information differently to base their decision. Therefore, we need a group decision-making ability. The theory of intuitionistic fuzzy (Rodríguez, Martínez, Torra, Xu, & Herrera, 2014) sets further extend both concepts by allowing the assessment of the elements by two functions: μ for membership and υ for non-membership, which belong to the real unit interval [0, 1] and whose sum belongs to the same interval, as well.

Other metrics such as entropy can be added to evaluate the architectures quality (Cloutier, Verma, Bone & Sommer, 2009). The work done so far tries to investigate the impact of entropy on other attributes of systems architectures, the effect of low or high entropy on systems physical architecture and finally what steps can be adopted to improve the architecture quality through its entropy value (Bone et al., 2010).

Novel approaches also propose to assess the approach of joint programs that appear to cost more than disjoint programs (Dwyer & Szajnfarber, 2014). A Framework is proposed by the authors that can help the stakeholders reconfigure their policy and identify risks to develop approaches. These strategies will help maintain the cost-effectiveness (Dwyer et al., 2014).

Numerous systems have dissimilar goals, therefore integration and assimilation of information is needed to guide them to larger missions in the face of uncertainty and attacks. This research takes a step towards achieving that capability by introducing a new analysis framework that uses modeling tools to expose foreseeable SoS level impacts for decision makers early in the lifecycle, when such impacts can be manage less expensively and more solutions to possible problems can be put on the table.

BIBLIOGRAPHY

Ackoff, R. L. (1971). Towards a system of systems concepts. *Management science*, *17*(11), 661-671.

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.

Adra, S. F., & Fleming, P. J. (2011). Diversity management in evolutionary many-objective optimization. *Evolutionary Computation, IEEE Transactions on*, 15(2), 183-195.

Agarwal, S., & Ganguli, R. (2011). Refining automated modeling of operational data by identifying the most important input factors. *Mining Engineering*, *63*(12), 52-54.

Agarwal, S., & Gangull, R. (2011). Automating modeling of operational data to identify the most important factors. Paper presented at the *SME Annual Meeting and Exhibit and CMA 113th National Western Mining Conference 2011*, 142-143.

Agarwal,S., Wang, R., & Dagli, C., (2015) FILA-SoS, Executable Architectures using Cuckoo Search Optimization coupled with OPM and CPN-A module: A new Meta-Architecture Model for FILA-SoS, France, Complex Systems Design & Management (CSD&M) editor, Boulanger, Frédéric, Krob, Daniel, Morel, Gérard, Roussel, Jean-Claude, P 175-192 . Springer International Publishing.

Agarwal, S., Pape, L. E., & Dagli, C. H. (2014). A Hybrid Genetic Algorithm and Particle Swarm Optimization with Type-2 Fuzzy Sets for Generating Systems of Systems Architectures. *Procedia Computer Science, 36,* 57-64.

Agarwal, S., Saferpour, H. R., & Dagli, C. H. (2014). Adaptive Learning Model for Predicting Negotiation Behaviors through Hybrid K-means Clustering, Linear Vector Quantization and 2-Tuple Fuzzy Linguistic Model. *Procedia Computer Science, 36,* 285-292.

Agarwal, S., Pape, L. E., Kilicay-Ergin, N., & Dagli, C. H. (2014). Multi-agent Based Architecture for Acknowledged System of Systems. *Procedia Computer Science, 28,* 1-10.

Agarwal, S., Pape, L. E., Dagli, C. H., Ergin, N. K., Enke, D., Gosavi, A., ... & Gottapu, R. D. (2015). Flexible and Intelligent Learning Architectures for SoS (FILA-SoS): Architectural Evolution in Systems-of-Systems. *Procedia Computer Science, 44,* 76-85.

Ahn, J. H., Ryu, Y., & Baik, D. K. (2012). An Archietcture Description method for Acknowledged System of Systems based on Federated Architeture. *Advanced Science and Technology Letters*, *5*.

An, B., Lesser, V., & Sim, K. M. (2011). Strategic agents for multi-resource negotiation. *Autonomous Agents and Multi-Agent Systems, 23(1),* 114-153.

Analysis of Alternatives (AoA) Handbook A Practical Guide to Analyses of Alternatives July 2008 Office of Aerospace Studies Air Force Materiel Command (AFMC) OAS/A9

Arnold, A., Boyer, B., & Legay, A. (2013). Contracts and Behavioral Patterns for SoS: The EU IP DANSE approach. arXiv preprint arXiv:1311.3631.

Axelrod, R. M. (2006). *The evolution of cooperation.* Basic books, Inc., Publishers, New York, USA.

Baarslag, T. (2014). *What to Bid and When to Stop* (Doctoral dissertation, TU Delft, Delft University of Technology).

Baarslag, T., Hindriks, K., & Jonker, C. (2011). Towards a quantitative concession-based classification method of negotiation strategies. In *Agents in Principle, Agents in Practice* (pp. 143-158). Springer Berlin Heidelberg.

Baarslag, T., Hindriks, K., & Jonker, C. (2013). Acceptance conditions in automated negotiation. In *Complex Automated Negotiations: Theories, Models, and Software Competitions* (pp. 95-111). Springer Berlin Heidelberg.

Baarslag, T., Hindriks, K., & Jonker, C. (2014). Effective acceptance conditions in real-time automated negotiation. *Decision Support Systems, 60,* 68-77.

Babar, M. A., Zhu, L., & Jeffery, R. (2004). A framework for classifying and comparing software architecture evaluation methods. In *Software Engineering Conference, 2004. Proceedings. 2004 Australian* (pp. 309-318). IEEE.

Back, T., & Schwefel, H. P. (1996, May). Evolutionary computation: An overview. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on* (pp. 20-29). IEEE.

Bader, J., & Zitzler, E. (2011). HypE: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary Computation, 19(1),* 45-76.

Bahrammirzaee, A., Chohra, A., & Madani, K. (2013). An adaptive approach for decision making tactics in automated negotiation. *Applied intelligence, 39(3),* 583-606.

Baldwin, C. Y., & Clark, K. B. (2006). Modularity in the design of complex engineering systems (pp. 175-205). Springer Berlin Heidelberg.

Batista, L. S., Campelo, F., Guimaraes, F. G., & Ramírez, J. A. (2011, June). A comparison of dominance criteria in many-objective optimization problems. In *Evolutionary Computation (CEC), 2011 IEEE Congress on* (pp. 2359-2366). IEEE.

Bergey, J. K., Blanchette Jr, S., Clements, P. C., Gagliardi, M. J., Klein, J., Wojcik, R., & Wood, W. (2009). *US Army Workshop on Exploring Enterprise, System of Systems, System, and Software Architectures.*

Bertsekas, D. P., & Tsitsiklis, J. N. (1995, December). Neuro-dynamic programming: an overview. In Decision and Control, 1995., Proceedings of the 34th IEEE Conference on (Vol. 1, pp. 560-564). IEEE.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *The Journal of Machine Learning Research, 13(1),* 281-305.

Bertsekas, D. P. (1999). Nonlinear programming (2nd ed.). *Athena Scientific.* ISBN 1-886529-00-0.

Beyer, H. G., & Schwefel, H. P. (2002). Evolution strategies–A comprehensive introduction. *Natural computing, 1(1),* 3-52.

Bi, X., & Xiao, J. (2012). Classification-based self-adaptive differential evolution and its application in multi-lateral multi-issue negotiation. *Frontiers of Computer Science, 6(4),* 442-461.

Bigham, J., & Du, L. (2003, July). Cooperative negotiation in a multi-agent system for real-time load balancing of a mobile cellular network. In Proceedings of the second international joint conference on Autonomous agents and multiagent systems (pp. 568-575). ACM.

Binmore, K., & Vulkan, N. (1999). Applying game theory to automated negotiation. Netnomics, 1(1), 1-9.

BKCASE Editorial Board. 2014. *The Guide to the Systems Engineering Body of Knowledge (SEBoK)*, v. 1.3. R.D. Adcock (EIC). Hoboken, NJ: The Trustees of the Stevens Institute of Technology. Accessed DATE. www.sebokwiki.org.

Bodner, D A; Medvidovic, N ; Lane, J ; Boehm, B; Kessler, W ; Rouse, W ; Edwards, G ; Kirkland, K ; Krka, I ; Podar, A ; Popescu, D. 2011. Requirements Management for Net-Centric Enterprises. Phase 2. Accession Number : ADA589806

Bondy, J.A.; Murty, U.S.R. (2008), *Graph Theory*, Springer, ISBN 978-1-84628-969-9

Bone, M. A., Cloutier, R., Korfiatis, P., & Carrigy, A. (2010, June). System architecture: Complexities role in architecture entropy. In System of Systems Engineering (SoSE), 2010 5th International Conference on (pp. 1-6). IEEE.

Bounova, G., de Weck, O.L. (2012). Overview of metrics and their correlation patterns for multiple-metric topology analysis on heterogeneous graph ensembles, Phys. Rev. E 85, 016117.

Newman, M. E. (2006). Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, *103(23),* 8577-8582.

Bouwens, c. L. (2013). *Systems geometry: a methodology for analyzing emergent system of systems behaviors* (Doctoral dissertation, University of Central Florida Orlando, Florida).

Breivik, Ø., Allen, A. A., Maisondieu, C., & Olagnon, M. (2013). Advances in search and rescue at sea. *Ocean Dynamics*, *63*(1), 83-88.

Brucker, P., Drexl, A., Möhring, R., Neumann, K., & Pesch, E. (1999). Resource-constrained project scheduling: Notation, classification, models, and methods. *European journal of operational research*, *112*(1), 3-41.

Buhmann, Martin D. (2003*), Radial Basis Functions: Theory and Implementations,* Cambridge University Press, ISBN 978-0-521-63338-3.

Bustince, H., Herrera, F., & Montero, J. (Eds.). (2008). *Fuzzy sets and their extensions: Representation, aggregation and models.* Springer-Verlag Berlin Heidelberg.

Buttner, R. (2006, December). A classification structure for automated negotiations. In *Web Intelligence and Intelligent Agent Technology Workshops, 2006. WI-IAT 2006 Workshops. 2006 IEEE/WIC/ACM International Conference on* (pp. 523-530). IEEE.

Callanan, B., & Weiler, D. (2008, May).War Budgeting Strategies: Case Studies of The Gulf War and The Iraq War.

Cara, Ana Belén, Christian Wagner, Hani Hagras, Héctor Pomares, and Ignacio Rojas. "Multi-objective Optimization and Comparison of Non-Singleton Type-1 and Singleton Interval Type-2 Fuzzy Logic Systems." (2013): 1-1.

Carbonneau, R., Kersten, G. E., & Vahidov, R. (2008). Predicting opponent's moves in electronic negotiations using neural networks. *Expert Systems with Applications*, *34*(2), 1266-1273.

Carlsson, C., & Fullér, R. (1996). Fuzzy multiple criteria decision making: Recent developments. *Fuzzy sets and systems, 78(2),* 139-153.

Cavallo, A., & Ireland, V. (2014). Preparing for complex interdependent risks: A system of systems approach to building disaster resilience. *International Journal of Disaster Risk Reduction*, *9*, 181-193.

Celino, I., & Kotoulas, S. (2013). Smart Cities [Guest editors' introduction].*IEEE Internet Computing*, *17*(6), 8-11.

Celino, I., Contessa, S., Corubolo, M., Dell'Aglio, D., Della Valle, E., Fumeo, S., & Krüger, T. (2012). Linking smart cities datasets with human computation–the case of urbanmatch. In *The Semantic Web–ISWC 2012* (pp. 34-49). Springer Berlin Heidelberg.

Chan, H. K. (2011). Supply chain systems—Recent trend in research and applications. Systems Journal, IEEE, 5(1), 2-5.

Chandana, S., & Leung, H. (2010). A system of systems approach to disaster management. *Communications Magazine, IEEE*, *48(3)*, 138-145.

Chattopadhyay, D., Ross, A. M., & Rhodes, D. H. (2008, April). A framework for tradespace exploration of systems of systems. In *6th Conference on Systems Engineering Research*, Los Angeles, CA.

Chen, S., Hong, X., Luk, B. L., & Harris, C. J. (2009). Construction of tunable radial basis function networks using orthogonal forward selection. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, *39*(2), 457-466.

Chen, S., & Weiss, G. (2013). An efficient automated negotiation strategy for complex environments. *Engineering Applications of Artificial Intelligence,26(10)*, 2613-2623.

Chiong, R. (Ed.). (2009*). Nature-inspired algorithms for optimization* (Vol. 193). Springer.

Chung, F. L., & Lee, T. (1996). On fuzzy associative memory with multiple-rule storage capacity. Fuzzy Systems, *IEEE Transactions on*, *4(3)*, 375-384.

Cloutier, R., Verma, D., Bone, M., & Sommer, K. (2009, July). 4.1. 3 System Architecture Entropy. In *INCOSE International Symposium* (Vol. 19, No. 1, pp. 622-636).

Coehoorn, R. M., & Jennings, N. R. (2004, March). Learning on opponent's preferences to make effective multi-issue negotiation trade-offs. In Proceedings of the 6th international conference on Electronic commerce (pp. 59-68). ACM.

Coello, C. A. C. (1999). List of references on evolutionary multiobjective optimization. Laboratorio Nacional de Informática Avanzada, México.

Coello, C. A. C. (1999). An updated survey of evolutionary multiobjective optimization techniques: State of the art and future trends. In Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on (Vol. 1). IEEE.

Coello Coello, C. A. (2002). Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art.*Computer methods in applied mechanics and engineering*, *191*(11), 1245-1287.

Coleman, J. W., Malmos, A. K., Larsen, P. G., Peleska, J., & Hains, R. (2012). COMPASS tool vision for a system of systems Collaborative Development Environment. In *International Conference on System of Systems Engineering*,.2012 7th International Conference on , vol., no., pp.451,456, 16-19

CPS20: CPS 20 years from now - visions and challenges, a CPSWeek workshop, in Berlin, Germany, April 14 2014, http://www.cyphers.eu/.

Creel, R., & Ellison, B. (2008). System-of-systems influences on acquisition strategy development. *Software Engineering Institute*.

Dagli, C. H., & Kilicay-Ergin, N. (2008). System of Systems Architecting. *System of Systems Engineering*, 77-100.

Dagli, C. H., Singh, A., Dauby, J. P., & Wang, R. (2009, December). Smart systems architecting: computational intelligence applied to trade space exploration and system design. In Systems Research Forum (Vol. 3, No. 02, pp. 101-119). World Scientific Publishing Company.

Dagli, C.H. (2013). *An Advanced Computational Approach to System of Systems Analysis & Architecting Using Agent-Based Behavioral Model* (Systems Engineering Research Center Final Technical Report SERC-2013-TR-021-3).

Dagli, C., Ergin, N., Enke, D., Gosavi, A., Qin, R., Colombi, J., Rebovich, G., & Pape, L. (2013). An Advanced Computational Approach to System of Systems Analysis & Architecting Using Agent-Based Behavioral Model (No. SERC-2013-TR-021-2). Missouri University of Science and Technology Rolla.

Dahmann, J,. Lane, J., G. Rebovich, and K. Baldwin. "A model of systems engineering in a system of systems context." In *Proceedings of the Conference on Systems Engineering Research, Los Angeles, CA, USA (April 2008)*. 2008.

Dahmann, J., Baldwin, K. J., & Rebovich Jr, G. (2009, April). Systems of Systems and Net-Centric Enterprise Systems. In *7th Annual Conference on Systems Engineering Research, Loughborough*.

Dahmann, J., Rebovich, G., Lowry, R., Lane, J., & Baldwin, K. (2011, April). An implementers' view of systems engineering for systems of systems. In *Systems Conference (SysCon), 2011 IEEE International* (pp. 212-217). IEEE.

Dahmann, J. S., & Baldwin, K. J. (2008, April). Understanding the current state of US defense systems of systems and the implications for systems engineering. In *Systems Conference, 2008 2nd Annual IEEE* (pp. 1-7). IEEE.

Dahmann, J., Rebovich, G., & Turner, G. (2014, March). An actionable framework for system of systems and mission area security engineering. In Systems Conference (SysCon), 2014 8th Annual IEEE (pp. 12-17). IEEE.

Dai Pra, P., Runggaldier, W. J., & Rudari, C. (1997). On dynamic programming for sequential decision problems under a general form of uncertainty.Mathematical methods of operations research, 45(1), 81-107.

Darabi, H. R., & Mansouri, M. (2013). The Role of Competition and Collaboration in Influencing the Level of Autonomy and Belonging in System of Systems.

Davendralingam, N., DeLaurentis, D., Fang, Z., Guariniello, C., Han, S. Y., Marais, K., Mour, A., & Uday, P. (2014). An Analytic Workbench Perspective to Evolution of System of Systems Architectures. *Procedia Computer Science,28*, 702-710.

Deb, K. (1999). Solving goal programming problems using multi-objective genetic algorithms. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on* (Vol. 1). IEEE.

Deb, K.; Jain, H., "An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints," *Evolutionary Computation, IEEE Transactions* on , vol.18, no.4, pp.577,601, Aug. 2014

DeLaurentis, D. A., Crossley, W. A., & Mane, M. (2011). Taxonomy to guide systems-of-systems decision-making in air transportation problems. Journal of Aircraft, 48(3), 760-770.

Deshmukh, P. K., & Gholap, Y. (2012, December). Dept. of Post Grad. Comput. Eng., JSPM'S Rajarshi Shahu Coll. of Eng., Pune, India. In *Hybrid Intelligent Systems (HIS), 2012 12th International Conference on* (pp. 52-56). IEEE.

De Weck, O. L., Neufville, R. D., & Chaize, M. (2004). Staged deployment of communications satellite constellations in low earth orbit. *Journal of Aerospace Computing, Information, and Communication*, *1*(3), 119-136.

Dodgson, J. S., Spackman, M., Pearman, A., & Phillips, L. D. (2009). Multi-criteria analysis: a manual. Department for Communities and Local Government: London.

Dombkins, D. H. Project Managed Change: the Application of Project Management Techniques to strategic change program. Centre for Corporate Change working paper no, 062 (1996). Australian Graduate School of Management, The University of New South Wales, 1996.

Dombkins, D. H. (2013).Realizing Complex Policy: Using a Systems-of-Systems Approach to Develop and Implement Policy. *Editor's Introduction, Volume II, Issue 5,* 22.

Dong, P., Han, Y., Guo, X., & Xie, F. (2015). A Systematic Review of Studies on Cyber Physical System Security. *International Journal of Security and Its Applications, 9(1),* 155-164.

Dwyer, M. M., & Szajnfarber, Z. (2014). A Framework to Assess the Impacts of Jointness. In *4th International Engineering Systems Symposium*.

Dwyer, M., Szajnfarber, Z., Cameron, B., Selva, D., & Crawley, E. (2014). The Cost of Jointness and How to Manage It. In *AIAA SPACE*.

Dyer, J. S. (2005). MAUT—multiattribute utility theory. In *Multiple criteria decision analysis: state of the art surveys* (pp. 265-292). Springer New York.

Engelbrecht, A. P. (2006). *Fundamentals of computational swarm intelligence*. John Wiley & Sons.

Ergin, N. K.,(2014), Improving Collaboration in Search and Rescue System of Systems, *Procedia Computer Science, Volume 36*, Pages 13-20.

Fang, Z., & DeLaurentis, D. (2014). Dynamic Planning of System of Systems Architecture Evolution. *Procedia Computer Science, 28,* 449-456.

Faratin, P., Sierra, C., & Jennings, N. R. (1998). Negotiation decision functions for autonomous agents. *Robotics and Autonomous Systems, 24(3),* 159-182.

Faratin, P., Sierra, C., & Jennings, N. R. (2002). Using similarity criteria to make issue trade-offs in automated negotiations. *Artificial Intelligence, 142(2),* 205-237

Farmani, R., & Wright, J. A. (2003). Self-adaptive fitness formulation for constrained optimization. *Evolutionary Computation, IEEE Transactions on,7*(5), 445-455.

Fatima, S. S., Wooldridge, M., & Jennings, N. R. (2002, July). Multi-issue negotiation under time constraints. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1* (pp. 143-150). ACM.

Fatima, S. S., Wooldridge, M., & Jennings, N. R. (2006). Multi-issue negotiation with deadlines. *Journal Artificial Intelligence Research.(JAIR), 27,* 381-417.

Fiedler, M. (1973). Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, *23*(2), 298-305.

Flynn, C., & Richardson, J. (2013). Joint Operational Access and the Global Response Force: Redefining Readiness. Military Review, 93(4), 38.

Folmer, E., van Gurp, J., & Bosch, J. (2003, May). Scenario-based Assessment of Software Architecture Usability. In ICSE Workshop on SE-HCI (pp. 61-68).

Fraser, R., & Hawkins, C. (2014). Building a System of Systems For Disaster Management Workshop: Joint Issues Statement.

Freeman, L. (1994). Displaying Hierarchical Clusters. INSNA Connections, 17(2), 46-52.

Fry, D. N., & DeLaurentis, D. A. (2011, June). Measuring net-centricity. InSystem of Systems Engineering (SoSE), 2011 6th International Conference on(pp. 264-269). IEEE.

Gagliardi, M., & Wood, B. (October 2013) Identifying Architectural Challenges in System of Systems NDIA Systems Engineering Conference.

Garg, R., & Singh, D. (2011). ε–Pareto Dominance Based Multi-objective Optimization to Workflow Grid Scheduling. In *Contemporary Computing (pp. 29-40). Springer Berlin Heidelberg.*

Gatti, N., & Amigoni, F. (2004, July). A cooperative negotiation protocol for physiological model combination. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2* (pp. 655-662). IEEE Computer Society.

Garza-Fabre, M., Pulido, G. T., & Coello, C. A. C. (2009). Ranking methods for many-objective optimization. In *MICAI 2009: Advances in Artificial Intelligence* (pp. 633-645). Springer Berlin Heidelberg.

Gorod, A., White E. B., Ireland, V., Gandhi, J. S., & Sauser, B., (Eds.). (2014). Case Studies in System of Systems, Enterprise Systems, and Complex Systems Engineering. CRC Press.

Grira, N., Crucianu, M., & Boujemaa, N. (2004). Unsupervised and semi-supervised clustering: a brief survey. A review of machine learning techniques for processing multimedia content, Report of the MUSCLE European Network of Excellence (FP6).

Guo, X., Wang, X., Wang, M., & Wang, Y. (2012, November). A new objective reduction algorithm for many-objective problems: employing mutual information and clustering algorithm. In *Computational Intelligence and Security (CIS), 2012 Eighth International Conference on* (pp. 11-16). IEEE.

Guo, Y. (2009). *An investigation of model-based techniques for automotive electronic system development* (Doctoral dissertation, University of Warwick).

Guttman, R. H., & Maes, P. (1998). Cooperative vs. competitive multi-agent negotiations in retail electronic commerce. In *Cooperative Information Agents II Learning, Mobility and Electronic Commerce for Information Discovery on the Internet* (pp. 135-147). Springer Berlin Heidelberg.

Hadka, D., & Reed, P. (2013). Borg: An auto-adaptive many-objective evolutionary computing framework. *Evolutionary computation, 21(2)*, 231-259.

Hadian, S., & Madani, K. (2015). A system of systems approach to energy sustainability assessment: Are all renewables really green?. *Ecological Indicators, 52*, 194-206.

Hagen, T. E. (2007). An Architectural Process for Achieving Robustness.

Hajela, P., & Lee, J. (1996). Constrained genetic search via schema adaptation: an immune network solution. *Structural optimization*, *12*(1), 11-15.

Han, S. Y., & DeLaurentis, D. (2013). Development Interdependency Modeling for System-of-Systems (SoS) using Bayesian Networks: SoS Management Strategy Planning. *Procedia Computer Science, 16,* 698-707.

Han, S. Y., Marais, K., & DeLaurentis, D. (2012, October). Evaluating system of systems resilience using interdependency analysis. In Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on (pp. 1251-1256). IEEE.

Hassan, R., Cohanim, B., De Weck, O., & Venter, G. (2005, April). A comparison of particle swarm optimization and the genetic algorithm. In *Proceedings of the 1st AIAA multidisciplinary design optimization specialist conference (pp. 18-21).*

Haykin, S. S.(2009). *Neural networks and learning machines* (Vol. 3). Upper Saddle River: Pearson Education.

He, Z., Yen, G. G., & Zhang, J. (2014). Fuzzy-Based Pareto Optimality for Many-Objective Evolutionary Algorithms. *Evolutionary Computation, IEEE Transactions on, 18(2)*, 269-285.

Herrera, F., & Martínez, L. (2000). A 2-tuple fuzzy linguistic representation model for computing with words. *Fuzzy Systems, IEEE Transactions on, 8(6),* 746-752.

Hilliard, R., Kurland, M. J., & Litvintchouk, S. D. (1997, April). MITRE’s Architecture Quality Assessment. In *1997 MITRE Software Engineering and Economics Conference* (pp. 2-3).

Hindriks, K., & Tykhonov, D. (2008, May). Opponent modelling in automated multi-issue negotiation using bayesian learning. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*-Volume 1 (pp. 331-338). International Foundation for Autonomous Agents and Multiagent Systems.

Hindriks, K., Jonker, C. M., & Tykhonov, D. (2009, September). The benefits of opponent models in negotiation. In *Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT'09. IEEE/WIC/ACM International Joint Conferences on* (Vol. 2, pp. 439-444). IET.

Hodges, C. 2014. COBWEB – Citizen Observatories Web: Ecology meets the crowd. In GIS and Remote Sensing: the End of Fieldwork? Remote sensing and its role in ecological assessment, Chartered Institute of Ecology and Environmental Management (CIEEM) Welsh Section Conference and AGM, Aberystwyth, 21st February 2014.

Holland, J. H.; (2006). "Studying Complex Adaptive Systems."*Journal of Systems Science and Complexity* **19** (1): 1-8.

Horn, J., Nafpliotis, N., & Goldberg, D. E. (1994, June). A niched Pareto genetic algorithm for multiobjective optimization. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on* (pp. 82-87). Ieee.

Hothorn, T., & Everitt, B. S. (2009). A handbook of statistical analyses using R. CRC Press.

Hu, C. F., Teng, C. J., & Li, S. Y. (2007). A fuzzy goal programming approach to multi-objective optimization problem with priorities. *European Journal of Operational Research, 176(3),* 1319-1333.

Huang, J., & Xie, W. (1998). Genetic algorithm with fuzzy fitness evaluation. *Journal of Electronics (China), 15(3),* 254-258.

Hüllermeier, E. (2005). Fuzzy methods in machine learning and data mining: Status and prospects. *Fuzzy Sets and Systems, 156(3),* 387-406.

IBM Smart Traffic 2010. http://www.ibm.com/smarterplanet/us/en/traffic_congestion/ideas. Accessed May 2014

Ilango, M. R., & Mohan, V. (2010). A survey of grid based clustering algorithms. *International Journal of Engineering Science and Technology, 2(8),* 3441-3446.

Ishibuchi, H., Tsukamoto, N., & Nojima, Y. (2008, June). Evolutionary many-objective optimization: A short review. In *IEEE Congress on Evolutionary Computation* (pp. 2419-2426).

Ito, T., Zhang, M., Robu, V., Fatima, S., & Matsuo, T. (Eds.). (2009). *Advances in agent-based complex automated negotiations* (Vol. 233). Springer.

Ito, T., Zhang, M., Robu, V., Fatima, S., & Matsuo, T. (2012). *New trends in agent-based complex automated negotiations*. Springer.

Jamakovic, A., & Uhlig, S. (2007, May). On the relationship between the algebraic connectivity and graph's robustness to node and link failures. In *Next Generation Internet Networks, 3rd EuroNGI Conference on* (pp. 96-102). IEEE.

Janishidi, M. (2008, December). System of Systems-Innovations for 21st Century. In *Industrial and Information Systems, 2008. ICIIS 2008. IEEE Region 10 and the Third international Conference on* (pp. 6-7). IEEE.

Jennings, N. R., Faratin, P., Lomuscio, A. R., Parsons, S., Wooldridge, M. J., & Sierra, C. (2001). Automated negotiation: prospects, methods and challenges. *Group Decision and Negotiation, 10(2),* 199-215.

Johnson, S. C. (1967). Hierarchical clustering schemes. *Psychometrika*, 32(3), 241-254.

Jonker, C. M., Robu, V., & Treur, J. (2007). An agent architecture for multi-attribute negotiation using incomplete preference information. *Autonomous Agents and Multi-Agent Systems, 15(2),* 221-252.

Jordan, P. R., Kiekintveld, C., & Wellman, M. P. (2007, May). Empirical game-theoretic analysis of the TAC supply chain game. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems* (p. 193). ACM.

Kaplan, J. M. (2006). A new conceptual framework for net-centric, enterprise-wide, system-of-systems engineering. National Defense University Washington Dc Center for Technology and National Security Policy.

Kaufman, L., & Rousseeuw, P. J. (1990). Partitioning around medoids (program pam). Finding groups in data: an introduction to cluster analysis, 68-125.

Kaufman, L., & Rousseeuw, P. J. (2009). Finding groups in data: an introduction to cluster analysis (Vol. 344). John Wiley & Sons.

Kazman, R., Abowd, G., Bass, L., & Clements, P. (1996). Scenario-based analysis of software architecture. *Software, IEEE, 13(6),* 47-55.

Kazman, R., Klein, M., Barbacci, M., Longstaff, T., Lipson, H., & Carriere, J. (1998, August). The architecture tradeoff analysis method. In *Engineering of Complex Computer Systems, 1998. ICECCS'98. Proceedings. Fourth IEEE International Conference on* (pp. 68-78). IEEE.

Kennedy, J., & Eberhart, R. C. (1997, October). A discrete binary version of the particle swarm algorithm. In *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on* (Vol. 5, pp. 4104-4108). IEEE.

Ketchen, D. J., & Shook, C. L. (1996). The application of cluster analysis in strategic management research: an analysis and critique. *Strategic management journal, 17(6),* 441-458.

Kilgour, D. M., Chen, Y., & Hipel, K. W. (2010). Multiple criteria approaches to group decision and negotiation. *In Trends in Multiple Criteria Decision Analysis* (pp. 317-338). Springer US.

Kilicay-Ergin, N., & Dagli, C. (2008, April). Executable modeling for system of systems architecting: An artificial life framework. In *Systems Conference, 2008 2nd Annual IEEE* (pp. 1-5). IEEE.

Kilicay-Ergin, N. (2014). Improving Collaboration in Search and Rescue System of Systems. *Procedia Computer Science*, 36, 13-20.

Klein, D., & Hannan, E. (1982). An algorithm for the multiple objective integer linear programming problem. European Journal of Operational Research, 9(4), 378-385.

Kohonen, T. (1997). Learning vector quantization. In Self-Organizing Maps (pp. 203-217). *Springer Berlin Heidelberg*.

Konak, A., Coit, D. W., & Smith, A. E. (2006). Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering & System Safety, 91*(9), 992-1007.

Konur, D., & Dagli, C. (2014). Military system of systems architecting with individual system contracts. *Optimization Letters*, 1-19.

Köppen, M., Vicente-Garcia, R., & Nickolay, B. (2005, January). Fuzzy-pareto-dominance and its application in evolutionary multi-objective optimization. *In Evolutionary Multi-Criterion Optimization* (pp. 399-412). Springer Berlin Heidelberg.

Kosko, B. (1992). Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence/Book and Disk (Vol. 1). Prentice hall.

Kotov, V. (1997). "System of Systems as Communicating Structures." Hewlett Packard Computer Systems Laboratory, 1-14.

Kowalczyk, R., & Bui, V. (2000). On Fuzzy e-Negotiation Agents: Autonomous negotiation with incomplete and imprecise information. In *Database and Expert Systems Applications, 2000. Proceedings.* 11th International Workshop on (pp. 1034-1038). IEEE.

Kraus, S. (2001). Automated negotiation and decision making in multiagent environments. In Multi-agent systems and applications (pp. 150-172). Springer Berlin Heidelberg.

Krothapalli, N. K. C., & Deshmukh, A. V. (1999). Design of negotiation protocols for multi-agent manufacturing systems. *International Journal of Production Research, 37(7),* 1601-1624.

Langfelder, P., Zhang, B., & Horvath, S. (2008). Defining clusters from a hierarchical cluster tree: the Dynamic Tree Cut package for R. *Bioinformatics, 24(5),* 719-720.

Langton, C. G. (1990). Computation at the edge of chaos: phase transitions and emergent computation. Physica D: Nonlinear Phenomena, 42(1), 12-37.

Laudy, C., Petersson, H., & Sandkuhl, K. (2010, July). Architecture of knowledge fusion within an Integrated Mobile Security Kit. In *Information Fusion (FUSION), 2010 13th Conference* on (pp. 1-8). IEEE.

Liang, Q., & Mendel, J. M. (2000). Interval type-2 fuzzy logic systems: theory and design. *Fuzzy Systems, IEEE Transactions on, 8(5),* 535-550.

Lin, R., Kraus, S., Wilkenfeld, J., & Barry, J. (2006). An automated agent for bilateral negotiation with bounded rational agents with incomplete information. Frontiers in Artificial Intelligence and Applications, 141, 270.

Liu, S. (2011). Employing system of systems engineering in China's emergency management. *Systems Journal, IEEE, 5(2),* 298-308.

Liu, J. Q., Nishimura, H., & Umehara, H. (2012, March). On applying the method of "system of systems" in robustness analysis and autonomous control of dynamics-aware internet architecture. In *Systems Conference (SysCon), 2012 IEEE International* (pp. 1-6). IEEE.

Local4Global: "SYSTEM-OF-SYSTEMS THAT ACT LOCALLY FOR OPTIMIZING GLOBALLY", Project Number: 61153, Project Start Date: 01/10/2013. http://local4global-fp7.eu/.

Luo, X., Jennings, N. R., Shadbolt, N., Leung, H. F., & Lee, J. H. M. (2003). A fuzzy constraint based model for bilateral, multi-issue negotiations in semi-competitive environments. *Artificial Intelligence, 148(1),* 53-102.

Luzeaux, D., System-of-Systems (and Large-Scale Complex Systems) Engineering, presentation at CSDM conference, 2013. [2] Maier, M. W. (1998). *Architecting principles for systems-of-systems. Systems Engineering, 1(4),* 267-284.

MacQueen, J. (1967, June). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* (Vol. 1, No. 14, pp. 281-297).

Maia, P., Cavalcante, E., Gomes, P., Batista, T., Delicato, F. C., & Pires, P. F. (2014, August). On the Development of Systems-of-Systems based on the Internet of Things: A Systematic Mapping. In *Proceedings of the 2014 European Conference on Software Architecture Workshops* (p. 23). ACM.

Maier, M. W. (1998). Architecting principles for systems-of-systems. *Systems Engineering, 1(4),* 267-284.

Malan, R., & Bredemeyer, D. (2001). Architecture Resources. Defining Non-Functional Requirements.

Mamdani, E. H. (1977). Application of fuzzy logic to approximate reasoning using linguistic synthesis, *IEEE Transactions on Computers 26(12)*: 1182–1191.

Marler, R. T., & Arora, J. S. (2010). The weighted sum method for multi-objective optimization: new insights. *Structural and multidisciplinary optimization, 41(6),* 853-862.

Marsa-Maestre, I., Lopez-Carmona, M. A., Klein, M., Ito, T., & Fujita, K. (2012). Addressing Utility Space Complexity In Negotiations Involving Highly Uncorrelated, Constraint-Based Utility Spaces. *Computational Intelligence*.

Marsa-Maestre, I., Lopez-Carmona, M. A., Ito, T., Zhang, M., Bai, Q., & Fujita, K. (2014). *Novel Insights in Agent-based Complex Automated Negotiation.* Imprint: Springer.

Martí, J., Ventura, C., Hollman, J., Srivastava, K., & Juarez, H. (2015). I2Sim modelling and simulation framework for scenario development, training, and real-time decision support of multiple interdependent critical infrastructures during large emergencies. In *NATO (OTAN) MSG-060 Symposium on" How is Modelling and Simulation Meeting the Defence Challenges out to*.

Mashor, M. Y. (2000). Hybrid training algorithm for RBF network. *International Journal of the computer, the Internet and Management, 8(2),* 50-65.

Matos, N., Sierra, C., & Jennings, N. R. (1998, July). Determining successful negotiation strategies: An evolutionary approach. In Multi Agent Systems, 1998. *Proceedings. International Conference* on (pp. 182-189). IEEE.

Mendel, J. M., & John, R. B. (2002). Type-2 fuzzy sets made simple. *Fuzzy Systems, IEEE Transactions on, 10(2),* 117-127.

Mendel, J., & Wu, D. (2010). Perceptual computing: aiding people in making subjective judgments (Vol. 13). John Wiley & Sons.

Merson, P. (2002) Architecture Assessment. Process Documentation – version 1.0.

Mezura-Montes, E., & Coello Coello, C. A. (2011). Constraint-handling in nature-inspired numerical optimization: past, present and future. *Swarm and Evolutionary Computation*, *1*(4), 173-194.

Michalewicz, Z., & Janikow, C. Z. (1991, July). Handling Constraints in Genetic Algorithms. In *ICGA* (pp. 151-157).

Miller, M. Z., Pogaru, S. S., & Mavris, D. N. (2013). Smart Grid: Constructing a System of Systems Model Using Both Qualitative and Quantitative Assessments. In *Complex Systems Design & Management* (pp. 177-192). Springer Berlin Heidelberg.

Moon, T. K. (1996). The expectation-maximization algorithm. *Signal processing magazine, IEEE, 13(6),* 47-60.

Montecchi, L., Lollini, P., & Bondavalli, A. (2014, May). A DSL-Supported Workflow for the Automated Assembly of Large Stochastic Models. In *Dependable Computing Conference (EDCC),* 2014 Tenth European (pp. 82-93). IEEE.

Nahavandi, S., Creighton, D., Le, V. T., Johnstone, M., & Zhang, J. (2015). Future Integrated Factories: A System of Systems Engineering Perspective. In *Integrated Systems: Innovations and Applications* (pp. 147-161). Springer International Publishing.

Ncube, C., Lim, S. L., & Dogan, H. (2013, July). Identifying top challenges for international research on requirements engineering for systems of systems engineering. In *Requirements Engineering Conference (RE),* 2013 21st IEEE International (pp. 342-344). IEEE.

Nejad, H. T. N., Sugimura, N., Iwamura, K., & Tanimizu, Y. (2008). Agent-based dynamic process planning and scheduling in flexible manufacturing system. In *Manufacturing Systems and Technologies for the New Frontier* (pp. 269-274). Springer London.

Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *The computer journal, 7(4),* 308-313.

Nguyen, T. D., & Jennings, N. R. (2006). Managing commitments in multiple concurrent negotiations. *Electronic Commerce Research and Applications, 4(4),* 362-376.

Obal, L., & Lin, F. (2015). A Framework for Healthcare Information Systems: Exploring a Large System of Systems using System Dynamics. *Communications of the IIMA*, *5*(3), 4.

Pal, N. R., & Bezdek, J. C. (1995). On cluster validity for the fuzzy c-means model. Fuzzy Systems, *IEEE Transactions on, 3(3),* 370-379.

Pape, L., Agarwal, S., Giammarco, K., & Dagli, C. (2014). Fuzzy Optimization of Acknowledged System of Systems Meta-architectures for Agent based Modeling of Development. *Procedia Computer Science*, *28*, 404-411.

Pape, L., & Dagli, C. (2013). Assessing robustness in systems of systems meta-architectures. *Procedia Computer Science*, *20*, 262-269.

Pape, L., Giammarco, K., Colombi, J., Dagli, C., Kilicay-Ergin, N., & Rebovich, G. (2013). A fuzzy evaluation method for system of systems meta-architectures. *Procedia Computer Science*, *16*, 245-254.

Paulen, R., & Engell, S. (February 2014). DYMASOS – Dynamic Management of Physically Coupled Systems of Systems, published on ERCIM News 97, April 2014, Special theme: Cyber-Physical Systems, February 25, 2014.

Powell, W. B. (2007). Approximate Dynamic Programming: Solving the curses of dimensionality (Vol. 703). John Wiley & Sons.

Pyster, A., Olwell, D., Squires, A., Hutchison, N., Enck, S., & Anthony, J. (2014). A Guide to the Systems Engineering Body of Knowledge (SEBoK).*Version 1.3. Hoboken, NJ (US): Stevens Institute of Technology*. http://sebokwiki.org/w/downloads/SEBoKv1.3_full.pdf

Qi, Y., Ma, X., Liu, F., Jiao, L., Sun, J., & Wu, J. (2014). Moea/d with adaptive weight adjustment. *Evolutionary computation, 22(2)*, 231-264.

Rahwan, I., Kowalczyk, R., & Pham, H. H. (2002, January). Intelligent agents for automated one-to-many e-commerce negotiation. In *Australian Computer Science Communications* (Vol. 24, No. 1, pp. 197-204). Australian Computer Society, Inc..

Rauschecker, U., Ford, S. J., & Athanssopoulou, N. (2014). Developing a Vision for Multi-site Manufacturing System of Systems. In *Enabling Manufacturing Competitiveness and Economic Sustainability* (pp. 79-84). Springer International Publishing.

Reniers M. A., & Engell, S. 2014A European Roadmap on Cyber-Physical Systems of Systems. ERCIM News 2014(97).

Rhodes, D. H., Ross, A. M., & Nightingale, D. J. (2009, March). Architecting the system of systems enterprise: Enabling constructs and methods from the field of engineering systems. In *Systems Conference, 2009 3rd Annual IEEE*(pp. 190-195). IEEE.

Rodríguez, R. M., Martínez, L., Torra, V., Xu, Z. S., & Herrera, F. (2014). Hesitant fuzzy sets: State of the art and future directions. International Journal of Intelligent Systems, 29(6), 495-524.

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

Sage, A. P., & Cuppan, C. D. (2001). On the systems engineering and management of systems of systems and federations of systems. *Information, Knowledge, Systems Management*, *2*(4), 325-345.

Salvador, S., & Chan, P. (2004, November). Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms. In *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on* (pp. 576-584). IEEE.

Samad, T., & Parisini, T. (2011). Systems of systems. *The Impact of Control Technology*, 175-183.

Sato, A., & Yamada, K. (1996). Generalized learning vector quantization. *Advances in neural information processing systems*, 423-429.

Sato, A., & Yamada, K. (1998). An analysis of convergence in generalized LVQ. In *ICANN 98* (pp. 171-176). Springer London.

Saxena, D. K., Duro, J. A., Tiwari, A., Deb, K., & Zhang, Q. (2013). Objective reduction in many-objective optimization: Linear and nonlinear algorithms. *Evolutionary Computation, IEEE Transactions on, 17(1)*, 77-99.

Schutze, O., Lara, A., & Coello Coello, C. A. (2011). On the influence of the number of objectives on the hardness of a multiobjective optimization problem. *Evolutionary Computation, IEEE Transactions on, 15(4),* 444-455.

Schwartz, M. (2010, April). Defense acquisitions: How DoD acquires weapon systems and recent efforts to reform the process. Library of Congress Washington Dc Congressional Research Service.

Schwenker, F., Kestler, H. A., & Palm, G. (2001). Three learning phases for radial-basis-function networks. *Neural networks*, *14*(4), 439-458.

Schwind, N., Okimoto, T., Inoue, K., Chan, H., Ribeiro, T., Minami, K., & Maruyama, H. (2013, May). Systems resilience: a challenge problem for dynamic constraint-based agent systems. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems (pp. 785-788).* International Foundation for Autonomous Agents and Multiagent Systems.

Siebert, J. (2010). Aggregate Utility Factor Model: A Concept for Modeling Pair-wise Dependent Attributes in Multiattribute Utility Theory.

Siemieniuch, C., Sinclair, M., Lim, S. L., Henson, M. S., Jamshidi, M., & DeLaurentis, D. (2013). Project Title *Trans-Atlantic Research and Education Agenda in Systems of Systems (T-AREA-SoS).*

Sierra, C., Faratin, P., & Jennings, N. R. (1999). A service-oriented negotiation model between autonomous agents. In *Collaboration between Human and Artificial Societies* (pp. 201-219). Springer Berlin Heidelberg.

Simon, H. A. (1991). *The architecture of complexity* (pp. 457-476). Springer US.

Singh, A. (2011). Architecture value mapping: using fuzzy cognitive maps as a reasoning mechanism for multi-criteria conceptual design evaluation.

Sing, J. K., Basu, D. K., Nasipuri, M., & Kundu, M. (2003, October). Improved k-means algorithm in the design of RBF neural networks. In TENCON 2003. *Conference on Convergent Technologies for the Asia-Pacific Region* (Vol. 2, pp. 841-845). IEEE.

Singh, H. K., Isaacs, A., & Ray, T. (2011). A Pareto corner search evolutionary algorithm and dimensionality reduction in many-objective optimization problems. *Evolutionary Computation, IEEE Transactions on, 15(4),* 539-556.

Sivaraj, R., & Ravichandran, T. (2011). A review of selection methods in genetic algorithm. *International journal of engineering science and technology, 3(5),* 3792-3797.

Somervuo, P., & Kohonen, T. (1999). Self-organizing maps and learning vector quantization for feature sequences. *Neural Processing Letters, 10(2),* 151-159.

Song, Q., & Chissom, B. S. (1993). Fuzzy time series and its models. *Fuzzy sets and systems, 54(3),* 269-277.

Squires, A., Olwell, D., Roedler, G., & Ekstrom, J. J. (2012, July). Gaps in the Body of Knowledge of Systems Engineering. In *INCOSE International Symposium* (Vol. 22, No. 1, pp. 1967-1976).

Storn, R., & Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization, 11(4),* 341-359.

Systems Engineering Research Center (SERC), 2015. Enterprise Systems and Systems of Systems (ESOS). http://www.sercuarc.org/technical-reports/enterprise-systems-and-systems-of-systems/

Tibshirani, R., Walther, G., & Hastie, T. (2001). Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology), 63(2),* 411-423.

Tran, P., Douglas, G., & Watson, C. Joint Interoperability Certification. Joint Interoperability Test Command. 2005, 11.

Trebi-Ollennu, A., & White, B. A. (1997). Multiobjective fuzzy genetic algorithm optimisation approach to nonlinear control system design. *IEE Proceedings-Control Theory and Applications, 144*(2), 137-142.

Trentesaux, D., Knothe, T., Branger, G., & Fischer, K. (2015). Planning and Control of Maintenance, Repair and Overhaul Operations of a Fleet of Complex Transportation Systems (pp. 175-186). Springer International Publishing.

Tzafestas, S. G. (1994). Fuzzy systems and fuzzy expert control: an overview. *The Knowledge Engineering Review*, *9(03*), 229-268.

Uday, P., & Marais, K. B. (2014). Resilience-based System Importance Measures for System-of-Systems. Procedia Computer Science, 28, 257-264.

Uhlir, P. F., Chen, R. S., Gabrynowicz, J. I., & Janssen, K. (2009). Toward Implementation of the Global Earth Observation System of Systems Data Sharing Principles. *Data Science Journal*, *8*, GEO1-GEO91.

Ullman, D., O'Donnell, J., Edwards, C., Fake, T., & Morschauser, D. (2003). *Use of coastal ocean dynamics application radar (CODAR) technology in US coast guard search and rescue planning* (No. CG-D-09-03). Coast Guard Research and Development Center Groton Ct.

Vaneman, W. K., & Triantis, K. (2007). Evaluating the productive efficiency of dynamical systems. Engineering Management, IEEE Transactions on, 54(3), 600-612.

Vaneman, W. K., & Triantis, K. (2014, 2nd Dec). Designing Resiliency into a System of Systems. System of Systems Engineering Community Information Exchange (SoSECIE). Accessed on 15th January, 2105. http://www.acq.osd.mil/se/webinars/2014_12_02_SoSECIE_Vaneman-brief.pdf

Vetschera, R., Filzmoser, M., & Mitterhofer, R. (2014). An analytical approach to offer generation in concession-based negotiation processes. *Group Decision and Negotiation, 23(1)*, 71-99.

Wang, M. (2015). Editorial: Smart cities of the future: Creating tomorrow's education toward effective skills and career development today. *Knowledge Management & E-Learning: An International Journal (KM&EL)*, *6*(4), 344-355.

Wang, R., & Dagli, C. H. (2011). Executable system architecting using systems modeling language in conjunction with colored Petri nets in a model-driven systems development process. *Systems Engineering, 14(4)*, 383-409.

Wang, R., Agarwal,S., & Dagli, C. (2014). Executable System of Systems Architecture Using OPM in Conjunction with Colored Petri Net: A Module for Flexible Intelligent & Learning Architectures for System of Systems, In *Europe Middle East & Africa Systems Engineering Conference (EMEASEC).*

Wang, X., Shen, X., & Georganas, N. D. (2006, May). A fuzzy logic based intelligent negotiation agent (fina) in eCommerce. *In Electrical and Computer Engineering, 2006. CCECE'06.* Canadian Conference on (pp. 276-279). IEEE.

Wang, Y., & Yang, Y. (2009). Particle swarm optimization with preference order ranking for multi-objective optimization. *Information Sciences, 179(12),* 1944-1959.

Wanyama, T., & Homayoun Far, B. (2007). A protocol for multi-agent negotiation in a group-choice decision making process. *Journal of Network and Computer Applications, 30(3),* 1173-1195.

Wei, G. W. (2010). A method for multiple attribute group decision making based on the ET-WG and ET-OWG operators with 2-tuple linguistic information. *Expert Systems with Applications, 37(12),* 7895-7900.

Wollkind, S., Valasek, J., & Ioerger, T. R. (2004, August). Automated conflict resolution for air traffic management using cooperative multiagent negotiation. In *AIAA guidance, navigation, and control conference* (pp. 16-19).

Wooldridge, M., & Parsons, S. (2000, May). On the use of logic in negotiation. In *Proceedings of the Workshop on Agent Communication Languages, Barcelona, Spain*.

Wu. D., (2013) A Brief Tutorial on Interval Type-2 Fuzzy Sets and Systems.

Xu, R., & Wunsch, D. (2005). Survey of clustering algorithms. *Neural Networks, IEEE Transactions on, 16(3),* 645-678.

Yang, S., Li, M., Liu, X., & Zheng, J. (2013). A Grid-Based Evolutionary Algorithm for Many-Objective Optimization. *IEEE Trans. Evolutionary Computation, 17(5),* 721-736.

Yingchao, Z. (2012, July). Sci. & Technol. on Complex Syst. Simulation Lab., Beijing Inst. of Syst. Eng., Beijing,China. In System of Systems Engineering (SoSE), 2012 7th International Conference on (pp. 509-513). IEEE.

Yen, G.G.; Zhenan He, "Performance Metric Ensemble for Multiobjective Evolutionary Algorithms," *Evolutionary Computation, IEEE Transactions* on , vol.18, no.1, pp.131,144, Feb. 2014

Yu, C., Ren, F., & Zhang, M. (2013). An adaptive bilateral negotiation model based on Bayesian learning. *In Complex Automated Negotiations: Theories, Models, and Software Competitions* (pp. 75-93). Springer Berlin Heidelberg.

Zadeh, L. A. (1965). Fuzzy sets. *Information and control, 8(3),* 338-353.

Zhang, L. (2015). Applying System of Systems Engineering Approach to Build Complex Cyber Physical Systems. In *Progress in Systems Engineering* (pp. 621-628). Springer International Publishing.

Zheng, R., Chakraborty, N., Dai, T., & Sycara, K. (2013, May). Multiagent negotiation on multiple issues with incomplete information. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems* (pp. 1279-1280). International Foundation for Autonomous Agents and Multiagent Systems.

VITA

In August, 2015 Siddhartha Agarwal received his Ph.D. in Systems Engineering from Missouri University of Science and Technology. He was researching in formulation of a domain independent framework for generating meta-architectures for System of Systems. His research interests included optimization, modeling & simulation, machine learning, and computational intelligence.

He received his M.S. degree in Mining Engineering from University of Alaska Fairbanks, U.S.A. in 2010 and the Bachelor of Technology in Mining Engineering from the Indian Institute of Technology-Banaras Hindu University, India in 2006. His work experiences included Iron & Steel industry, Alaska Department of Natural Resources and mining software engineer role.

Siddhartha Agarwal has been a member of the International Council on Systems Engineering (INCOSE), and Institute for Operations Research and the Management Sciences (INFORMS). His research on the system architecture generation and implementation using computational intelligence won the INCOSE Foundation/Stevens Institute Doctoral Award for Promising Research in Systems Engineering and Integration for 2014. He also won the Outstanding Ph.D. Research Award (Systems Engineering & Overall) in the department of Engineering Management & Systems Engineering at Missouri University of Science & Technology for 2013.