28 Feb 2015

Flexible and Intelligent Learning Architectures for SOS (FILA-SoS)

Cihan H. Dagli
Missouri University of Science and Technology, dagli@mst.edu

David Lee Enke
Missouri University of Science and Technology, enke@mst.edu

Nil Ergin

Dincer Konur
Missouri University of Science and Technology, konurd@mst.edu

et. al. For a complete list of authors, see https://scholarsmine.mst.edu/engman_syseng_facwork/525

Flexible and Intelligent Learning Architectures for SoS (FILA-SoS)

Volume 13 – Meta-Architecture Generation Model: Flexible Architecting

Technical Report SERC-2015-TR-021-4

February 28, 2015

Principal Investigators

Dr. Cihan H. Dagli, Missouri University of Science and Technology

Research Team

Co-PI: Dr. David Enke, Missouri S&T

Co-PI: Dr. Nil Ergin, Penn State University

Co-PI: Dr. Dincer Konur, Missouri S&T

Co-PI: Dr. Ruwen Qin, Missouri S&T

Co-PI: Dr. Abhijit Gosavi, Missouri S&T

Dr. Renzhong Wang

Missouri S&T Graduate Students

Louis Pape II, Siddhartha Agarwal and Ram Deepak Gottapu

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 (Task Order 033, RT 48). SERC is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

Multi-faceted systems of the future will entail complex logic and reasoning with many levels of reasoning in intricate arrangement. The organization of these systems involves a web of connections and demonstrates self-driven adaptability. They are designed for autonomy and may exhibit emergent behavior that can be visualized. Our quest continues to handle complexities, design and operate these systems. The challenge in Complex Adaptive Systems design is to design an organized complexity that will allow a system to achieve its goals. This report attempts to push the boundaries of research in complexity, by identifying challenges and opportunities. Complex adaptive system-of-systems (CASoS) approach is developed to handle this huge uncertainty in socio-technical systems.

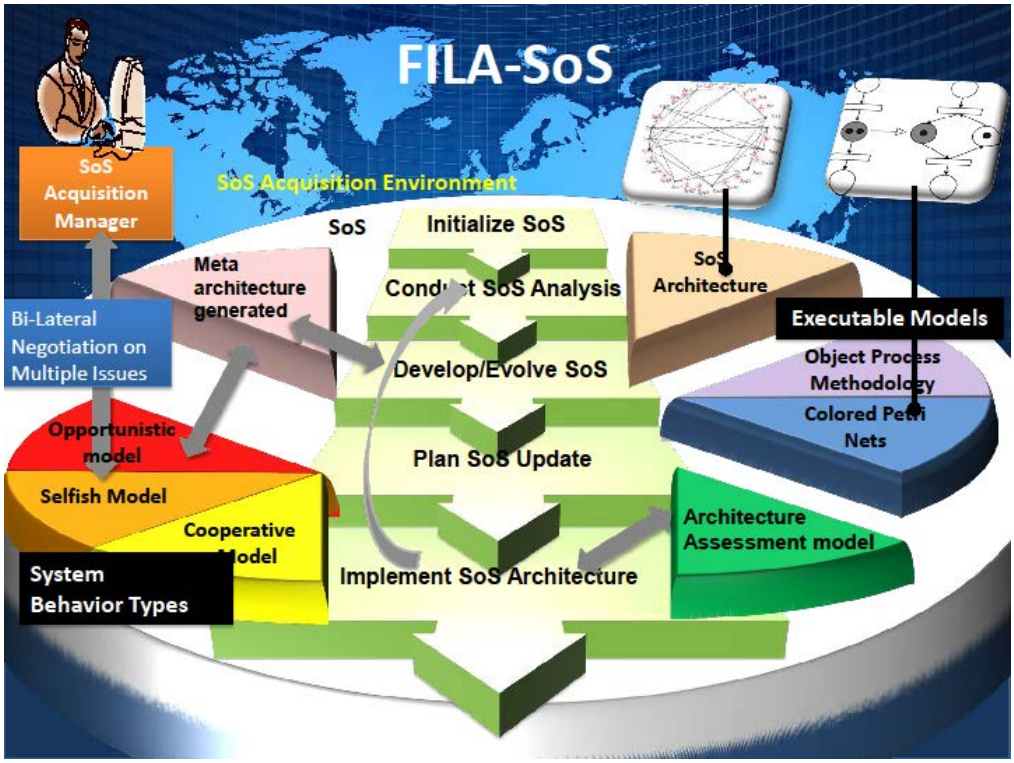
Although classically (Dahmann, Rebovich, Lowry, Lane, & Baldwin, 2011) four categories of SoS are described in literature namely; Directed, Collaborated, Acknowledged and Virtual. However, there exist infinitely many SoS on the edges of these categories thus making it a continuum. Many SoS with different configurations can fill this gap. These four types of SoS vary based on their degree of managerial control over the participating systems and their structural complexity. The spectrum of SoS ranges from Directed SoS that represents complicated systems to Virtual SoS that are complex systems.

Acknowledged SoS lie in between this spectrum. This particular SoS is the focal point of our research endeavor. Acknowledged SoS and Directed SoS share some similarities such as both have (Dahman & Baldwin, 2011) SoS objectives, management, funding and authority. Nevertheless, unlike Directed SoS, Acknowledged SoS systems are not subordinated to SoS. However, Acknowledged SoS systems retain their own management, funding and authority in parallel with the SoS. Collaborative SoS are similar to Acknowledged SoS systems in the fact that systems voluntarily work together to address shared or common interest.

Flexible and Intelligent Learning Architectures for SoS (FILA-SoS) integrated model is developed in this research task provides a decision making aid for SoS manager based on the wave model. The model developed called the FILA-SoS does so using straightforward system definitions methodology and an efficient analysis framework that supports the exploration and understanding of the key trade-offs and requirements by a wide range system-of-system stakeholders and decision makers in a short time. FILA-SoS and the Wave Process address four of the most challenging aspects of system-of-system architecting:

1. Dealing with the uncertainty and variability of the capabilities and availability of potential component systems
2. Providing for the evolution of the system-of-system needs, resources and environment over time
3. Accounting for the differing approaches and motivations of the autonomous component system managers
4. Optimizing system-of-systems characteristics in an uncertain and dynamic environment with fixed budget and resources

Some of the highlights of FILA-SoS are listed in terms of its capabilities, value added to systems engineering, ability to perform “What-if Analysis”, modularity of integrated models, its potential applications in the real world and future additions to the current version.



FILA-SoS has a number of unique capabilities such as integrated model for modeling and simulating SoS systems with evolution for multiple waves. It also has modularity in the structure where the models can be run independently and in conjunction with each other. Besides there are a couple of different models for both architecture generation and SoS behavior and various individual system behavior negotiation models between SoS and individual systems. In terms of value added FILA-SoS aids the SoS manager in future decision making. It also helps in understanding the emergent behavior of systems in the acquisition environment and impact on SoS architecture quality. FILA-SoS serves as an artifact to study the dynamic behavior of different type of systems (non-cooperative, semi-cooperative, cooperative). It enables us to identify intra and interdependencies among SoS elements and the acquisition environment. FILA-SoS can provide a “What-if” Analysis depending on variables such as SoS funding and capability priority that can be changed as the acquisition progresses through wave cycles. It has the ability to simulate any architecture through colored petri nets. In addition, it can simulate rules of engagement & behavior settings: all systems are non-cooperative, all systems are semi-cooperative, and all systems are cooperative or a combination. Some of the potential applications include modeling a wide variety of complex systems models such as logistics, and cyber-physical systems. It also acts as a test-bed for decision makers to evaluate operational guidelines and principles for managing various acquisition environment scenarios. Future Capabilities that are currently in progress are extending the model to include multiple interface alternatives among systems and incorporation of risk models into environmental scenarios.

Integrated Model Structure for FILA-SoS Version 1.0 is described. It provides a short description of all independent models that make up the FILA-SoS integrated model and reports the workings of the model with three notional System-of-Systems namely; Toy Problem for aircraft carrier performance assessment, ISR (intelligence surveillance and reconnaissance) and SAR (search and rescue).

The project reports span 17 volumes. Each report describes the various aspects of the FILA-SOS integrated model:

Volume 1: Integrated Model Structure

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; Toy Problem for 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.

Volume 2: Meta-Architecture Generation Multi-Level Model

Volume 2 describes Meta-Architecture Generation Multi-Level Model. The multi-level meta-architecture generation model considers constructing an SoS architecture such that each capability is provided by at least one system in the SoS and the systems in the SoS are able to communicate with each other. Secondly, it has multiple objectives for generating a set of SoS architectures namely; maximum total performance, minimum total costs and minimum deadline. Finally, the model establishes initial contracts with systems to improve performances.

Volume 3: Fuzzy-Genetic Optimization Model

Volume 3 illustrates the second meta-architecture generation model known as the Fuzzy-Genetic optimization model. This model is based on evolutionary multi-objective optimization for SoS architecting using genetic algorithms and four key performance attributes (KPA) as the objective functions. It also has a type-1 fuzzy assessor for dynamic assessment of domain inputs and that forms the fitness function for the genetic algorithm. It returns the best architecture (meta-architecture) consisting of systems and their interfaces. It is a generalized method with application to multiple domains such as Gulf War Intelligence/Surveillance/Reconnaissance Case, Aircraft Carrier Performance Assessment Case and Alaskan Maritime Search and Rescue Case.

Volume 4: Architecture Assessment Model

Volume 4 describes an Architecture Assessment Mode that can capture 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. Assessment is based on a given meta-architecture alternative. This is done using type-1 fuzzy sets and fuzzy inference engine. The model provides numerical values for meta-architecture quality.

Volume 5: Cooperative System Negotiation Model

Volume 5 specifically describes the Cooperative System Negotiation Model. 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.

Volume 6: Non-Cooperative System Negotiation Model

Volume 6 goes on to describe the Non-Cooperative System Negotiation Model in which 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.

Volume 7: Semi-Cooperative System Negotiation Model

Volume 7 describes the third and last system negotiation model, 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.

Volume 8: Incentive based Negotiation Model for System of Systems

Volume 8 explains the SoS negotiation model also called the Incentive Based Negotiation Model for System of Systems. This model is based on two key assumptions that are to design a contract to convince the individual systems to join the SoS development and motivate individual systems to do their tasks well. Game theory and incentive based contracts are used in the negotiation model that will maximize the welfare for parties involved in the negotiation. SoS utility function takes into account local objectives for the individual systems as well as global SoS objective whereas the incentive contract design persuades uncooperative systems to join the SoS development.

Volume 9: Model for Building Executable Architecture

Volume 9 illustrates the process of building Executable Architectures for SoS. The operations of the SoS is a dynamic process with participating system interacting with each other and exchange various kinds of resources, which can be abstract information or physical objects. This is done through a hybrid structure of OPM (Object process methodology) and CPN (Colored petri nets) modeling languages. The OPM model is intuitive and easy to understand. However, it does not support simulation, which is required for accessing the behavior related performance. This is achieved by mapping OPM to CPN, which is an executable simulation language. The proposed method can model the interactions between components of a system or subsystems in SoS. In addition, it can capture the dynamic aspect of the SoS and simulate the behavior of the SoS. Finally, it can access various behavior related performance of the SoS and access different

constitutions or configurations of the SoS which cannot be incorporated into the meta-architecture generation models of Volume 2 & 3.

Volume 10: Integrated Model Software Architecture and Demonstration FILA-SoS Version 1.0
Volume 10 elucidates the Integrated Model Software Architecture and Demonstration based on the models described above. Volume 11 and thereon the reports are aimed at the upcoming newer version 2.0 of FILA-SoS.

Volume 11: Integrated Model Structure FILA-SoS Version 2.0
Volume 11 provides Integrated Model Structure for FILA-SoS Version 2.0 that could be implemented in a new software environment.

Volume 12: Complex Adaptive System-of-System Architecture Evolution Strategy Model for FILA-SoS Version 2.0
Volume 12 provides a model to answer the first research question "What is the impact of different constituent system perspectives regarding participating in the SoS on the overall mission effectiveness of the SoS?" 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.

Volume 13: On the Flexibility of Systems in System of Systems Architecting: A new Meta-Architecture Generation Model for FILA-SoS Version 2.0
Volume 13 is termed the Flexibility of Systems in System of Systems Architecting: A new Meta-Architecture Generation Model for FILA-SoS Version 2.0. The research question is answered through an alternative technique to meta-architecture generation besides the one described in Volume 2.

Volume 14: Assessing the Impact on SoS Architecture Different Level of Cooperativeness: A new Model for FILA-SoS Version 2.0
Volume 14 proposes a new method for Assessing the Impact on SoS Architecture Different Level of Cooperativeness. Second research question is answered through a model that allows different levels of cooperativeness of individual systems.

Volume 15: Incentivizing Systems to Participate in SoS and Assess the Impacts of Incentives: A new Model for FILA-SoS Version 2.0
Volume 15 is an extension of previous systems negotiation models based on incentivizing and is aptly called Incentivizing Systems to Participate in SoS and Assess the Impacts of Incentives: A new Model for FILA-SoS Version 2.0. It also provides an approach to answer the third research question "How should decision-makers incentivize systems to participate in SoS, and better understand the impact of these incentives during SoS development and effectiveness?". This model is based on the fact that providing incentives only depending on the outcome may not be enough to attract the attention of the constituent systems to participate in SoS mission. Therefore, this model extends the approach as described in Volume 8 while considering the

uncertainty in the acquisition environment. The incentive contract is designed based on the objectives of the SoS and the individual systems. Individual system's objective is to secure highest incentives with minimal effort while the SoS manager's goal is to convince individual systems to join the SoS development while maximizing its own utility.

Volume 16: Integrated Model Software Architecture for FILA-SoS Version 2.0
Volume 16 gives an overview of the integrated model architecture in version 2.0 of the software. It includes all old and new models previously mentioned.

Volume 17: FILA-SoS Version 1.0 Validation with Real Data
Volume 17 describes the validation of the FILA-SoS Version 1.0 with a real life data provided by MITRE Corporation by from a moderately sized SoS.

INTRODUCTION

MOTIVATION FOR RESEARCH

In the real world, systems are complex, non-deterministic, evolving, and have human centric capabilities. 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 include transportation, health, energy, cyber physical systems, economic institutions and communication infrastructures.

In addition, the idea of "System-of-Systems" is an emerging and important multidisciplinary area. An SoS is defined as a set or arrangement of systems that results when independent and useful systems are integrated into a larger system that delivers unique capabilities greater than the sum of the capabilities of the constituent parts. Either of the systems alone cannot independently achieve the overall goal. 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).

Since SoS grow in complexity and scale with the passage of time it requires architectures that will be necessary for understanding and governance and for proper management and control. 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). It narrows the scope of the fairly large domain space and boundary. 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 (Dagli & Ergin, 2008). 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. There are four types of SoS that are described below:

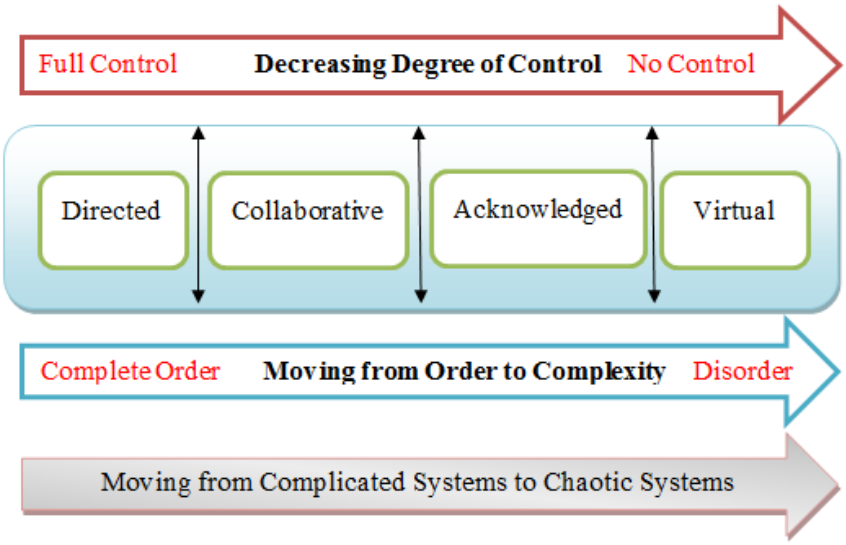


Figure 1 Schematic Drawing of Four Classical Types of SoS Based on Degree of Control and Degree of Complexity

Virtual

- Virtual SoS lack a central management authority and a centrally agreed upon purpose for the system-of-systems.
- Large-scale behavior emerges—and may be desirable—but this type of SoS must rely upon relatively invisible mechanisms to maintain it.

Collaborative

- In collaborative SoS the component systems interact more or less voluntarily to fulfill agreed upon central purposes.

Acknowledged (*FILA-SoS integrated model is based on Acknowledged SoS*)

- Acknowledged SoS have recognized objectives, a designated manager, and resources for the SoS; however, the constituent systems retain their independent ownership, objectives, funding, and development and sustainment approaches.
- Changes in the systems are based on collaboration between the SoS and the system.

Directed

- Directed SoS's are those in which the integrated system-of-systems is built and managed to fulfill specific purposes.
- It is centrally managed during long-term operation to continue to fulfill those purposes as well as any new ones the system owners might wish to address.
- The component systems maintain an ability to operate independently, but their normal operational mode is subordinated to the central managed purpose.

This research is based on Acknowledged SoS. The major objectives of the 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.

The dynamic planning for a SoS is a challenging endeavor. Department of Defense (DoD) programs 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 DoD to be able to look at the future scenarios and critically assess the impact of technology and stakeholder changes. The DoD currently is looking for options that signify affordable acquisition selections and lessen the cycle time for early acquisition and new technology addition. FILA-SoS provides a decision aid in answering some of the questions.

This volume gives an overview of a novel methodology known as the Flexible Intelligent & Learning Architectures in System-of-Systems (FILA-SoS). Some the challenges that are prevalent in SoS architecting and how FILA-SoS attempts to address them is explained in the next section.

SYSTEM OF SYSTEM CHALLENGES

All these recent developments are helping us to understand Complex Adaptive Systems. They are at the edge of chaos as they maintain dynamic stability through constant self-adjustment and evolution. Chaos and order are two complementary states of our world. A dynamic balance exists between these two states.

Order and structure are vital to life. Order ensures consistency and predictability and makes the creation of systems possible. However, too much order leads to rigidity and suppresses creativity. Chaos constantly changes the environment creating disorder and instability but can also lead to emergent behavior and allows novelty and creativity. Thus, sufficient order is necessary for a system to maintain an ongoing identity, along with enough chaos to ensure growth and development. The challenge in Complex Adaptive Systems design is to design an organized complexity that will allow a system to achieve its goals. SoS is a complex systems by its nature due to the following characteristics that are component systems are operationally independent elements and also managerially independent of each other. This means that component systems preserve existing operations independent of the SoS. SoS has an evolutionary development and due to the large scale complex structure shows an emergent behavior. Emergence means the SoS performs functions that do not reside in any one component system.

2012 INCOSE SoS working group survey identified seven ‘pain points’ raising a set of questions for systems engineering of SoS which are listed in Table 1 (Dahman, 2012).

Table 1 System of Systems and Enterprise Architecture Activity

Pain Points	Question
Lack of SoS Authorities & Funding	What are effective collaboration patterns in systems of systems?
Leadership	What are the roles and characteristics of effective SoS leadership?
Constituent Systems	What are effective approaches to integrating constituent systems into a SoS?
Capabilities & Requirements	How can SE address SoS capabilities and requirements?
Autonomy, Interdependencies & Emergence	How can SE provide methods and tools for addressing the complexities of SoS interdependencies and emergent behaviors?
Testing, Validation & Learning	How can SE approach the challenges of SoS testing, including incremental validation and continuous learning in SoS?
SoS Principles	What are the key SoS thinking principles, skills and supporting examples?

The importance and impact on systems engineering of each pain point is illustrated below:

- **Lack of SoS Authorities & Funding and Leadership** pose several and severe governance and management issues for SoS. This conditions has a large impact on the ability to implement systems engineering (SE) in the classical sense to SoS. In addition, this problem affects the modeling & simulation activities.
- **Constituent Systems** play a very important role in the SoS. As explained earlier usually they have different interests and ambitions to achieve, which may or may not be aligned with the SoS.. Similarly models, simulations and data for these systems will naturally have to be attuned to the specific needs of the systems, and may not lend themselves easily to supporting SoS analysis or engineering
- **Autonomy, Interdependencies & Emergence** is ramifications of the varied behaviors and interdependencies of the constituent systems making it complex adaptive systems. Emergence comes naturally in such a state, which is often unpredictable. While modeling & simulation can aid in representing and measuring these complexities, it is often hard to achieve real life emergence. This is due to limited understanding of the issues that can bring up serious consequences during validation.
- **Capability of the SoS** and the individual systems capability needs may be high level and need definition in order to align them with the requirements of the SoS mission. The SoS mission is supported by constituent systems, which may not be able (or willing) to address them.
- **Testing, Validation & Learning** becomes difficult since the constituent systems continuously keep evolving, adapting, as does the SoS environment which includes stakeholders, governments, etc. Therefore creating a practical test-bed for simulating the large dynamic SoS is a challenge in itself. Again modeling & simulation can solve part of the problem such as enhancing live test and addressing risk in SoS when testing is not feasible; however, this requires a crystal clear representation of the SoS which can be difficult as discussed in earlier points.

- **SoS Principles** are still being understood and implemented. Therefore, the rate of success is yet to be addressed formally. This poses some pressure on the progress of SoS engineering. Similarly, there is an absence of a well-established agreeable space of SoS principles to drive development and knowledge. This constricts the effective use of potentially powerful tools.

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 environment 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 are the 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

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.” 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. The methodology outlined in this research and technical report falls under a multi-level plug-and-play type of modeling approach to address various aspects of SoS acquisition environment: SoS architecture evaluation, SoS architecture evolution, and SoS acquisition process dynamics including behavioral aspects of constituent systems.

HOW DOES FILA-SOS ADDRESS SOS PAIN POINTS

The first pain point is Lack of SoS Authorities & Funding which begs a question "What are effective collaboration patterns in systems of systems?"

Since there is lack of SoS Authority but more so persuasion involved in the workings of a SoS, systems are allowed to negotiate with the SoS manager. Deadline for preparation, funding and performance required to complete the mission are some of the issues that form the negotiation protocol. Besides different combination of behavior types assigned to the systems can help us gauge the best effective collaboration patterns in systems of systems after the end of negotiations.

The leadership issues pose the question, "What are the roles and characteristics of effective SoS leadership?" This is addressed by incorporating views from multiple stakeholders while assessing the architecture's quality. In addition, we maintain that the characteristics are similar to what an Acknowledged SoS manager would have while distributing funds and resources among systems for a joint operation. The SoS manager also has the opportunity to form his decision based on most likely future scenarios, thus imparting him an edge as compared to other models. This will improve the process of acquisition in terms of overall effectiveness, less cycle time and integrating legacy systems. Overall, the role of the leadership is presented a guide than someone who would foist his authority.

The third pain point question, "What are effective approaches to integrating constituent systems into a SoS? is addressed below. A balance has to be maintained during acquisition between amount of resources used and the degree of control exercised by the SoS manager on the constituent systems. The meta-architecture generation is posed as a multi-objective optimization problem to address this pain point. The constituent systems and the interfaces between them are selected while optimizing the resources such as operations cost, interfacing cost, performance levels etc. The optimization approach also evaluates the solutions based on views of multiple stakeholders integrated together using a fuzzy inference engine.

How can SE address capabilities and requirements? is the fourth pain point and is answered in this paragraph. Organizations that acquire large-scale systems have transformed their attitude to acquisition. Hence, these organizations now want solutions to provide a set of capabilities, not

a single specific system to meet an exact set of specifications. During the selection process of systems it is ensured that, a single capability is provided by more than one system. The idea is to choose at least one systems having unique capability to form the overall capability of the SoS.

The fifth pain point on autonomies, emergence and interdependencies is one of the most important objectives of this research. This objective can be described as “How can SE provide methods and tools for addressing the complexities of SoS interdependencies and emergent behaviors?”. Each system has an autonomous behavior maintained through pre-assigned negotiation behaviors, differ operations cost, interfacing cost and performance levels while providing the same required capability. The interfacing among systems is encouraged to have net-centric architecture. The systems communicate to each other through several communication systems. This ensures proper communication channels. Together the behavior and net-centricity make it complex systems thus bringing out the emergence needed to address the mission.

FILA-SoS is an excellent integrated model for addressing the complexities of SoS interdependencies and emergent behaviors as explained in the above paragraphs.

As for the sixth pain point on testing, validation and learning goes, FILA-SoS has been tested on three notional examples so far the ISR, Search and Rescue (SAR) and the Toy problem for Aircraft Carrier Performance Assessment. For ISR (refer to Figure 2) a guiding physical example is taken from history. During the 1991 Gulf War, Iraqi forces used mobile SCUD missile launchers called Transporter Erector Launchers (TELS) to strike at Israel and Coalition forces with ballistic missiles. Existing intelligence, surveillance, and reconnaissance (ISR) assets were inadequate to find the TELs during their vulnerable setup and knock down time. The “uninhabited and flat” terrain of the western desert was in fact neither of those things, with numerous Bedouin goat herders and their families, significant traffic, and thousands of wadis with culverts and bridges to conceal the TELs and obscure their movement.

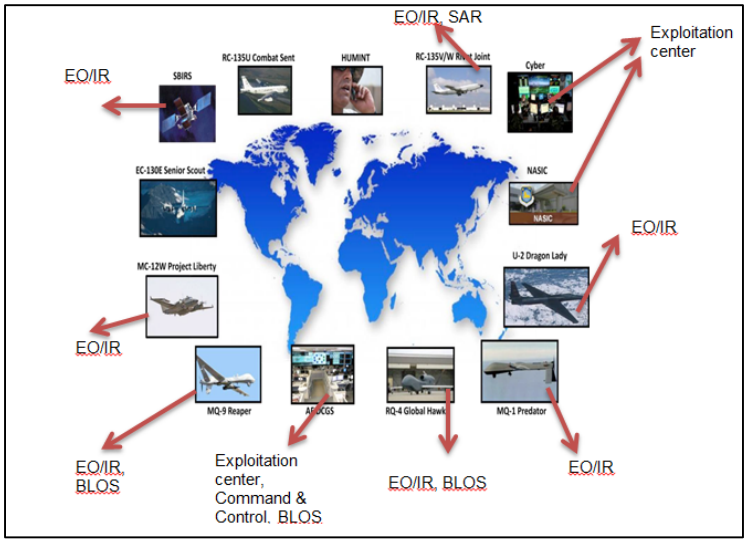


Figure 2 ISR System-of-Systems for Testing FILA-SoS

A Coast Guard Search and Rescue (SAR) (Figure 3) SoS engineering and development problem is selected for serving the Alaskan coast. Detailed information about this case study can be found in Dagli et al (2013). There is increasing use of the Bering Sea and the Arctic by commercial fisheries, oil exploration and science, which increases the likelihood of occurrence of possible SAR scenarios.

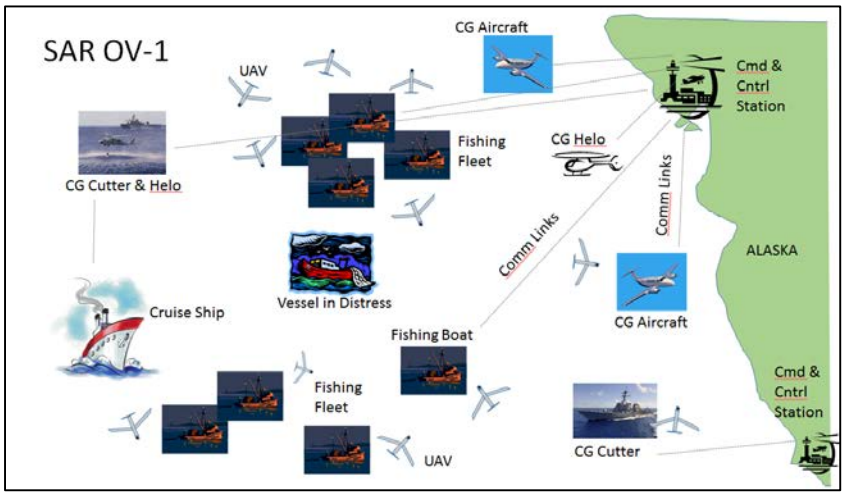


Figure 3 SAR System-of-Systems for Testing FILA-SoS

The toy problem for assessing the performance of the aircraft carrier involves multiple systems such as satellites, uav's and ground station that support the aircraft carrier to fulfill the mission (refer to Figure 4). The results have been obtained for multiple waves of the evolution process for all the examples.

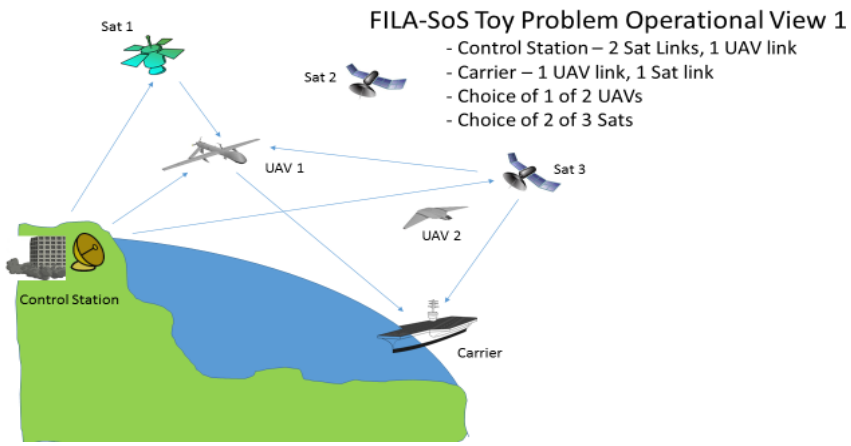


Figure 4 Aircraft Carrier Performance Assessment for Testing FILA-SoS

These example discussed above clearly show the domain independence of FILA-SoS.

FILA-SoS is a novel method of making sequential decisions over a period for SoS development. The goal is to apply the integrated model to dynamically evolve SoS architecture and optimize SoS architecture, design and validate through simulation tools. The integrated model structure can be applied to various application areas including development of dynamic water treatment SoS architecture, development of dynamic Air Traffic Management SoS, and development of autonomous ground transport SoS. 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, cyber-physical systems and similar 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.

OVERVIEW OF THE FILA-SOS INTEGRATED MODEL

In this section an overview of FILA-SoS is described. The model developed called the FILA-SoS is using straightforward system definitions methodology and an efficient analysis framework that supports the exploration and understanding of the key trade-offs and requirements by a wide range system-of-system stakeholders and decision makers in a short time. FILA-SoS and the Wave Process address four of the most challenging aspects of system-of-system architecting:

- Dealing with the uncertainty and variability of the capabilities and availability of potential component systems.
- Providing for the evolution of the system-of-system needs, resources and environment over time.
- Accounting for the differing approaches and motivations of the autonomous component system managers.
- Optimizing system-of-systems characteristics in an uncertain and dynamic environment with fixed budget and resources

---

DEFINITION OF VARIABLES FOR SOS

This list comprises of the notation for variables used to solve the Acknowledged SoS architectural evolution problem:

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

c_j: *j* ∈ ***J***, ***J****= {1, 2,…, M}*:
Constituent system capabilities required

s_i: *i* ∈ ***I***, ***I****= {1, 2,…, N}*:
Total number of systems present in the SoS problem

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

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

$a_{ij} = 0\ otherwise$

P_i: Performance of system i for delivering all capabilities $\sum_j a_{ij}$

F_i: Funding of system i for delivering all capabilities $\sum_j a_{ij}$

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

IF_{ik} Interface between systems $i\ and\ k$ *s.t. s≠ k, k ∈ I*

IC_i: The cost for development of interface for system i

OC_i: The cost of operations for system i

KP_r : *r* ∈ ***R***, *R= {1, 2,…, Z}*:
The key performance attributes of the SoS

FA: Funding allocated to SoS Manager

p= {1, 2,…, P}:
Number of negotiation attributes for bilateral negotiation

t_{max}: Total round of negotiations possible

t : Current round of negotiation (epochs)

t_{max}: Total round of negotiations possible

$V_{pi}^{SoS}(t)$: 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

The model involves a list of stakeholders such as the Acknowledged SoS manager, system owners/managers, SoS environment etc.

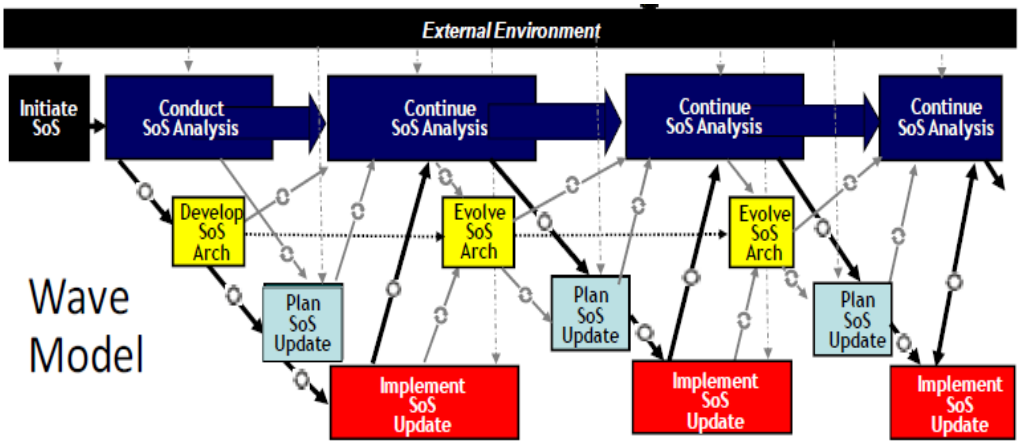


Figure 5 The Wave Model of SoS initiation, Engineering, and Evolution

FILA-SoS follows the Dahmann's proposed SoS Wave Model process for architecture development of the DoD acquisition process as depicted in Figure 5. 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.

Processes involved in the wave model and their analog in FILA-SoS can be explained through the first stage of Initializing the SoS. In terms of initializing, wave process requires to understand the SoS objectives and operational concept (CONOPS), gather information on core systems to support desired capabilities. This 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 to the FILA-SoS for the participating systems s_i. FILA-SoS requires t_{max} the number of negotiation cycles, selection of the meta-architecture modelling procedure and system negotiation models assigned to participating systems.

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. For FILA-SoS the baseline architecture is called as the meta-architecture. Meta-architecture is basically picking up the systems s_i and their respective capabilities a_{ij}. Meta-architecture modelling requires the values for KP_t , the key performance attributes of the SoS, P_i (Performance of system i) , F_i (Funding of system i), and D_i deadline to participate in this round of mission development for system i which is

assumed to be the total for all capabilities possessed by system i. The cost for development of a single interface for system i, IC_i and OC_i the cost of operations for system i is also needed at this stage of the model. 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. Within FILA-SoS this signals the command to send connectivity request to individual systems and starting the negotiation between SoS and individual systems. This stage requires the number of negotiation attributes P for a bilateral negotiation between Acknowledged SoS manager and each systems i selected in the meta-architecture and t_{max} which denotes the total round of negotiations possible.

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. To reflect that in FILA-SoS determines which systems to include based on the negotiation outcomes and form a new 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. In the FILA-SoS the negotiated architecture quality is evaluated based on KP_r, key performance attributes of the SoS. If the architecture quality is not up to a predefined quality or TQ the threshold architecture quality the Acknowledged SoS manager and systems i selected in the meta-architecture go for renegotiations. 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. FILA-SoS also has the proficiency to convert the meta-architecture into an executable architecture using the Object Process Model (OPM) and Colored Petri Nets (CPN) for overall functionality and capability of the meta-architecture. These executable architectures are useful in providing the much-needed information to the SoS coordinator for assessing the architecture quality and help him in negotiating better.

Some of the highlights of FILA-SoS are described in terms of its capabilities, value added to systems engineering, ability to perform “What-if Analysis”, modularity of integrated models, its potential applications in the real world and future additions to the current version. The most important capability of FILA-SoS is it being an integrated model for modeling and simulating SoS systems with evolution for multiple waves. Secondly, all models within FILA-SoS can be run independently and in conjunction with each other. Thirdly, there are two model types that represent SoS behavior and various individual system behaviors. Finally, it has the capacity to study negotiation dynamics between SoS and individual systems.

The value added by FILA-SoS to systems engineering is it aids the SoS manager in future decision making, can help in understanding the emergent behavior of systems in the acquisition environment and its impact on SoS architecture quality. Besides, it has three independent systems behavior models, which are referred to as cooperative, semi-cooperative and non-cooperative. These behavior models are used to Study the dynamic behavior of different type of systems while they are negotiating with SoS manager. In addition, FILA-SoS assists in identifying intra and interdependencies among SoS elements and the acquisition environment.

FILA-SoS also can facilitate a “What-if” Analysis using variables such as SoS funding and capability priority that can be changed as the acquisition progresses though wave cycles. The parameter setting for all negotiation models can be changed and rules of engagement can be simulated for different combinations of systems behaviors.

Potential Application of FILA-SoS include complex systems models such as logistics, cyber-physical systems. In addition, it can act as test-bed for decision makers to evaluate operational guidelines and principles for managing various acquisition environment scenarios. While the future capabilities that we would like to be included are extending the model to include multiple interface alternatives among systems and incorporation of risk models into environmental scenarios.

INDEPENDENT MODULES OF 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. An overall view is presented in Figure 6.

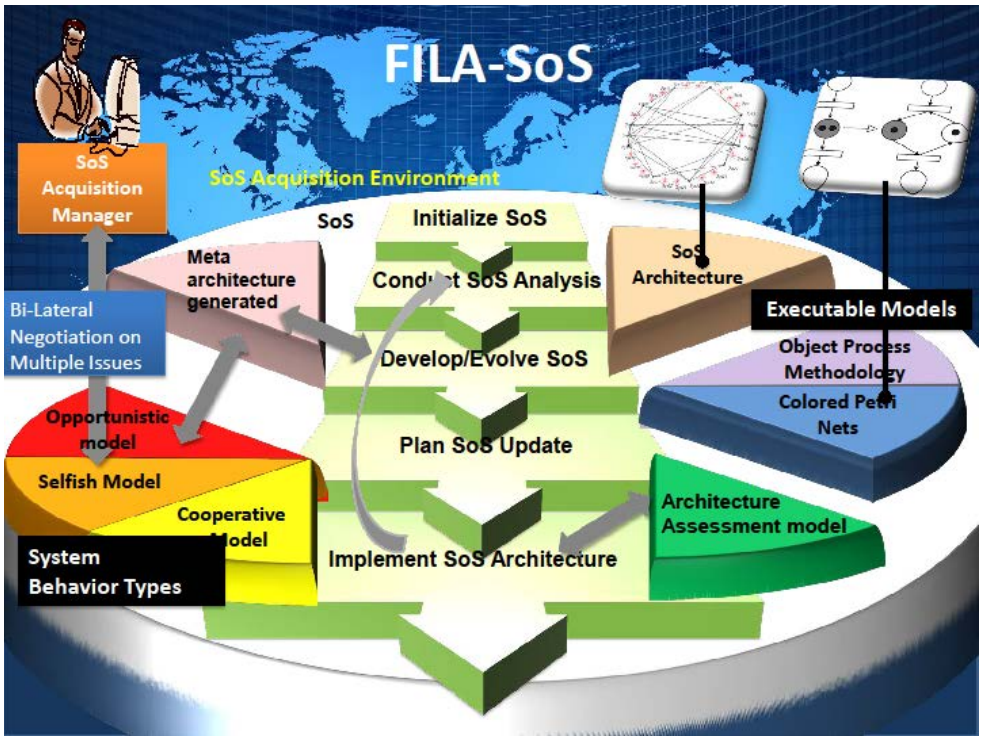


Figure 6 Integrated modules within FILA- SoS

All the independent models are listed below for reference:

- Meta-Architecture Generation Model
- Architecture Assessment Model
- SoS Negotiation Model
- System Negotiation Model: Non-Cooperative

- System Negotiation Model: Cooperative
- System Negotiation Model: Semi-Cooperative
- Executable Architecting Model: OPM & CPN
- Overall Negotiation Framework

The first meta-architecture generation method is fuzzy-genetic optimization model (Pape, Agarwal, Giammarco & Dagli, 2014). This model is based on evolutionary multi-objective optimization for SoS architecting with many key performance attributes (KPA). It also has a type-1 fuzzy assessor for dynamic assessment of domain inputs and that forms the fitness function for the genetic algorithm. It returns the best architecture (meta-architecture) consisting of systems and their interfaces. It is a generalized method with application to multiple domains such as Gulf War Intelligence/Surveillance/Reconnaissance Case and Alaskan Maritime Search and Rescue Case.

The second meta-architecture generation model is based on multi-level optimization (Konur & Dagli, 2014). In this model, architecting is done in two rounds: the first being the initiating the SoS by selecting the systems to be included in the SoS and then improving the SoS's performance by allocating funds to participating systems. The model is generic based on multiple attributes such as maximum performance, minimum cost and minimum deadline. It based on a Stackelberg game theoretical approach between the SoS architect and the individual systems.

The particle swarm optimization (Agarwal, Pape, & Dagli, 2014) technique for meta-architecture generation is similar to fuzzy-genetic model. Except for the fact that evolutionary optimization technique in this case is based on swarm intelligence. In addition, there are some new key performance attributes used to calculate the architectures quality. Cuckoo search optimization (Agarwal, Wang, & Dagli, 2014) based meta-architecture is again anew biologically inspired method of optimization. It has been shown that it in certain cases it performs better than PSO.

The first architecture assessment method is based on type-1 fuzzy logic systems (FLS) (Pape et al., 2013). The Key Performance Parameters (KPP) chosen are performance, affordability, flexibility, and robustness. It can capture the viewpoints of multiple stakeholders'. It can also accommodate any number of KPPs.

Another architecture assessment method is based on type-2 fuzzy modular nets (Agarwal, Pape & Dagli, 2014). The attributes used for evaluation were Performance, Affordability, Developmental Modularity, Net-Centricity and Operational Robustness. Type-1 fuzzy sets are able to model the ambiguity in the input and output variables. However, type-1 fuzzy sets are insufficient in characterizing the uncertainty present in the data. Type-2 fuzzy sets proposed by Zadeh (1975) can model uncertainty and minimize its effects in FLS (Mendel & John, 2002).

It is not possible to implement such meta-architecture without persuading the systems to participate, hence to address the issue a negotiation model is proposed based on game theory (Ergin, 2104). It is an incentive based negotiation model to increase participation of individual systems into Search and Rescue SoS. The model provides a strategy for SoS management to determine the appropriate amount of incentives necessary to persuade individual systems while

achieving its own goal. The incentive contract is designed based on the objectives of the SoS and the individual systems. Individual system's objective is to secure highest incentives with minimal effort while the SoS manager's goal is to convince individual systems to join the SoS development while maximizing its own utility. Determining the incentives for individual systems can be formulated as a multi-constraint problem where SoS manager selects a reward for the individual system such that the reward will maximize SoS manager's expected utility while satisfying the constraints of the individual systems.

Another negotiation model based on clustering and neural networks is developed (Agarwal, Saferpour & Dagli, 2014). This model involves adapting the negotiation policy based on individual systems behavior that is not known to the SoS manager. The behavior is predicted by clustering the difference of multi-issue offers. Later the clustered data is trained using supervised learning techniques for future prediction.

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). To facilitate analysis of interactions between the participating systems in achieving the overall SoS capabilities, an executable architecture model is imperative. In this research, a modeling approach that combines the capabilities of OPM and CPN is proposed. Specifically, OPM is used to specify the formal system model as it can capture both the structure and behavior aspects of a system in a single model. CPN supplements OPM by providing simulation and behavior analysis capabilities. Consequently, a mapping between OPM and CPN is needed. OPM modeling supports both object-oriented and process-oriented paradigm. CPN supports state-transition-based execution semantics with discrete-event system simulation capability, which can be used to conduct extensive behavior analyses and to derive many performance metrics.

ON THE FLEXIBILITY OF SYSTEMS IN SYSTEM OF SYSTEMS ARCHITECTING: A NEW META-ARCHITECTURE GENERATION MODEL VERSION 2.0

System of Systems (SoS) architecting requires analyzing a set of individual systems simultaneously in order to build a connected SoS, which can provide the capabilities needed. In general, the systems can provide a set of capabilities and the SoS architect needs to decide which systems to include in the SoS so that each capability is provided by at least one system. In this case, the systems are inflexible, i.e., a selected system will contribute to the SoS with all the capabilities it can provide. On the other hand, if SoS architect can incentivize systems to contribute with specific capabilities instead of all of their capabilities, it might be possible to build a better SoS in terms of not only one objective but all objectives considered. In this study, we compare SoS architecting with inflexible and flexible systems and quantify the value of the flexibility of the systems. We formulate the SoS architecting problems with inflexible and flexible systems as multi-objective nonlinear binary programming models and propose an evolutionary algorithm for each model. The evolutionary algorithms output a set of Pareto efficient SoS's for the architect. Upon comparing the Pareto fronts of inflexible and flexible models, we quantify the value of systems' flexibilities. We analyze the effects of systems' flexibility levels.

SOS ARCHITECTING AND SYSTEM FLEXIBILITY

In many industry, service, and defense enterprises, system engineering plays an important role, as it is able to simultaneously capture the different dynamics among the elements of the whole enterprise working towards common goals. A system can be considered as the smallest element of the overall enterprise and it contributes to the enterprise with its own individual components and unique capabilities. Kaplan (2006) notes that integration of many systems, their capabilities, and the cumulative abilities achieved from their interoperability are crucial for gaining competitive advantage in large business and defense projects. A System of Systems (SoS) is the collection of individual and independent systems that are brought together for specific goals (Gorod et al., 2008, Klein and Vliet, 2013). SoS architecting administers appropriate integration of the systems, ensures connection among the individual systems, and guarantees that the requirements are met overall. Many engineering, design, organizational, information, technology management, and decision making models in manufacturing, health, energy, transportation, logistics, and military are represented as SoS architectures (Jamshidi, 2008, Jamshidi, 2011). In this volume, we analyze a multi-objective SoS architecting problem.

Most of the projects undertaken by the DoD are SoS architecting problems (DoD, 2008). Not only defense projects, but also many strategy development projects for military missions are SoS architecting problems (Owens, 1996, Manthorpe, 1996) and military systems are integrated as SoS architectures (Bergey et al., 2009). DoD (2008) definition of SoS, which is adopted in this study as well, is capability based and SoS is defined as the collection of systems integrated to provide required capabilities. As noted by Domercant and Mavris (2010), this capability based definition is reasonable as military missions are recently more related to capabilities based planning. Furthermore, Dahmann and Baldwin (2008) highlight that independent control of the

individual systems will not achieve operational goals; hence, SoS architecting is crucial in defense projects. Owens (1996), Manthorpe (1996), and Dahmann and Baldwin (2008) list examples of SoS architectures in DoD. Specifically, Kaplan (2006) and Smith et al. (2011) both emphasize that the missions (purposes) are the main drivers for architecting SoS for military projects. The SoS architecting problem analyzed in this volume requires providing a set of capabilities.

There are two main components of SoS: the capabilities, which are determined based on the mission's goals/targets, and the systems, who can contribute with specific capabilities. The SoS architect is the agent constructing the SoS and the constructed SoS should be capable, that is, it should be able to provide a set of precise capabilities. A capability is defined as a skill for performing definite functions (DoD, 2008). Intelligence, surveillance, reconnaissance, defense (air or missile), health, and communication skills are the general capabilities needed in military missions (DoD, 2008, Dahmann and Baldwin, 2008, Bergey et al., 2009). For instance, a capability can be the ability to track moving targets (DoD, 2008). Manthorpe (1996) lists a set of nine capabilities identified for joint war fighting and Konur and Dagli (2014) note that specific search, radar, command and control, exploitation, and communication capabilities are required for targeting Scud TELs during Gulf War. The systems are the entities equipped with such capabilities. Vehicles, softwares, and other systems such as aircrafts, fighters, platforms equipped with weapons, sensors, communication tools and computers, and radars are military systems (Manthorpe, 1996, Dahmann and Baldwin, 2008, Konur and Dagli, 2014). For instance, Owens (1996) gives a list of military systems.

DoD must often combine military systems to perform mission goals (Kaplan, 2006) and Owens (1996) notes that military systems are coming together as SoS architectures. Different agents such as executive offices, principal staff assistants, staff boards, and military committees can take the role of the SoS architect and the SoS architect's problem is then to determine which systems with which capabilities should be included in the architecture (Kaplan, 2006). While architecting the SoS, the SoS architect should take into account the individual system properties and the communication among the systems contributing to the SoS. Different systems can provision different capabilities with distinct costs, performance levels, and schedules; and, the SoS architecture should consist of a set of systems such that each capability is provided by at least one system, i.e., SoS is capable. Furthermore, the SoS architect should ensure that the systems are connected by enabling communication among the systems included in the SoS. Similar SoS architecting models have been investigated in many military projects such as air defense (Maier, 1998, Sommerer et al., 2012), ballistic missile defense (Ender et al., 2010, Garrett et al., 2011), navy carrier strike (Adams and Meyers, 2011), and future combat systems (Pernin et al., 2012). This volume uses operations research tools to analyze SoS architecting problem with two types of systems: inflexible and flexible.

In particular, flexibility can be associated with an individual system or the SoS itself. Roughly, flexibility of a system or a SoS architecture can be described as the system's or the SoS architecture's ability to respond to changes (Saleh et al., 2001, Saleh et al., 2009, Valerdi et al., 2008, Gorod et al., 2008, Ross et al., 2008). This volume analyzes the effects of system flexibility in the process of architecting the SoS. Specifically, a system is defined as inflexible when

engineering design changes within the system are not possible. An inflexible system will, therefore, have a set of fixed capabilities integrated within and it will contribute to the SoS with those capabilities. In case of inflexible systems, the SoS architecting problem is to select the best systems within the SoS considering the architecture objectives. On the other hand, it might be of benefit to the SoS architect that a system, instead of providing all of its capabilities, collaborate with the SoS architect and contribute to the SoS with a subset of its capabilities. Through design changes, some of the capabilities available in a system can be disintegrated from the system and the SoS architect can benefit from the reduction in cost and/or completion time of the SoS (Dahmann and Baldwin, 2008). We refer to such a system as a flexible system. As noted by Kaplan (2006), a flexible system can be guided by the SoS architect. Therefore, in case of flexible systems, the SoS architecting problem is to determine which systems will contribute to the SoS with which capabilities.

Specifically, flexibility of a system can be considered as its cooperativeness with the SoS architect. A flexible system can be considered as cooperative since the system can modify its settings as guided by the SoS architect. Therefore, in this volume, we focus on analyzing the effects of different levels of cooperativeness of the systems on the SoS architectures by proposing mathematical formulations, solutions methods, and numerical analysis for different levels of system flexibility.

The effects of system flexibility and its level on the cost, performance, and agility of the SoS architecture are examined by mathematically formulating SoS architecting problem with both inflexible and flexible systems as multi-objective optimization problems. The flow of actions in both SoS architecting problems is as follows. Prior to physical architecting of the SoS, a set of capabilities required for the SoS are defined considering the mission goals and the systems that can provide these capabilities are specified (the set of the systems with similar capabilities constitute a family of systems, DoD, 2008). During the SoS architecting, in case of inflexible systems, the SoS architect selects the systems to be included in the SoS, the systems contribute to the SoS with their capabilities, and the SoS architect ensures the connectedness of the SoS by establishing the communication interfaces among the systems. In case of flexible systems, the SoS architect selects the systems to be included in the SoS as well as the capabilities that the systems will provide, the systems modify their designs and contribute to the SoS with the requested capabilities, and the SoS architect ensures the connectedness of the SoS by establishing the communication interfaces among the systems. Pernin et al. (2012) note that one can utilize three main objectives in formulating SoS architectures: performance, schedule, and cost. Therefore, similar to Konur and Dagli (2014) as well, in both of the cases, it is assumed that the SoS architect constructs a capable and connected SoS regarding three objectives: maximization of total performance, minimization of completion time, and minimization of total cost.

Two common approaches adopted for solving multi-objective optimization models are reducing the multi-objective model into a single-objective model and generating the set of non-dominated solutions. A multi-objective model can be reduced to a single-objective model by associating weights to the individual objective functions and creating a single objective function as the sum

of the weighted objective functions. Another approach for reduction to single-objective model is to minimize the maximum of the deviations of the objective functions from their own individual optimums. Nevertheless, reduction to a single-objective model assumes preferences for the decision maker and returns a single solution based on these preferences. On the other hand, generating a set of non-dominated solutions provides the decision maker with alternative solutions, among which the decision maker can select one. The set of non-dominated solutions is often referred to as the Pareto front and in this volume and we attempt to approximate the Pareto fronts for the SoS architecting problems.

Due to complexity of the SoS architecting problems, we develop evolutionary algorithms for each SoS architecting problem. While these evolutionary algorithms share a common fitness evaluation and termination operations, they differ in the way they represent the SoS architectures and mutate them. Analysis of the SoS architecting problem with inflexible and flexible systems enable us to quantify the benefits of system flexibility in SoS architecting. In particular, upon comparing the Pareto fronts of the SoS architecting problems with inflexible and flexible systems, we note that system flexibility can improve SoS architectures. However, the benefits of system flexibility depend on the level of system flexibility.

In this volume, our contributions are in providing mathematical formulations and developing solution approaches for SoS architecting problem with inflexible and flexible systems. Both the mathematical formulations and the solution algorithms presented are generic, that is, they are easy to modify to capture different settings. Furthermore, we quantitatively demonstrate the benefits of system flexibility and the level of flexibility. While we consider flexibility as the cooperativeness of the systems, different cooperativeness approaches can be modeled using the settings explained in this volume. Specifically, we define flexibility as cooperativeness since the flexible systems cooperate with the SoS architect in re-designing their systems. The level or the willingness of a system to cooperate is associated with a parameter, which we refer to as the incentive charge for being flexible. Next, we explain the details of the mathematical formulations.

SoS Architecting Models with Inflexible and Flexible Systems

Consider a SoS that requires n capabilities and let the capabilities be indexed by i such that $i \in I, I = \{1,2, \dots, n\}$. As noted previously, these capabilities are defined based on the goals/targets of the military mission under consideration. The systems that are equipped with required capabilities and might be included in the SoS are identified using the military inventories. Suppose that there are m systems that can provide the capabilities and let the systems be indexed by j such that $j \in J, J = \{1,2, \dots, m\}$. In particular, each system can provide all or some of the capabilities required and let

$$a_{ij} = \begin{cases} 1: if\ system\ j\ can\ provide\ capability\ i, \\ 0: \qquad\qquad\qquad\qquad otherwise \end{cases}$$

and $\boldsymbol{A}$ be the $n \times m$-matrix of a_{ij} values. Systems have varying characteristics as the system providers distinguish from each other in the engineering of their system designs, the contractors they use for assembling their systems, the properties of the subsystems they utilize, and the

resources they use in their systems. We, therefore, assume that the individual systems have different performance levels for providing the capabilities they can provide due to these varying characteristics. Again, due to these varying characteristics and distinct performance levels, the cost and the integration time for a system to be able to provide a specific capability can be different. Therefore, we assume that the systems have different performance levels, charges, and completion times for providing capabilities. Specifically, let p_{ij}, c_{ij}, and d_{ij} denotes the system j's performance level, charge, and ready-time for providing capability i, respectively, and let $\boldsymbol{P}$, $\boldsymbol{C}$, and $\boldsymbol{D}$ denote the $n \times m$-matrix of p_{ij}, c_{ij}, and d_{ij} values, respectively.

The SoS architect's problem is to construct a fully connected and capable SoS with maximum total performance, minimum completion time, and minimum total cost. A SoS is considered fully connected when any system $j_1 \in J$ included in the SoS can communicate with any other system $j_2 \in J$ included in the SoS. The communication between two systems is achieved through an interface, which has a cost for being integrated into the SoS. Specifically, let $h_{j_1 j_2}$ be the cost of establishing an interface from system j_1 to system j_2. It is assumed that a system can communicate with itself, therefore, $h_{jj} = 0 \; \forall j \in J$. In the case two systems j_1 and j_2 in the SoS are considered communicated when there are an interface from system j_1 to system j_2 and an interface from system j_1 to system j_2, the cost of connecting systems j_1 and $j_2$amounts to $h_{j_1 j_2} + h_{j_2 j_1}$. On the other hand, if two systems j_1 and j_2 in the SoS are considered communicated when there is an interface from system j_1 to system j_2 or an interface from system from system j_2 to system j_1, the cost of connecting systems amounts to $\min\{h_{j_1 j_2}, h_{j_2 j_1}\}$. For both of the cases, the SoS architect should decide on connecting the systems in the SoS. Let

$$y_{j_1 j_2} = \begin{cases} 1: if\ there\ is\ connection\ between \text{ systems } j_1 \text{ and } j_2, & j_1, j_2 \in J, \\ 0: & otherwise \end{cases}$$

and let $\boldsymbol{Y}$ be the $m \times m$-matrix of $y_{j_1 j_2}, j_1, j_2 \in J$ values. Then, the cost of establishing a connection between systems j_1 and j_2 can be defined as $w_{j_1 j_2} = w_{j_2 j_1}$such that $w_{j_1 j_2} = h_{j_1 j_2} + h_{j_2 j_1}$ in the case two interfaces are required (one for communicating system j_1 with system j_2 and one for communicating system j_2 with system j_1) and $w_{j_1 j_2} = \min\{h_{j_1 j_2}, h_{j_2 j_1}\}$ in the case one interface is sufficient for communicating systems j_1 and j_2. It should be remarked that, with this definition of $y_{j_1 j_2}$ values, connecting either system to the other is sufficient for achieving a connection between two systems included in the SoS, therefore, $y_{j_1 j_2} + y_{j_2 j_1} \geq 1$ when both systems j_1 and j_2 are included in the SoS. In formulating the SoS architect's problem, we add constraints assuring the connectedness of the systems included in the SoS.

A SoS is defined to be capable when each capability is provided by at least one system. We formulate the SoS architect's problem with two types of systems: inflexible and flexible. In case of inflexible systems, the systems, who are selected by the SoS architect to be a part of the SoS, contribute to the SoS with all of the capabilities they can provide. That is, the systems (or the system providers) are not collaborative and they cannot or are not willing to change the engineering design of their systems. On the other hand, in case of flexible systems, the SoS architect can guide the systems to provide not necessarily all but some of the capabilities they

can provide. That is, the system providers can modify their system designs as requested by the SoS architect.

The objectives of the SoS architect, however, are the same with inflexible and flexible systems. In particular, Kaplan (2006) notes that agility, performance, and cost are considered by DoD in creating the collection of systems. Therefore, maximization of the total performance and minimizations of the completion time and total cost are used as the SoS architect's objectives (which are the objectives suggested by Pernin et al. (2012) and used by Konur and Dagli (2014) for SoS architecting). While the definitions of the total performance and completion time are similar for SoS architecting with both inflexible and flexible systems, the total cost function is slightly different depending on the system type. In particular, with both system types, the total performance of a SoS is defined as the sum of the performances of the capabilities required in the SoS and, the performance of a capability in the SoS is equal to the sum of the performance levels offered by the systems included in the SoS for providing that capability. The completion time of a SoS is defined as the time required to have every system ready to provide every capability they need to. That is, with both system types, the ready time of a system is the maximum of the times it takes to provide the capabilities it will contribute with and, the completion time of the SoS is the maximum of the included systems' ready times. The total cost of the SoS is equal to the sum of the costs charged by the systems for providing the capabilities plus the connection costs among the included systems in case of inexible systems. In case of exible systems, on the other hand, the SoS architect pays the system providers for changing their system designs. Next, we mathematically formulate the SoS architect's problem with inflexible and flexible systems.

SoS Architecting with Inflexible Systems

When the systems are inflexible, the SoS architect's main decision is to determine which systems to select to be included within the SoS. Let

$$S_j = \begin{cases} 1: if\ system\ j\ is\ selected\ by\ the\ SoS\ architect, \\ 0: \qquad\qquad\qquad\qquad otherwise \end{cases}$$

and let $\boldsymbol{S}$ be the m-vector of S_j values. The SoS's performance for capability i can be defined differently considering various architecting settings. For instance, if the performance of a capability in the SoS is the maximum of the performance levels by the selected systems providing that capability, the SoS's performance for capability i can be defined as $\max_{j \in J} \{S_j a_{ij} p_{ij}\}$. As noted previously, we assume that the performance of a specific capability is the sum of this capability's performance levels provided by the systems included in the SoS. For the settings of this study, this assumption is reasonable as the capabilities define military mission capacities such as attack power, search range, and control, which can be quantified by associated metrics and increase cumulatively with each system's contribution towards the capabilities. The SoS's performance for capability i as a function of S can then be defined as $\sum_j S_j a_{ij} p_{ij}$. At this point, we further assume that performances of different capabilities are additive, therefore, the total performance of the SoS with inflexible systems as a function of S reads

$$TP^1(\boldsymbol{S}) = \sum_{i \in I} \sum_{j \in J} S_j a_{ij} p_{ij} \qquad (1)$$

As discussed by Konur and Dagli (2014), one can modify Equation (1) to capture the cases where performance of different capabilities are of different importance to the SoS architect. In such a case, a weighted approach can be used to modify Equation (1). We note that the solution methods discussed in this study can be easily modified for different functional forms used in defining the total performance as well as capability performances.

The SoS's completion time is defined as the earliest time when all of the selected systems are ready with all of the capabilities they can provide. In particular, when an inflexible system is included in the SoS, the system's ready-time is the time when it is able to provide all of the capabilities it can provide. Considering the definition of d_{ij}, system j's ready-time is equal to $\max_{i \in I}\{a_{ij} d_{ij}\}$. Then, the completion time of the SoS with inflexible systems as a function of S can be defined as

$$TT^1(\boldsymbol{S}) = \max_{i \in I}\left\{\max_{j \in J}\{S_j a_{ij} d_{ij}\}\right\}. \quad (2)$$

In Equation (2), it is assumed that a SoS is complete when all of the systems provide their capabilities. In different architecting settings, one can assume that a SoS is complete whenever there is at least one system providing each capability, i.e., the SoS is capable. In such a case, capability i's ready-time by system j is equal to $\max\{S_j a_{ij} d_{ij}, (1 - a_{ij})M + (1 - S_j)M\}$, where M is a very large number (note that when $S_j = 0$, or $S_j = 1$ but $a_{ij} = 0$, it means that system j takes a very long time to provide capability i, which practically implies that system j is not providing capability i). Then, the earliest ready-time for capability i in the SoS is equal to $\min_{j \in J}\{\max\{S_j a_{ij} d_{ij}, (1 - a_{ij})M + (1 - S_j)M\}\}$. It then follows that the earliest time when all of the required capabilities are ready in the SoS is equal to $\max_{i \in I}\left\{\min_{j \in J}\{\max\{S_j a_{ij} d_{ij}, (1 - a_{ij})M + (1 - S_j)M\}\}\right\}$.

The total cost of the SoS is equal to the sum of the costs of the systems plus the cost of connections among the selected systems. An inflexible system's cost is equal to the sum of the costs charged for providing the capabilities, i.e., system j's cost is $\sum_{i \in I} a_{ij} c_{ij}$. Then, the cost of the systems included in the SoS amounts to $\sum_{i \in I} \sum_{j \in J} S_j a_{ij} c_{ij}$. Note that given $\boldsymbol{S}$, one can determine $\boldsymbol{Y}$ very easily. Particularly, it can be observed that $y_{j_1 j_2} + y_{j_2 j_1} = 1$ if $S_{j_1} + S_{j_2} = 2$; and $y_{j_1 j_2} + y_{j_2 j_1} = 0$ if $S_{j_1} + S_{j_2} \leq 1$. In formulating the SoS architecting problem with inflexible systems, we will include constraints that will assure that the selected systems are connected. Then, the cost of connections among the selected systems amounts to $\sum_{j_1 \in J} \sum_{j_2 \in J} w_{j_1 j_2} y_{j_1 j_2}$. It then follows that the total cost of the SoS with inflexible systems as a function of $\boldsymbol{S}$ and $\boldsymbol{Y}$ reads as

$$TC^1(\boldsymbol{S}, \boldsymbol{Y}) = \sum_{i \in I} \sum_{j \in J} S_j a_{ij} c_{ij} + \sum_{j_1 \in J} \sum_{j_2 \in J} h_{j_1 j_2} y_{j_1 j_2}. \quad (3)$$

The SoS architecting problem with inflexible systems (SoS-I) can then be formulated as follows:

Formulation 1 SoS Architect with Inflexible Systems (SoS-I)

Maximize

$$TP^1(\boldsymbol{S})$$

Minimize

$$TT^1(\boldsymbol{S})$$

$$TC^1(\boldsymbol{S},\boldsymbol{Y})$$

Subject-to

$$\sum_{j\in J} S_j a_{ij} \geq 1 \quad \forall\, i \in I \tag{4}$$

$$y_{rs} + y_{sr} \geq S_r + S_s - 1 \quad \forall\, r,s \in J \tag{5}$$

$$S_j \in \{0,1\} \quad \forall\, j \in J \tag{6}$$

$$y_{rs} \in \{0,1\} \quad \forall\, r,s \in J \tag{7}$$

where $TP^1(\boldsymbol{S})$, $TT^1(\boldsymbol{S})$, and $TC^1(\boldsymbol{S},\boldsymbol{Y})$ are defined in Equations (1), (2), and (3), respectively. Constraints (4) guarantee that each capability is provided by at least one of the systems included in the SoS. Constraints (5) assure that there is a connection between any distinct pair of the selected systems. Note that if $S_j + S_k = 2$, constraints (5) imply that $y_{jk} + y_{kj} \geq 1$; however, since $y_{jk} + y_{kj} = 1$ is sufficient for connectedness and an additional connection between systems j and k increases costs while not changing to the SoS's total performance and completion time, one has either $y_{jk} = 1$ or $y_{kj} = 1$ but not both $y_{jk} = y_{kj} = 1$. Similarly, it can be argued that if $S_j + S_k \leq 1$, $y_{jk} = y_{kj} = 0$. Constraints (6) and (7) give the binary definitions of the decision variables.

SoS Architecting with Flexible Systems

When the systems are exible, the SoS architect's main decision is to determine which systems will be requested to provide which capabilities. Let

$$x_{ij} = \begin{cases} 1\text{: } if\ capability\ i\ is\ requested\ from\ system\ j, \\ 0\text{: } \qquad\qquad\qquad\qquad otherwiswe, \end{cases}$$

and let $\boldsymbol{X}$ be the $n \times m$-matrix of x_{ij} values. Note that by definition of a_{ij}, we have $x_{ij} \leq a_{ij}$. That is, the SoS architect will not request a capability from a system which cannot provide that capability. A system is selected in the SoS architecture if it is asked to provide at least one capability. Let

$$Z_j = \begin{cases} 1\text{: } if \sum_{i\in I} x_{ij} \geq 1\,, \\ 0\text{: } \quad otherwiswe, \end{cases}$$

that is, Z_j is the binary variable indicating selection of system j and let $\boldsymbol{Z}$ be the m-vector of Z_j values. It should be remarked that $\boldsymbol{Z}$ and $\boldsymbol{S}$ are different. In particular, while $\boldsymbol{S}$ is the decision variables vector in case of inflexible systems, $\boldsymbol{Z}$ is the auxiliary decision variables vector, determined by $\boldsymbol{X}$, in case of flexible systems. Nonetheless, the relation between $\boldsymbol{Y}$ and a given $\boldsymbol{S}$ is the same as the relation between $\boldsymbol{Y}$ and a given $\boldsymbol{Z}$. That is, $y_{j_1 j_2} + y_{j_2 j_1} = 1$ if $Z_{j_1} + Z_{j_2} = 2$; and, $y_{j_1 j_2} + y_{j_2 j_1} = 0$ if $Z_{j_1} + Z_{j_2} \leq 1$.

Following the definition of total performance stated in Equation (1), one can note that the total performance of the SoS with flexible systems as a function of $\boldsymbol{X}$ can be stated as follows:

$$TC^1(\boldsymbol{S}, \boldsymbol{Y}) = \sum_{i \in I} \sum_{j \in J} S_j a_{ij} c_{ij} + \sum_{j_1 \in J} \sum_{j_2 \in J} h_{j_1 j_2} y_{j_1 j_2} \quad (8)$$

Similar to Equation (2), the completion time of the SoS with flexible systems as a function of X reads as

$$TT^2(\boldsymbol{SX}) = \max_{i \in I} \left\{ \max_{j \in J} \{x_{ij} d_{ij}\} \right\} \quad (9)$$

The total cost of the SoS with flexible systems is slightly different than the definition of the total cost of the SoS with inflexible systems. In particular, similar to Equation (3), the total cost of the SoS with flexible systems includes the sum of the costs charged by the systems for providing the capabilities they are requested to and the cost of the connections among the selected systems. In addition, the total cost of the SoS with flexible systems includes the incentives paid by the SoS architect to the systems for their collaborations to be flexible. Specifically, when the SoS architect requests only a subset of the capabilities a system can provide, the system provider needs to disassemble the unrequested capabilities and make engineering design changes in its system accordingly. This, of course, is a costly process. Therefore, we assume that the SoS architect is subject to incentive charges e_{ij} for requesting system j not to provide capability i, which would be provided otherwise and let $\boldsymbol{E}$ be the $n \times m$-matrix of e_{ij} values. Then, the total cost of the SoS with flexible systems as a function of $\boldsymbol{X}$ and $\boldsymbol{Y}$ amounts to

$$TC^2(\boldsymbol{X}, \boldsymbol{Y}) = \sum_{i \in I} \sum_{j \in J} x_{ij} c_{ij} + \sum_{i \in I} \sum_{j \in J} (a_{ij} - x_{ij}) e_{ij} + \sum_{j_1 \in J} \sum_{j_2 \in J} h_{j_1 j_2} y_{j_1 j_2} \quad (10)$$

where the first term is the costs charged by the systems for providing the capabilities they are requested to, the second term is the incentive costs paid by the SoS architect to the systems for not providing the capabilities they originally could, and the last term is the cost of connections among the selected systems. Note that when x_{ij}, system j provides capability i; hence, the SoS architect is subject to charges of c_{ij}. On the other hand, if $x_{ij} = 0$ when $a_{ij} = 1$, this means that the SoS architect is requesting system j not to provide capability i; thus, the SoS architect is subject to charges of e_{ij}.

The SoS architecting problem with flexible systems (SoS-F) can then be formulated as follows:

Formulation 2 SoS Architect with Flexible Systems (SoS-F)

Maximize

$$TP^2(\boldsymbol{X}, \boldsymbol{Y})$$

Minimize

$$TT^2(\boldsymbol{X})$$

$$TC^2(\boldsymbol{X}, \boldsymbol{Y})$$

Subject-to

$$x_{ij} \leq a_{ij} \quad \forall i \in I \tag{11}$$

$$\sum_{j \in J} x_{ij} a_{ij} \geq 1 \quad \forall i \in I \tag{12}$$

$$y_{rs} + y_{sr} \geq Z_r + Z_s - 1 \quad \forall r, s \in J \tag{13}$$

$$Z_j \leq \sum_{i \in I} x_{ij} \quad \forall j \in J \tag{14}$$

$$Z_j \geq 1/n \sum_{i \in I} x_{ij} \quad \forall j \in J \tag{15}$$

$$x_{ij} \in \{0,1\} \quad \forall i \in I \, \forall j \in J \tag{16}$$

$$Z_j \in \{0,1\} \quad \forall j \in J \tag{17}$$

$$y_{rs} \in \{0,1\} \quad \forall r, s \in J \tag{18}$$

where $TP^2(\boldsymbol{X})$, $TT^2(\boldsymbol{X})$, and $TC^2(\boldsymbol{X}, \boldsymbol{Y})$ are defined in Equations (8), (9), and (10), respectively. Constraints (11) ensure that the SoS architect can request the capabilities a system can provide. Constraints (12) and (13) are defined similar to constraints (4) and (5), respectively. Constraints (14) and (15) guarantee that a system is selected in the SoS if at least one capability is requested from it; and, not selected otherwise. Particularly, if $\sum_{i \in I} x_{ij} = 0$constraint (7) indicates that $Z_j = 0$ as $Z_j \in \{0,1\}$; and, if $\sum_{i \in I} x_{ij} > 0, 0 < 1/n \sum_{i \in I} x_{ij} \leq 1$; hence, constraint (15) indicates that $Z_j = 1$ as $Z_j \in \{0,1\}$. Constraints (16), (17), and (18) give the binary definitions. Next, we explain the details of the solution methods for SoS-I and SoS-F.

SoS Architecting Algorithms with Inflexible and Flexible Systems

Note that both SoS-I and SoS-F are bi-objective binary-integer non-linear optimization problems. Two common methods for solving multi-objective optimization problems are Pareto front generation (where the decision maker is provided with a set of solutions, among which a solution is selected) and reduction to single-objective formulation (where different weights are assigned to different objectives considering the decision maker's preferences or the maximum deviation from the optimum solution of the individual objectives is minimized and a solution is provided to the decision maker). In this volume, we adopt the former method and approximate the Pareto front (PF) of SoS-I and SoS-F by generating a set of Pareto effcient SoS's for each case. To do so, due to the binary definitions of the decision variables, we propose two evolutionary heuristic

algorithms; one for SoS-I, denoted by EA-I, and one for SoS-F, denoted by EA-F. Both of these algorithms consist of four main steps: (i) chromosome representation and initialization, (ii) fitness evaluation, (iii) mutation, and (iv) termination. Basically, an evolutionary algorithm works as follows. Given a set of solutions (chromosomes), i.e., a population, the best chromosome(s) are selected through fitness evaluation to generate the next population. The best chromosomes of a population constitute the parent chromosomes of the next population. The next population is generated by mutating the parent chromosomes of the current population. These steps are repeated until a certain termination criteria is met. Steps (ii) and (iv) are common in both of the algorithms we propose, while steps (i) and (iii) are different due to the distinct characteristics of SoS-I and SoS-F. We, therefore, first explain the common steps (ii) and (iv), and then, steps (i) and (iii) for each algorithm.

PARETO FRONT APPROXIMATION AND TERMINATION

Let $\boldsymbol{SoS}$ denote a solution for SoS-I or SoS-F and let (TP, TT, TC) be the total performance, completion time, and total cost of $\boldsymbol{SoS}$, respectively. Note that $\boldsymbol{SoS} = (\boldsymbol{S}, \boldsymbol{Y})$ and $(TP, TT, TC) = (TP^1(\boldsymbol{S}), TT^1(\boldsymbol{S}),\ TC^1(\boldsymbol{S}, \boldsymbol{Y}))$ for SoS-I, and $\boldsymbol{SoS} = (\boldsymbol{X}, \boldsymbol{Y})$ and $(TP, TT, TC) = (TP^2(\boldsymbol{X}), TT^2(\boldsymbol{X}), TC^2(\boldsymbol{X}, \boldsymbol{Y}))$ for SoS-F. Now suppose that a set of solutions R is given and let $\boldsymbol{SoS^r} \in R$be the r^{th} solution $r \leq |R|$, such that (TP^r, TT^r, TC^r) defines the total performance, completion time and total cost of $\boldsymbol{SoS^r}$. In fitness evaluation of EA-I and EA-F, the purpose is to select the best chromosomes out of a given population, i.e., the parent chromosomes that will be used in generating the next population. To do so, since both SoS-I or SoS-F are bi-objective optimization problems, we focus on generating the Pareto efficient solutions out of a given population.

A solution is Pareto efficient if it is not Pareto dominated by another solution. Unless $(TP^{r_1}, TT^{r_1}, TC^{r_1}) = (TP^{r_2}, TT^{r_2}, TC^{r_2})$, $\boldsymbol{SoS^{r_1}}$ Pareto dominates $\boldsymbol{SoS^{r_2}}$if $TP^{r_1} \geq TP^{r_2}$, $TT^{r_1} \leq TT^{r_2}$, and $TC^{r_1} \leq TC^{r_2}$ (Berube et al., 2009). Therefore, the following procedure can be used to generate all of the Pareto efficient solutions within a given set of solutions R, denoted by PF(R).

Algorithm 1 Determining PF(R)

Step 0) Set $t = 1$
Step 1) While $t \leq |R| - 1$
Step 2) Set $w = t + 1$
Step 3) While $w \leq |R|$
Step 3.1) If $(TP^t, TT^t, TC^t) \neq (TP^w, TT^w, TC^w), TP^t \geq TP^w, TT^t \leq TT^w,$ and $TC^t \leq TC^w$
Step 3.1.1) Set $R := R \backslash \{\boldsymbol{SoS^w}\}$ and $w := w - 1$
Step 3.2) If $(TP^t, TT^t, TC^t) \neq (TP^w, TT^w, TC^w), TP^t \geq TP^w, TT^t \leq TT^w,$ and $TC^t \leq TC^w$
Step 3.2.1) Set $R := R \backslash \{\boldsymbol{SoS^t}\}$ and $w \coloneqq |R|$ and $t := t - 1$
Step 3.3) Set $w := w + 1$
Step 3.4) Set $t := t + 1$
Step 3.5) Return $PF(R) = R$

Given a population R, PF(R) is taken as the set of parent chromosomes for the next population. If PF(R) is not changing over a pre-specified number of populations, defined as K, in EA-I and EA-F, algorithms are terminated. The latest PF(R) is the set of solutions returned for the decision maker. Next, the details of steps (i) and (iii) for each algorithm are explained.

EVOLUTIONARY ALGORITHM FOR SOS-I

Recall that $\boldsymbol{S}$ is the binary decision variables vector in SoS-I and given $\boldsymbol{S}$, one can easily determine the corresponding $\boldsymbol{Y}$. Therefore, $\boldsymbol{S}$ is sufficient for defining a SoS with inflexible systems. This suggests that $\boldsymbol{S}$ can be used as a chromosome in EA-I. The details of the chromosome representation and initialization and mutation steps of EA-I are as follows.

- **Chromosome Representation and Initialization**: As noted above, the chromosome is defined by $\boldsymbol{S}$. Initially, we generate nm feasible chromosomes. Note that not every binary m-vector is feasible for SoS-I. As suggested by constraints (4), $\boldsymbol{S}$ is feasible when $\sum_{j\in J} S_j a_{ij} \geq 1\ \forall i \in I$. Therefore, in generating the initial population of chromosomes, we first randomly generate a binary m-vector $R = [R_1, R_2, \dots, R_m]$. If $\sum_{j\in J} R_j a_{ij} = 0$ for some $i \in I$, R is infeasible. In case R is infeasible, for each $i \in I$ such that $\sum_{j\in J} R_j a_{ij} = 0$, a system j such that $a_{ij} = 1$ is randomly selected and we set $R_j = 1$. Then, the final R is accepted as a feasible chromosome $\boldsymbol{S}$.
- **Mutation**: Given a set of parent chromosomes, which is achieved by applying Procedure PF on the current population, the next set of chromosomes consists of the parent chromosomes and the newly generated chromosomes through mutation. Including the parent chromosomes within the next population guarantees that the Pareto front is not worsening over populations. New chromosomes are generated by applying a neighbor mutation on each gene of every parent chromosome. The neighbor mutation works as follows. Consider a parent chromosome S and a gene $l \leq m$. If $S_l = 0$, we set $S_l = 1$. Note that with the addition of a new system into a feasible SoS, the SoS will continue to be feasible. Therefore, by setting $S_l = 1$ when $S_l = 0$, a new feasible chromosome is generated. On the other hand, when $S_l = 1$, to avoid infeasibility of the mutant chromosome, we set $S_l = 0$ if $\sum_{j\in J: j\neq l} S_j a_{ij} \geq 1\ \forall i \in I$. That is, we exclude system l from the SoS defined by the current parent chromosome as long as its exclusion does not cause infeasibility by making SoS incapable. Note that one can generate at most m new chromosomes out of a given parent chromosome with the neighbor mutation.

EVOLUTIONARY ALGORITHM FOR SOS-F

Recall that $\boldsymbol{X}$ is the binary decision variables vector in SoS-F. Given $\boldsymbol{X}$, one can determine $\boldsymbol{Z}$ by setting $Z_j = 1$ if $\sum_{i\in I} x_{ij} \geq 1$ and $Z_j = 0$ otherwise. Then, using $\boldsymbol{Z}$, one can determine the corresponding $\boldsymbol{Y}$. That is, $\boldsymbol{X}$ represents a SoS by itself; therefore, we construct EA-F such that it evolves with $\boldsymbol{X}$. The details of the chromosome representation and initialization and mutation steps of EA-F are as follows.

- **Chromosome Representation and Initialization:** We adopt the binary matrix

representation of $\boldsymbol{X}$ as the chromosome. The j[th] column of $\boldsymbol{X}$ defines the j[th] gene of the chromosome. We set the initial population size equal to nm and we generate a feasible chromosome as follows. Note that there are $\sum_{j\in J} a_{ij}$ systems that can provide capability i and, as suggested by constraints (12), $\sum_{j\in J} a_{ij}\, x_{ij} \geq 1$ in a feasible $\boldsymbol{X}$. Therefore, in generating a feasible chromosome, we first randomly generate the number, v_i, between 1 and $\sum_{j\in J} a_{ij}$ to be the number of systems, from which capability i will be requested. Then, for each capability i, we randomly select v_i systems among the systems with $a_{ij} = 1$ and set $x_{ij} = 1$. Repeating this process for each capability, a feasible $\boldsymbol{X}$ is generated.

- **Mutation**: Similar to EA-I, given a set of parent chromosomes, the next set of chromosomes consists of the parent chromosomes and the newly generated chromosomes through mutation to have non-worsening Pareto fronts over populations. New chromosomes are generated by applying two mutations on each gene of every parent chromosome: adding request and dropping request. Consider a parent chromosome $\boldsymbol{X}$, which defines a feasible SoS, and a gene $l \leq m$, representing the l [th] system. Adding request aims at requesting one additional capability from system l, if possible. Specifically, adding request is executed by randomly selecting a capability i from the set of capabilities that can be provided by system l, i.e., $a_{il} = 1$, but currently not requested from system l, i.e., $x_{il} = 0$. Then, in case there exists at least one such capability with $x_{il} = 0$. and $a_{il} = 1$, we set $x_{il} = 1$ for one of such capabilities, which is randomly selected. Dropping request aims at requesting one less capability from system l, if possible. Specifically, dropping request is executed by randomly selecting a capability i from the set of capabilities that are currently requested from system l, i.e., $x_{il} = 1$, such that the SoS remains capable if capability i is not provided by system l. That is, we set $x_{il} = 0$ for a randomly selected capability i from the set of capabilities such that $x_{il} = 1$ and $\sum_{j\in J: j\neq l} x_{ij} a_{ij} \geq 1\ \forall\, i \in I$. One can generate at most $2m$ new chromosomes out of a given parent chromosome.

SoS Architecting Analyses with Inflexible and Flexible Systems

In this section, we conduct a numerical study to analyze (i) the benefits of SoS architecting with flexible systems compared to SoS architecting with in inflexible systems and (ii) the effects of flexibility levels on the SoS architectures. To do so, we consider different problem sizes and different problem classes. In particular, we consider 9 different problem sizes, each of which corresponds to a combination of $n = \{5,10,15\}$ and $m = \{5,10,15\}$. Given n and m, we randomly generate 10 problem instances assuming that $p_{ij} \sim U[10,20]$, $d_{ij} \sim U[5,10]$, $c_{ij} \sim U[20,40]$, and $h_{rs} \sim U[1,5]$, where $U[a,b]$ denotes the continuous uniform distribution with range $[a,b]$. Furthermore, for each problem instance, we randomly generate the binary $\boldsymbol{A}$ matrix such that $\sum_{j\in J} a_{ij} \geq 1\ \forall\, i \in I$, i.e., there is at least on system who can provide each capability (this ensures that the problem instance is feasible).

A given problem instance **{A, P, D, C}** is solved four times assuming inflexibility, low flexibility, medium flexibility, and high flexibility. When inflexibility is assumed, SoS-I is solved using EA-I

with the given **{A, P, D, C}**. In case of flexible systems, we define three levels of flexibility by considering the relation between c_{ij} and e_{ij} values. Recall that c_{ij} is system j's charge for providing capability i and e_{ij} is the charge for asking system j not to provide capability i. In generating e_{ij} values, we assume $e_{ij} = b_{ij}c_{ij}$, where b_{ij} is used to define the level of flexibility. We assume that $b_{ij} \sim U[0,1]$ as it is reasonable and practical to assume that $c_{ij} \leq e_{ij}$ since c_{ij} includes the cost of the capability and its assembly to the system while e_{ij} is the cost of disassembly of the capability from the system. Note that when b_{ij}values are low, system j can be accepted as highly flexible. On the other hand, large b_{ij}values indicate that system j is less flexible. Then, for a given problem instance **{A, P, D, C}**, its flexible versions **{A, P, D, C, E}** with low, medium, and high flexibility are defined by assuming $b_{ij} \sim U[0.6,0.8]$ $b_{ij} \sim U[0.4,0.6]$, and $b_{ij} \sim U[0.2,0.4]$, respectively. When there is flexibility, SoS-F is solved using EA-F with the given **{A, P, D, C, E}**. Let PF^I denote the Pareto front returned by EA-I in case of inflexibility and let PF^{Fl} , PF^{Fm} , and PF^{Fh} denote the Pareto fronts returned by EA-F in cases of low, medium, and high flexibilities, respectively.

We generate 10 problem instances for each problem size and solve it four times as explained above. All of the algorithms are coded in Matlab 2014 and executed on a personal computer with 3Ghz processor and 8GB RAM. For different problem sizes and flexibility levels considered, Tables 3.1, 3.2, 3.3, and 3.4 summarize the average values over all 10 problem instances solved for the following statistics of EA-I and EA-F: the number of populations evaluated (pop. #), the number of SoS's evaluated per population (i.e., the average population size, denoted as |R|), the number of Pareto efficient SoS's per population (i.e., the parent size, denoted as |PF(R)|), the number of Pareto efficient SoS's returned at termination (denoted as $|PF^I|$, $|PF^{Fl}|$, $|PF^{Fm}|$, and $|PF^{Fh}|$ for inflexibility, low flexibility, medium flexibility, and high flexibility cases, respectively), and the computational time in seconds (CPU).

As can be seen in the tables, on average, inflexible problem instances can be solved in less computational time with EA-I compared to the flexible problem instances solved with EA-F. Furthermore, one can note that flexible problem instances have more solutions within their Pareto fronts on average. These results are expected as the flexible problem instances have larger solution sets compared to the inflexible problem instances. In particular, while there are 2^m possible SoS's (including feasible and infeasible solutions) for SoS-I, there are 2^{nm} possible SoS's for SoS-F. Finally, as expected, as the problem size gets larger, the number of populations evaluated, the number of SoS's evaluated per population, the size of the Pareto front, and the computational time increase.

Table 2 Computational Statistics of EA-I for Inflexibility

		Inflexibility				
n	m	pop. #	|R|	|PF (R)|	$\|PF^{I}\|$	CPU
5	5	6.5	4.9	2.6	3.0	0.00
	10	11.4	71.0	16.2	20.3	0.09
	15	16.7	576.9	72.7	96.7	3.21
10	5	6.1	3.7	3.1	3.7	0.00
	10	8.5	83.0	27.8	36.1	0.09
	15	13.4	790.8	125.7	170.5	3.34
15	5	6.0	3.3	3.3	3.9	0.00
	10	8.5	85.4	37.8	49.0	0.10
	15	11.8	813.3	145.0	195.3	3.03
avg.		9.9	270.3	48.2	64.3	1.10

Table 3 Computational Statistics of EA-F for Low Flexibility

		Low Flexibility				
n	m	pop. #	|R|	|PF (R)|	$\|PF^{Fl}\|$	CPU
5	5	4.4	7.0	5.1	5.2	0.02
	10	8.2	63.8	25.9	28.4	1.32
	15	11.1	244.0	64.4	78.3	21.19
10	5	7.1	38.2	19.3	21.5	0.51
	10	13.1	241.3	67.9	82.5	14.31
	15	16.8	940.4	166.6	211.6	196.10
15	5	8.5	39.8	19.5	21.2	0.28
	10	16.2	594.1	126.4	157.3	68.17
	15	23.5	2454.3	328.0	417.0	1539.03
avg.		12.1	513.7	91.5	113.7	204.55

Table 4 Computational Statistics of EA-F for Medium Flexibility

		Medium Flexibility				
n	m	pop. #	|R|	|PF (R)|	$\|PF^{Fm}\|$	CPU
5	5	4.8	8.7	6.4	6.7	0.04
	10	7.0	53.2	24.3	27.6	0.76
	15	10.0	242.9	65.4	81.2	12.22
10	5	7.0	44.5	21.0	23.9	0.88
	10	12.5	332.4	89.2	112.7	28.94
	15	17.5	1073.8	193.1	251.0	251.60
15	5	8.0	46.5	21.5	24.5	0.35
	10	17.2	624.5	139.4	173.3	85.02
	15	23.1	3499.7	458.1	610.1	3494.34
avg.		11.9	658.5	113.1	145.7	430.46

Table 5 Computational Statistics of EA-F for High Flexibility

		High Flexibility				
n	m	pop. #	|R|	|PF (R)|	$\|PF^{Fh}\|$	CPU
5	5	4.9	8.5	6.3	6.5	0.04
	10	7.7	59.5	26.2	30.0	1.36
	15	10.5	298.7	78.0	97.3	21.57
10	5	6.5	35.2	19.3	22.2	0.46
	10	12.5	312.4	87.0	108.9	26.08
	15	17.2	1312.1	226.4	298.7	423.28
15	5	9.5	53.7	25.9	29.0	0.60
	10	17.8	706.3	158.7	203.9	110.52
	15	24.4	3236.3	423.8	565.7	2933.12
avg.		12.3	669.2	116.8	151.4	390.78

Next, we focus on analyzing the effects of flexibility and flexibility incentive charges on the SoS architectures. For each analysis, we quantitatively and qualitatively compare the Pareto fronts of the same problem instance with different flexibility levels as follows. Consider two Pareto fronts PF^I and PF^F. For quantitative comparison, we focus on the size of the Pareto fronts and the percentage of the problem instances where the Pareto fronts have the same size or one is greater than the other, i.e., the percentage of the problem instances where $|PF^I| > |PF^F|$, $|PF^I| = |PF^F|$, and $|PF^I| < |PF^F|$. For qualitative comparison, we focus on comparing the solutions within the Pareto fronts. In particular, we determine whether PF^I dominates PF^F or vice versa. To determine the dominance between PF^I and PF^F, we use the following definition:

> Unless $PF^I \equiv PF^F$, PF^IPareto dominates PF^F, denoted as $PF^I \gg PF^F$, if $PF^U = PF^I$, where $PF^U = PF(PF^I \cup PF^F)$. That is, PF^F includes no solution that Pareto dominates any solution in PF^I.

Procedure PF can be used to determine the set of the Pareto efficient SoS architectures within the union set of the Pareto efficient SoS architectures with inflexibility and flexibility, i.e., $PF^U = PF(PF^I \cup PF^F)$. For qualitative comparison, we document the percentage of the problem instances where $PF^I \equiv PF^F$ (i.e., each solution in one set has a matching solution in the other in terms of objective function values), $PF^I \sim PF^F$ (i.e., neither PF^I dominates PF^F nor PF^F dominates PF^I), $PF^I \gg PF^F$ (i.e., PF^I dominates PF^F), and $PF^I \ll PF^F$ (i.e., PF^F dominates PF^I). Furthermore, we compare the percentage of the Pareto efficient SoS architectures in PF^I as well as in PF^U and the percentage of the Pareto efficient SoS architectures in PF^F as well as in PF^U.

EFFECTS OF FLEXIBILITY

To analyze the effects of flexibility, we compare inflexibility to low, medium, and high flexibilities quantitatively and qualitatively. Table 6 summarizes the quantitative comparison results. Similar to Table 7, it can also be observed that flexibility results in more Pareto efficient SoS architectures compared to inflexibility on average. Furthermore, it can be observed that as the level of

flexibility increases, the percentage of the problem instances where flexibility results in more SoS architectures increases while the percentage of the problem instances where inflexibility results in more SoS architectures tends to decrease. In particular, one can note that, on average, for 88%, 89%, and 90% of the problem instances, inflexibility results in less Pareto efficient SoS architectures compared to flexibility with low, medium, and high levels of flexibility, respectively, and, for 10%, 9%, and 9% of the problem instances, inflexibility results in more Pareto efficient SoS architectures compared to flexibility with low, medium, and high levels of flexibility, respectively.

Table 6 Quantitative Comparison of Inflexibility to Flexibility

		Inflexibility vs. Low Flexibility			Inflexibility vs. Medium Flexibility			Inflexibility vs. High Flexibility		
		$\|PF^I\| > \|PF^I\| <$	$\|PF^I\| =$		$\|PF^I\| >$	$\|PF^I\| =$		$\|PF^I\| <$	$\|PF^I\| >$	$\|PF^I\| =$
n	m	$\|PF^{FL}\|$	$\|PF^{FL}\|$	$\|PF^{FL}\|$	$\|PF^{Fm}\|$	$\|PF^{Fm}\|$	$\|PF^{Fm}\|$	$\|PF^{Fh}\|$	$\|PF^{Fh}\|$	$\|PF^{Fh}\|$
5	5	0%	10%	90%	0%	10%	90%	0%	10%	90%
	10	0%	0%	100%	0%	10%	90%	20%	0%	80%
	15	50%	10%	40%	60%	0%	40%	50%	0%	50%
10	5	0%	0%	100%	0%	0%	100%	0%	0%	100%
	10	0%	0%	100%	0%	0%	100%	0%	0%	100%
	15	30%	0%	70%	20%	0%	80%	10%	0%	90%
15	5	0%	0%	100%	0%	0%	100%	0%	0%	100%
	10	10%	0%	90%	0%	0%	100%	0%	0%	100%
	15	0%	0%	100%	0%	0%	100%	0%	0%	100%
avg.		10.0%	2.2%	87.8%	8.9%	2.2%	88.9%	8.9%	1.1%	90.0%

As expected, flexibility of the systems results in more Pareto efficient SoS architectures. However, quantitative comparison does not indicate better SoS architectures due to flexibility. Therefore, we compare the Pareto fronts of the problem instances solved with inflexibility, and different levels of flexibilities qualitatively. Table 7 presents the results of the qualitative comparison of inflexibility to flexibility for different levels of flexibilities.

Table 7 Qualitative Comparison of Inflexibility to Flexibility

		Inflexibility vs Low Flexibility						
n	m	$PF^I \equiv PF^{Fl}$	$PF^I \gg PF^{Fl}$	$PF^I \ll PF^{Fl}$	$PF^I \sim PF^{Fl}$	$\lvert PF^U \rvert$	% of PF^I in PF^U	% of PF^{Fl} in PF^U
5	5	0%	0%	10%	90%	6.3	40%	72%
	10	0%	0%	0%	100%	37.8	53%	51%
	15	0%	0%	0%	100%	150.9	63%	39%
10	5	0%	0%	0%	100%	22.2	24%	80%
	10	0%	0%	0%	100%	107.2	39%	62%
	15	0%	0%	0%	100%	355.1	48%	52%
15	5	0%	0%	0%	100%	21.6	18%	85%
	10	0%	0%	0%	100%	198.3	27%	73%
	15	0%	0%	0%	100%	586.5	34%	66%
avg.		0%	0%	1%	99%	165.1	39%	64%

		Inflexibility vs Medium Flexibility						
n	m	$PF^I \equiv PF^{Fm}$	$PF^I \gg PF^{Fm}$	$PF^I \ll PF^{Fm}$	$PF^I \sim PF^{Fm}$	$\lvert PF^U \rvert$	% of PF^I in PF^U	% of PF^{Fm} in PF^U
5	5	0%	0%	10%	90%	7.6	36%	75%
	10	0%	0%	0%	100%	37.9	52%	52%
	15	0%	0%	0%	100%	151.4	60%	41%
10	5	0%	0%	0%	100%	24.8	23%	81%
	10	0%	0%	0%	100%	134.0	34%	67%
	15	0%	0%	0%	100%	385.8	43%	57%
15	5	0%	0%	0%	100%	23.9	17%	86%
	10	0%	0%	0%	100%	208.8	25%	75%
	15	0%	0%	0%	100%	759.7	28%	72%
avg.		0%	0%	1%	99%	192.7	35%	67%

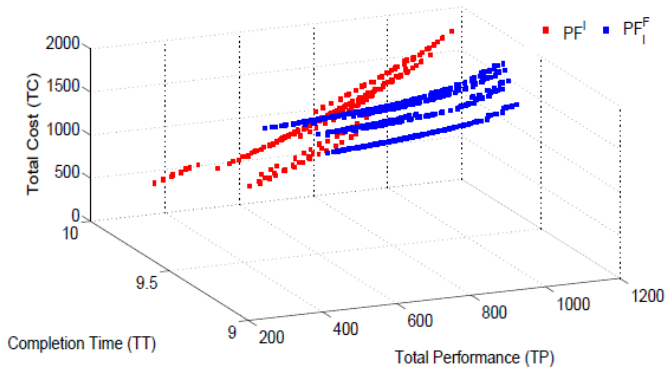
		Inflexibility vs High Flexibility						
n	m	$PF^I \equiv PF^{Fh}$	$PF^I \gg PF^{Fh}$	$PF^I \ll PF^{Fh}$	$PF^I \sim PF^{Fh}$	$\lvert PF^U \rvert$	% of PF^I in PF^U	% of PF^{Fh} in PF^U
5	5	0%	0%	10%	90%	7.8	36%	76%
	10	0%	0%	0%	100%	40.8	51%	53%
	15	0%	0%	0%	100%	165.8	56%	46%
10	5	0%	0%	0%	100%	23.4	20%	83%
	10	0%	0%	0%	100%	130.6	33%	68%
	15	0%	0%	0%	100%	425.8	40%	60%
15	5	0%	0%	0%	100%	30.2	14%	88%
	10	0%	0%	0%	100%	240.1	21%	79%
	15	0%	0%	0%	100%	714.6	28%	72%
avg.		0%	0%	1%	99%	197.7	33%	69%

We have the following observations based on Table 7:

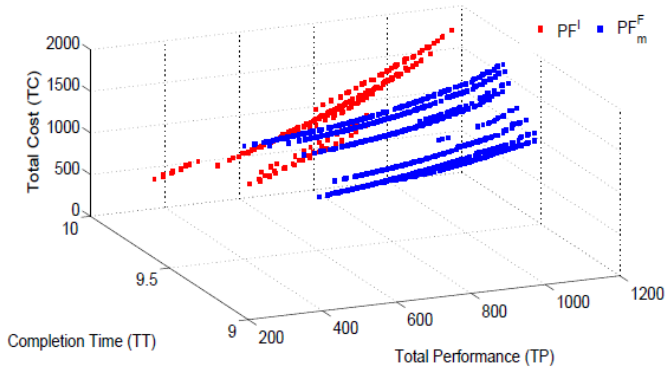
- In all of the problem instances solved, there was no instance where $PF^I \gg PF^F$. For one problem instance with n = 5, m = 5, $PF^I \ll PF^F$. For most of the problem instances (99% on average), there was no clear dominance between the Pareto front of the inflexibility compared to the Pareto fronts of the flexibilities. This result is expected as the SoS architect regards three objectives in his/her SoS architecting problem. In particular, compared to flexibility, if inflexibility results in higher costs and/or higher completion times, it also results in higher performance of the SoS; therefore, Pareto fronts of the flexibilities cannot dominate the Pareto front of the inflexibility.
- On the other hand, when the percentages of the Pareto efficient SoS architectures being included within the Pareto efficient SoS architectures of the union of the Pareto fronts are compared, one can observe that the flexibilities return more solutions than inflexibility, which are included in PF^U. This suggests that flexibility of the systems is able to generate more Pareto dominating solutions compared to the Pareto efficient solutions with inflexibility. Furthermore, as the level of flexibility increases, SoS architecting with flexible (inflexible) systems results in more (less) Pareto efficient solutions that are still Pareto efficient compared to the Pareto efficient solutions of the SoS architecting with inflexible (flexible) systems.

Figure 7 illustrates the Pareto front with inflexible systems to the Pareto front with flexible systems with low, medium, and high flexibilities for a problem instance with n = 15 and m = 15 (similar figures are observed for all problem instances solved). As can be noted in Figure 1 as well, SoS architecting with inflexible systems results in higher completion times and costs for similar performance levels of SoS architecting with flexible systems; however, SoS architecting with inflexible systems can reduce costs further than SoS architecting with flexible systems at expenses of high completion times and low performances. These observations suggest that system flexibility offers benefits for SoS architecting by providing more and better SoS architectures. Nevertheless, this does not mean that SoS architecting with inflexible systems will be dominated by SoS architecting with flexible systems.

(a) PF^I vs. PF_l^F



(b) PF^I vs. PF_m^F



(c) PF^I vs. PF_h^F

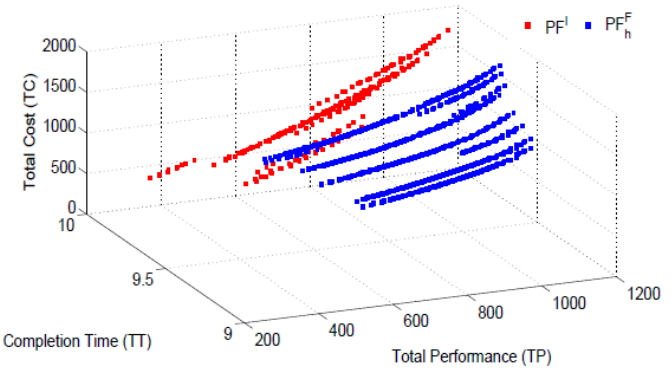


Figure 7 Comparison of the Pareto Fronts with Inflexibility and Flexibility: n = 15, m = 15

EFFECTS OF FLEXIBILITY LEVELS

To analyze the effects of flexibility levels, we compare different flexibility levels qualitatively. Table 8 summarizes the qualitative comparison results. In particular, Table 8 documents the qualitative comparison of SoS architecting with low flexibility to medium flexibility, SoS

architecting with low flexibility to high flexibility, and SoS architecting with medium flexibility to high flexibility. We have the following observations based on Table 8:

- In all of the problem instances solved, there was no instance where Pareto front with a lower flexibility level dominated the Pareto front with a higher flexibility level. For one problem instance with n = 5, m = 5, Pareto front with higher flexibility level dominated the Pareto front with lower flexibility. For most of the problem instances (99% on average), there was no clear dominance between the Pareto fronts of different flexibility levels. These results follow from the fact that as flexibility increases, the SoS architect is able to lower costs and completion times at an expense of lower performances.
- On the other hand, when the percentages of the Pareto efficient SoS architectures being included within the Pareto efficient SoS architectures of the union of the Pareto fronts are compared, one can observe that the higher flexibility levels return more solutions that lower flexibility levels, which are included in PF^U. This suggests that higher flexibility of the systems is able to generate more Pareto dominating solutions compared to the Pareto efficient solutions with lower flexibility. Furthermore, as the level of flexibility increases, SoS architecting with higher flexibility results in more Pareto efficient solutions that are still Pareto efficient compared to the Pareto efficient solutions of the SoS architecting with lower flexibility systems.

Figure 8 illustrates the Pareto fronts with different flexibilities for a problem instance with n = 15 and m = 15 (similar figures are observed for all problem instances solved). As can be noted in Figure 8 as well, SoS architecting with higher flexibility improves completion times and performances with similar costs. Nevertheless, SoS architecting with lower flexibility can result in higher performance at an expense of increased costs and completion times. These observations suggest that higher system flexibility offers benefits for SoS architecting by providing more and better SoS architectures. Nevertheless, this does not mean that SoS architecting with lower flexibility will be dominated by SoS architecting with higher flexibilities.

Table 8 Qualitative Comparison of Flexibility Levels

		Low Flexibility vs. Medium Flexibility						
n	m	$PF^{Fl} \equiv PF^{Fm}$	$PF^{Fl} \gg PF^{Fm}$	$PF^{Fl} \ll PF^{Fm}$	$PF^{Fl} \sim PF^{Fm}$	$\lvert PF^U \rvert$	% of PF^{Fl} in PF^U	% of PF^{Fm} in PF^U
5	5	10%	0%	0%	90%	7.9	36.6%	99.1%
	10	0%	0%	0%	100%	28.7	9.6%	99.5%
	15	0%	0%	0%	100%	88.3	7.0%	96.5%
10	5	0%	0%	0%	100%	25.1	15.2%	99.8%
	10	0%	0%	0%	100%	116.5	4.3%	97.9%
	15	0%	0%	0%	100%	258.2	3.6%	97.3%
15	5	0%	0%	0%	100%	25.5	9.2%	100.0%
	10	0%	0%	0%	100%	176.7	2.8%	98.6%
	15	0%	0%	0%	100%	644.7	5.7%	94.7%
avg.		1%	0%	0%	99%	152.4	10.4%	98.2%

Low Flexibility vs. High Flexibility

n	m	$PF^{Fl} \equiv PF^{Fl} \sim PF^{Fh}$	$PF^{Fl} \gg PF^{Fh}$	$PF^{Fl} \ll PF^{Fh}$	$\lvert PF^{U} \rvert$	% of PF^{Fl} in PF^{U}	% of PF^{Fh} in PF^{U}
5	5	10%	0%	90%	7.5	36.2%	100.0%
	10	0%	0%	100%	31.0	9.5%	100.0%
	15	0%	0%	100%	102.7	6.1%	97.0%
10	5	0%	0%	100%	24.2	16.3%	98.8%
	10	0%	0%	100%	112.4	3.6%	98.7%
	15	0%	0%	100%	305.0	3.1%	97.6%
15	5	0%	0%	100%	30.0	8.3%	100.0%
	10	0%	0%	100%	208.8	2.8%	98.3%
	15	0%	0%	100%	601.9	5.5%	94.9%
avg.		1%	0%	99%	158.2	10.2%	98.4%

Medium Flexibility vs. High Flexibility

n	m	$PF^{Fm} \equiv PF^{Fm} \sim PF^{Fh}$	PF^{F}	$PF^{Fm} \gg PF^{Fh}$	$PF^{Fm} \ll PF^{Fh}$	$\lvert PF^{U} \rvert$	% of PF^{Fm} in PF^{U}	% of PF^{Fh} in PF^{U}
5	5	10%	0%	0%	90%	7.5	36.2%	100.0%
	10	0%	0%	0%	100%	31.5	11.4%	97.8%
	15	0%	0%	0%	100%	101.1	6.2%	96.9%
10	5	0%	0%	0%	100%	24.0	16.1%	99.0%
	10	0%	0%	0%	100%	116.0	8.2%	94.0%
	15	0%	0%	0%	100%	315.4	8.3%	92.4%
15	5	0%	0%	0%	100%	30.0	8.3%	100.0%
	10	0%	0%	0%	100%	209.3	3.0%	98.1%
	15	0%	0%	0%	100%	585.2	3.3%	97.1%
avg.		1%	0%	0%	99%	157.8	11.2%	97.3%

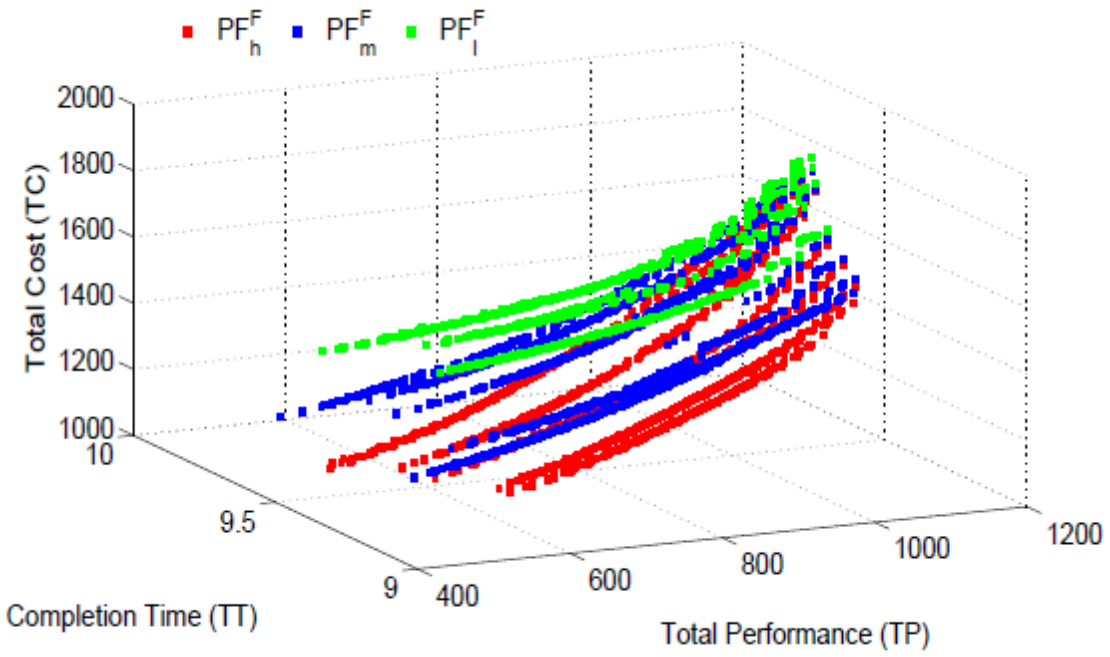


Figure 8 Comparison of the Pareto Fronts with Different Flexibilities: n = 15, m = 15

CONCLUDING REMARKS

In this volume, operations research tools are used to analyze the effects of system flexibility in System of Systems (SoS) architecting. SoS architecting finds many practical applications, especially, in defense/military projects. We formulated multi-objective optimization models for SoS architecting problem with inflexible and flexible systems. Due to the complexity of the models, evolutionary algorithms are developed to generate a set of Pareto efficient solutions for the SoS architecting problems. The models and the solution methods proposed are generic in the sense that they can be easily modified to capture different SoS architecting settings.

In particular, formulations and solution methods proposed for SoS architecting models with inflexible and flexible systems enable investigation of the potential benefits of system flexibility in SoS architecting. Flexibility is considered as the level of cooperativeness between the SoS architect and the systems as flexible systems can cooperate with the SoS architect since the SoS architect can guide flexible systems for modifying their systems. Through a numerical study, it is observed that system flexibility can provide the SoS architect with more alternative SoS architectures and most of these alternatives will be better compared to the alternative SoS architectures generated in case of inflexible systems. Nevertheless, due to the consideration of more than two objectives in the SoS architecting problem, one cannot clearly say that system flexibility will always improve all objectives considered. System flexibility can improve completion times and costs at an expense of reduced performance. This insight follows as flexible systems does not provide some of the capabilities they can, which reduces costs and the system's ready-

time (and thereby overall costs and completion time of the SoS) as well as the overall performance since the capability's cumulative performance decreases. Further numerical study suggests that higher flexibility levels may offer SoS architecting benefits. Similar to the comparison of inflexibility to flexibility, it is observed that when systems have higher flexibilities, the SoS architect can benefit from SoS architectures with lower costs and higher performances with similar costs (or one can equivalently say that lower completion times and costs with similar performances).

This volume contributes to the literature by analyzing SoS architecting with different types of flexibility using operations research tools. Future research directions include SoS architecting with adjustable flexibility levels. In this study, we assumed that the incentive charges for flexibility are fixed; however, it is possible that incentive charges vary and depending on the incentive charges, the systems' flexibility levels can change. Another future research direction is to analyze the effects of flexible systems in case of stochastic SoS architecting problems. Specifically, robust SoS architecting with inflexible and flexible systems is an open research area.

APPENDIX A: LIST OF PUBLICAS RESULTED AND PAPERS SUBMITTED FROM FILA-SOS RESEARCH

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)*.

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

Agarwal, S., & Dagli, C. H. (2013). Augmented Cognition in Human–System Interaction through Coupled Action of Body Sensor Network and Agent Based Modeling. *Procedia Computer Science*, *16*, 20-28.

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.

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., 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., 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., 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.

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.

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.

Acheson, P., Dagli, C., & Kilicay-Ergin, N. (2014). Fuzzy Decision Analysis in Negotiation between the System of Systems Agent and the System Agent in an Agent-Based Model. *arXiv preprint arXiv:1402.0029*.

Kilicay-Ergin, N. H., Acheson, P., Colombi, J. M., & Dagli, C. H. (2012). Modeling system of systems acquisition. In *SoSE* (pp. 514-518).

Acheson, P., Pape, L., Dagli, C., Kilicay-Ergin, N., Columbi, J., & Haris, K. (2012). Understanding System of Systems Development Using an Agent-Based Wave Model. *Procedia Computer Science*, *12*, 21-30.

Konur, D., & Dagli, C. (2014). Military system of systems architecting with individual system contracts*. Optimization Letters*, 1-19.

Dagli et al., 2015 Flexible and Intelligent Learning Architectures for SoS (FILA-SoS): Architectural evolution in Systems-of-Systems, 2015 Conference on Systems Engineering Research.

Ergin, D., & Dagli, C., Incentive Based Negotiation Model for System of Systems Acquisition. (Accepted by Systems Engineering Journal)

Wang, R., & Dagli, C., Search Based Systems Architecture Development Using Holistic Approach (Accepted to IEEE Systems Journal with minor revisions)

APPENDIX B: CITED AND RELATED REFERENCES

Abo-Sinna, M. A., & Baky, I. A. (2007). Interactive Balance Space Approach for Solving Multi-Level Multi-Objective Programming Problems. *Information Sciences, 177(16)*, 3397-3410.

Abraham, A., & Jain, L. (2005). *Evolutionary Multi-objective Optimization* (pp. 1-6). Springer London.

Acheson, P. (2010, April). Methodology for Object-Oriented System Architecture Development. In *Systems Conference, 2010 4th Annual IEEE* (pp. 643-646). IEEE.

Acheson, P., Dagli, C., & Kilicay-Ergin, N. (2014). Fuzzy Decision Analysis in Negotiation between the System of Systems Agent and the System Agent in an Agent-Based Model. *arXiv preprint arXiv:1402.0029*.

Acheson, P., Pape, L., Dagli, C., Kilicay-Ergin, N., Columbi, J., & Haris, K. (2012). Understanding System of Systems Development Using an Agent-Based Wave Model. *Procedia Computer Science*, *12*, 21-30.

Adams, K. M., & Meyers, T. J. (2011). The US Navy carrier strike group as a system of systems. *International Journal of System of Systems Engineering*, 2(2), 91-97.

Agarwal, S., & Dagli, C. H. (2013). Augmented Cognition in Human–System Interaction through Coupled Action of Body Sensor Network and Agent Based Modeling. *Procedia Computer Science*, *16*, 20-28.

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.

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., 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., 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., 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.

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*.

Alberts, D. S., (2011). *The Agility Advantage: A Survival Guide for Complex Enterprises and Endeavors*. Washington DC: Center for Advanced Concepts and Technology.

Alberts, D. S., Garstka, J. J., & Stein, F. P. (1999). {Network Centric Warfare: Developing and Leveraging Information Superiority}.

Alfaris, A. A. F. (2009). *The Evolutionary Design Model (EDM) for the Design of Complex Engineered Systems: Masdar City as a Case Study* (Doctoral dissertation, Massachusetts Institute of Technology).

Alves, M. J., Dempe, S., & Júdice, J. J. (2012). Computing the Pareto Frontier of a Bi-objective Bi-level Linear Problem using a Multi-objective Mixed-integer Programming Algorithm. *Optimization*, *61*(3), 335-358.

Amberg, M. (1996, October). Modeling Adaptive Workflows in Distributed Environments. In *Proc. of the 1st Int. Conf. on Practical Aspects of Knowledge Management, Basel, 30th-31th Oct*.

Anandalingam, G., & Friesz, T. L. (1992). Hierarchical Optimization: an Introduction. *Annals of Operations Research*, *34*(1), 1-11.

ASD(NII), D. (2010). *DoD Architecture Framework Version 2.02 (DoDAF v2.02)*. Washington DC: Department of Defense.

AT&L, O. U. S. D. (2008). Systems Engineering Guide for Systems of Systems. *Washington, DC: Pentagon*.

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

Bac, M., & Raff, H. (1996). Issue-by-issue Negotiations: the Role of Information and Time Preference. *Games and Economic Behavior*, *13*(1), 125-134.

Baky, I. A. (2009). Fuzzy Goal Programming Algorithm for Solving De-centralized Bi-level Multi-objective Programming Problems. *Fuzzy Sets and Systems*, *160*(18), 2701-2713.

Baky, I. A. (2010). Solving Multi-level Multi-objective Linear Programming Problems through Fuzzy Goal Programming Approach. *Applied Mathematical Modelling*, *34*(9), 2377-2387.

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.

Blanchard, B. S., & Fabryeky, W. J. (2010). *Systems Engineering and Analysis*. Upper Saddle River, NJ: Prentice Hall.

Bonabeau, E. (2002). Agent-based modeling: Methods and Techniques for Simulating Human Systems. *Proceedings of the National Academy of Sciences of the United States of America*, *99 (Suppl 3)*, 7280-7287.

Bouleimen, K. L. E. I. N., & Lecocq, H. O. U. S. N. I. (2003). A New Efficient Simulated Annealing Algorithm for the Resource-constrained Project Scheduling Problem and Its Multiple Mode Version. *European Journal of Operational Research*, *149*(2), 268-281.

Bradley, S. P., Hax, A. C., & Magnanti, T. L. Applied Mathematical Programming. 1977.

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.

Cara, A. B., Wagner, C., Hagras, H., Pomares, H., & Rojas, I. (2013). Multi-objective Optimization and Comparison of Non-singleton type-1 and Singleton interval type-2 Fuzzy Logic Systems. *Fuzzy Systems, IEEE Transactions on*, *21*(3), 459-476.

Chattopadhyay, D., Ross, A. M., & Rhodes, D. H. (2008, April). A Framework for Trade-space Exploration of Systems of Systems. In *6th Conference on Systems Engineering Research*, Los Angeles, CA.

Christian III, J. A. (2004). *A Quantitative Approach to Assessing System Evolvability*. Houston: NASA Johnson Space Center.

CJCSI 6212.01F. (12 Mar 2012). *Net Ready Key Performance Parameter (NR KPP).* Washington DC: US Dept of Defense.

Clouthier, R. J., Diamrio, M. J., & Polzer, H. W. (2009). Net Centricity and System of Systems. In M. Jamshidi, *System of Systems Engineering* (pp. 150-168). Hoboken NJ: John Wiley & Sons.

Clune, J., Mouret, J. B., & Lipson, H. (2013). The Evolutionary Origins of Modularity. *Proceedings of the Royal Society b: Biological sciences*, *280 (1755)*, 20122863.

Coello, C. A. C., & Lamont, G. B. (2004). *Applications of Multi-objective Evolutionary Algorithms* (Vol. 1). World Scientific.

Cohon, J. L. (1985). Multi-criteria Programming: Brief Review and Application. *Design optimization*, 163.

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*.

Contag, G., Laing, C., Pabon, J., Rosenberg, E., Tomasino, K., & Tonello, J. (2013). Nighthawk System Search and Rescue (SAR) Unmanned Vehicle (UV) System Development. *SE4150 Design Project, Naval Postgraduate School*.

Cox Jr., L. A. (2009). *Risk Analysis of Complex and Uncertain Systems (Vol. 129)*. US: Springer-Verlag.

Crossley, W. A., & Laananen, D. H. (1996). Conceptual Design of Helicopters via Genetic Algorithm. *Journal of Aircraft*, *33*(6), 1062-1070.

Dagli, C. H., & Kilicay-Ergin, N. (2008). System of Systems Architecting. Jamshidi, M. (Ed.). In *System of Systems Engineering: Innovations for the Twenty-first Century (Vol. 58). John Wiley & Sons.*, 77-100.

Dagli, C., Ergin, N., Enke, D., Gosavi, A., Qin, R., Colombi, J.,Agarwal,S., ... & 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 UNIV OF SCIENCE AND TECHNOLOGY ROLLA.

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.

Dahmann, J., Lane, J., Rebovich, G., & Baldwin, K. (2008, April). A Model of Systems Engineering in a System of Systems Context. In *Proceedings of the Conference on Systems Engineering Research, Los Angeles, CA, USA*.

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. (2012). INCOSE SoS Working Group Pain Points. In *Proc TTCP-JSA-TP4 Meeting*. Retrieved from http://www.ndia.org/Divisions/Divisions/SystemsEngineering/Documents/NDIA-SE-MS-SoS_2013-08-20_Dahmann.pdf

Dauby, J. P., & Dagli, C. H. (2011). The Canonical Decomposition Fuzzy Comparative Methodology for Assessing Architectures. *Systems Journal, IEEE*,*5*(2), 244-255.

Dauby, J. P., & Upholzer, S. (2011). Exploring Behavioral Dynamics in Systems of Systems. *Procedia Computer Science*, *6*, 34-39.

Deb, K. (2000). An Efficient Constraint Handling Method for Genetic Algorithms. *Computer Methods in Applied Mechanics and Engineering*, *186*(2), 311-338.

Deb, K. (2001). *Multi-objective Optimization using Evolutionary Algorithms* (Vol. 16). John Wiley & Sons.

Deb, K., & Gupta, H. (2006). Introducing Robustness in Multi-objective Optimization. *Evolutionary Computation*, *14*(4), 463-494.

Deb, K., & Sinha, A. (2009, January). Solving Bi-level Multi-objective Optimization Problems using Evolutionary Algorithms. In *Evolutionary Multi-Criterion Optimization* (pp. 110-124). Springer Berlin Heidelberg.

DeLaurentis, D., Marais, K., Davendralingam, N., Han, S. Y., Uday, P., Fang, Z., & Gurainiello, C. (2012). *Assessing the Impact of Development Disruptions and Dependencies in Analysis of Alternatives of System of Systems*. Hoboken NJ: Stevens Institute of Technology, Systems Engineering Research Center,

Department of the Navy, (1997). *Contractor Performance Assessment Reporting System (CPARS)*. Washington DC.

Díaz, E., Tuya, J., & Blanco, R. (2003, October). Automated Software Testing Using a Metaheuristic Technique based on Tabu Search. In *Automated Software Engineering, 2003. Proceedings. 18th IEEE International Conference on* (pp. 310-313). IEEE.

Díaz, E., Tuya, J., Blanco, R., & Javier Dolado, J. (2008). A Tabu Search Algorithm for Structural Software Testing. *Computers & Operations Research*,*35*(10), 3052-3072.

Director Systems and Software Engineering, OUSD (AT&L). (2008). *Systems Engineering Guide for Systems of Systems*. Available from http://acq.osd.mil/se/doc/SE-Guid-for-SoS.pdf.

Djavanshir, G. R., Alavizadeh, A., & Tarokh, M. J. (2012). From System-of-Systems to Meta-Systems: Ambiguities and Challenges. *System of Systems*.

DoD, A. S. D. "DoD Architecture Framework Version 2.0 (DoDAF V2. 0)."*Department of Defense, Washington DC* (2009).

DoD, Navy. "Contractor Performance Assessment Reporting System (CPARS)," Washington DC (1997).

DoD, Systems engineering guide for systems of systems (2008). Tech. rep., Systems and Software Engineering, Department of Defense.

Dolado, J. J. (2000). A Validation of the Component-based Method for Software Size Estimation. *Software Engineering, IEEE Transactions on*, *26*(10), 1006-1021.

Dolado, J. J. (2001). On the Problem of the Software Cost Function. *Information and Software Technology*, *43*(1), 61-72.

Dombkins, D. (1996). *Project Managed Change: The Application of Project Management Techniques to Strategic Change Programs*. Centre for Corporate Change, Australian Graduate School of Management, University of New South Wales.

Dombkins, D. (2007). Complex Project Management. *South Carolina: Booksurge Publishing*.

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.

Domerçant, J. C., & Mavris, D. N. (2011, March). Measuring the architectural complexity of military systems-of-systems. In *Aerospace Conference, 2011 IEEE* (pp. 1-16). IEEE.

Dudek, G., Jenkin, M. R., Milios, E., & Wilkes, D. (1996). A Taxonomy for Multi-agent Robotics. *Autonomous Robots*, *3*(4), 375-397.

Dudenhoeffer, D. D., & Jones, M. P. (2000). A Formation Behavior for Large-scale Micro-robot Force Deployment. In *Simulation Conference, 2000. Proceedings. Winter* (Vol. 1, pp. 972-982). IEEE.

Dutta, P. K. (1999). *Strategies and Games: Theory and Practice*. MIT Press.

Eichfelder, G. (2010). Multi-objective Bi-level Optimization. *Mathematical Programming*, *123*(2), 419-449.

Ender, T., Leurck, R. F., Weaver, B., Miceli, P., Blair, W. D., West, P., & Mavris, D. (2010). Systems-of-systems analysis of ballistic missile defense architecture effectiveness through surrogate modeling and simulation. *Systems Journal, IEEE*, 4(2), 156-166.

Epperly, T. G. W. (1995). *Global Optimization of Non-convex Non-linear Programs using Parallel Branch and Bound* (Doctoral dissertation, UNIVERSITY OF WISCONSIN–MADISON).

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

Faratin, P., Sierra, C., & Jennings, N. R. (1998). Negotiation Decision Functions for Autonomous Agents. *Robotics and Autonomous Systems*, *24*(3), 159-182.

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. (2004). An Agenda-based Framework for Multi-issue Negotiation. *Artificial Intelligence*, *152*(1), 1-45.

Flanigan, D., & Brouse, P. (2012). System of Systems Requirements Capacity Allocation. *Procedia Computer Science*, *8*, 112-117.

Fogel, D. B. (2006). *Evolutionary Computation: Toward a new Philosophy of Machine Intelligence* (Vol. 1). John Wiley & Sons.

Fonseca, C. M., & Fleming, P. J. (1995). An Overview of Evolutionary Algorithms in Multi-objective Optimization. *Evolutionary Computation*, *3*(1), 1-16.

Fonseca, C. M., & Fleming, P. J. (1998). Multi-objective Optimization and Multiple Constraint Handling with Evolutionary Algorithms. I: A Unified Formulation. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, *28*(1), 26-37.

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

Gao, J., Buldyrev, S. V., Stanley, H. E., & Havlin, S. (2011, January). Networks Formed from Interdependent Networks. *Nature Physics, 8,* 40-48, doi:10.1038/NPHYS2180.

Gao, Y., Zhang, G., & Lu, J. (2009). A Fuzzy Multi-objective Bi-level Decision Support System. *International Journal of Information Technology & Decision Making*, *8*(01), 93-108.

Garrett, R. K., Anderson, S., Baron, N. T., & Moreland, J. D. (2011). Managing the Interstitials, a System of Systems Framework Suited for the Ballistic Missile Defense System. *Systems Engineering*, *14*(1), 87-109.

Garvey, P., & Pinto, A. (2009, June). Introduction to Functional Dependency Network Analysis. In *The MITRE Corporation and Old Dominion, Second International Symposium on Engineering Systems, Massachusetts Institute of Technology, Cambridge, Massachusetts*.

Ge, B., Hipel, K. W., Yang, K., & Chen, Y. (2013). A Data-centric Capability-focused Approach for System-of-systems Architecture Modeling and Analysis. *Systems Engineering*, *16*(3), 363-377.

Gegov, A. (2010). *Fuzzy Networks for Complex Systems A Modular Rule Base Approach*. Springer-Verlag Berlin Heidelberg.

Giachetti, R. E. (2012). A Flexible Approach to Realize an Enterprise Architecture. *Procedia Computer Science*, *8*, 147-152.

Glover, F. (1989). Tabu Search – Part I. *ORSA Journal on Computing*, *1*(3), 190-206.

Glover, F. (1990). Tabu Search – Part II. *ORSA Journal on Computing*, *2*(1), 4-32.

Goldberg, D. E. (1990). Genetic Algorithms in Search, Optimization and Machine Learning. *Reading: Addison-Wesley*.

Gonzalez-Zugasti, J. P., Otto, K. N., & Baker, J. D. (2000). A Method for Architecting Product Platforms. *Research in Engineering Design*, *12*(2), 61-72.

Gorod, A., Gandhi, S. J., Sauser, B., & Boardman, J. (2008). Flexibility of system of systems. *Global Journal of Flexible Systems Management*, 9(4), 21-31.

Gries, M. (2004). Methods for Evaluating and Covering the Design Space during Early Design Development. *Integration, the VLSI journal*, *38*(2), 131-183.

Guariniello, C., & DeLaurentis, D. (2013). Dependency Analysis of System-of-Systems Operational and Development Networks. *Procedia Computer Science*,*16*, 265-274.

Haimes, Y. Y. (2012). Modeling Complex Systems of Systems with Phantom System Models. *Systems Engineering*, *15*(3), 333-346.

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.

Hansen, P., Jaumard, B., & Savard, G. (1992). New Branch-and-bound Rules for Linear Bi-level Programming. *SIAM Journal on Scientific and Statistical Computing*, *13*(5), 1194-1217.

Hassan, R., & Crossley, W. (2007). Approach to Discrete Optimization Under Uncertainty: The Population-Based Sampling Genetic Algorithm. *AIAA Journal*,*45*, 2799-2809.

Hassan, R., De Weck, O., & Springmann, P. (2004, May). Architecting a Communication Satellite Product Line. In *22nd AIAA International Communications Satellite Systems Conference & Exhibit (ICSSC). Monterey, CA*.

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.

Henson, S. A., Henshaw, M. J. D., Barot, V., Siemieniuch, C. E., Sinclair, M. A., Jamshidi, M., ... & DeLaurentis, D. (2013, June). Towards a Systems of Systems Engineering EU Strategic Research Agenda. In *System of Systems Engineering (SoSE), 2013 8th International Conference on* (pp. 99-104). IEEE.

Herroelen, W., De Reyck, B., & Demeulemeester, E. (1998). Resource-constrained Project Scheduling: a Survey of Recent Developments. *Computers & Operations Research*, *25*(4), 279-302.

Hill Climbing. (n.d.). Retrieved October 24, 2013, from http://en.wikipedia.org/wiki/Hill_climbing

Holland, J. H. (1973). Genetic Algorithms and the Optimal Allocation of Trials. *SIAM Journal on Computing*, *2*(2), 88-105.

Hunt, B. R., Lipsman, R. L., Rosenberg, J. M., Coombes, K. R., Osborn, J. E., & Stuck, G. J. (2006). *A Guide to MATLAB: for Beginners and Experienced Users*. Cambridge University Press.

Hwang, C. L., Masud, A. S. M., Paidy, S. R., & Yoon, K. P. (1979). *Multiple objective Decision Making, Methods and Applications: a State-of-the-art Survey* (Vol. 164). Berlin: Springer.

INCOSE, (2011). *SYSTEMS ENGINEERING HANDBOOK v 3.2.2.* San Diego: INCOSE.

Jamshidi, M. (2008). System of Systems Engineering - New Challenges for the 21st century. *Aerospace and Electronic Systems Magazine, IEEE*, *23*(5), 4-19.

Jamshidi, M. (Ed.). (2011). *System of systems engineering: innovations for the twenty-first century (Vol. 58)*. John Wiley & Sons.

Jackson, S., & Ferris, T. L. (2013). Resilience Principles for Engineered Systems. *Systems Engineering*, *16*(2), 152-164.

Jia, L., Wang, Y., & Fan, L. (2011, December). Uniform Design Based Hybrid Genetic Algorithm for Multi-objective Bi-level Convex Programming. In *Computational Intelligence and Security (CIS), 2011 Seventh International Conference on* (pp. 159-163). IEEE.

Johnston, W., Mastran, K., Quijano, N., & Stevens, M. (2013). Unmanned Vehicle Search and Rescue Initiative. *SE4150 Design Project, Naval Postgraduate School*.

Joint Staff (2010). CJCSM 3500.04C, *UNIVERSAL JOINT TASK LIST (UJTL)*. Washing DC: Department of Defense.

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.

Kaplan, J. (2006). *A New Conceptual Framework for Net-Centric, Enterprise-Wide, System-of-Systems Engineering (v)*. Washington, DC: Center for Technology and National Security Policy. National Defense University.

Karnik, N. N., Mendel, J. M., & Liang, Q. (1999). Type-2 Fuzzy Logic Systems. *Fuzzy Systems*, IEEE Transactions on, 7(6), 643-658.

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

Kilicay-Ergin, N., Enke, D., & Dagli, C. (2012). Biased Trader Model and Analysis of Financial Market Dynamics. *International Journal of Knowledge-based and Intelligent Engineering Systems*, *16*(2), 99-116.

Kinnunen, M. J. (2006). *Complexity Measures for System Architecture Models* (Doctoral dissertation, Massachusetts Institute of Technology).

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. S*cience, 220*(4598), 671-680.

Klein, J., & van Vliet, H. (2013, June). A systematic review of system-of-systems architecture research. In *Proceedings of the 9th international ACM Sigsoft conference on Quality of software architectures* (pp. 13-22). ACM.

Koch, P. N., Evans, J. P., & Powell, D. (2002). Inter-digitation for Effective Design Space Exploration using iSIGHT. *Structural and Multidisciplinary Optimization*, *23*(2), 111-126.

Konur, D., & Dagli, C. (2014). Military System of Systems Architecting with Individual System Contracts*. Optimization Letters*, 1-19.

Konur, D., & Golias, M. M. (2013). Cost-stable Truck Scheduling at a Cross-dock Facility with Unknown Truck Arrivals: A Meta-heuristic Approach. *Transportation Research Part E: Logistics and Transportation Review*, *49*(1), 71-91.

Kraus, S. (1996). An Overview of Incentive Contracting. *Artificial Intelligence*, *83*(2), 297-346.

Kraus, S. (2001). Automated Negotiation and Decision Making in Multi-agent 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.

Lafleur, J. M. (2012). *A Markovian State-Space Framework for Integrating Flexibility into Space System Design*. Atlanta: Georgia Institute of Technology School of Aerospace Engineering Doctoral Thesis.

Lamar, B. W., & Bedford, M. A. (2009). Min-additive Utility Functions. *MITRE Corporation*.

Lane, J. A., & Bohn, T. (2013). Using SysML Modeling to Understand and Evolve Systems of Systems. *Systems Engineering*, *16*(1), 87-98.

Li, C., & Chiang, T. W. (2013). Complex Neurofuzzy ARIMA Forecasting — A New Approach Using Complex Fuzzy Sets. *Fuzzy Systems, IEEE Transactions on*, *21*(3), 567-584.

Li, M., Lin, D., & Wang, S. (2010). Solving a Type of Bi-objective Bi-level Programming Problem Using NSGA-II. *Computers & Mathematics with Applications*, *59*(2), 706-715.

Lopes, F., Wooldridge, M., & Novais, A. Q. (2008). Negotiation among Autonomous Computational Agents: Principles, Analysis and Challenges. *Artificial Intelligence Review*, *29*(1), 1-44.

Lu, J., Zhang, G., & Dillon, T. (2008). Fuzzy Multi-objective Bi-level Decision Making by an Approximation Kth-Best Approach. *Journal of Multiple-Valued Logic & Soft Computing*, *14*.

Luzeaux, D., System-of-Systems (and Large-Scale Complex Systems) Engineering, presentation at CSDM Conference, 2013.

Maier, M. W. (1998). Architecting principles for systems-of-systems. *Systems Engineering*, 1(4), 267-284.

Maier, M. W., & Rechtin, E. (2009). *The Art of Systems Architecting, 3rd ed.* Boca Raton: CRC Press.

Malan, R., & Bredemeyer, D. (2001). Architecture Resources. *Defining Non-Functional Requirements*. Retreived from http://www.bredemeyer.com/pdf_files/ArchitectureDecisions.pdf

Manthorpe, W. H. (1996). *The emerging joint system of systems: A systems engineering challenge and opportunity for APL.* Johns Hopkins APL Technical Digest, 17(3), 305.

Maskin, E. S. (1996). Theories of the Soft Budget-constraint. *Japan and the World Economy*, *8*(2), 125-133.

Mekdeci, B., Shah, N., Ross, A. M., Rhodes, D. H., & Hastings, D. (2014). *Revisiting the Question: Are Systems of Systems just (traditional) Systems or Are they a new class of Systems?* Cambridge, MA: Systems Engineering Advancement Research Initiative (SEAri).

Mendel, J. M. (2013). On KM algorithms for Solving Type-2 Fuzzy Set Problems. *Fuzzy Systems, IEEE Transactions on*, *21*(3), 426-446.

Mendel, J. M., & John, R. B. (2002). Type-2 Fuzzy Sets Made Simple. *Fuzzy Systems, IEEE Transactions on*, *10*(2), 117-127.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, *21*(6), 1087-1092.

Mezura Montes, E., & Coello Coello, C. A. (2005). A Simple Multi-membered Evolution Strategy to Solve Constrained Optimization Problems. *Evolutionary Computation, IEEE Transactions on*, *9*(1), 1-17.

Michalewicz, Z., & Schoenauer, M. (1996). Evolutionary Algorithms for Constrained Parameter Optimization Problems. *Evolutionary Computation*, *4*(1), 1-32.

Miettinen, K. M., & Optimization, N. L. M. O. (1999). Kluwer Academic Publisher.

Migdalas, A., Pardalos, P. M., & Värbrand, P. (Eds.). (1998). *Multilevel Optimization: Algorithms and Applications* (Vol. 20). Springer.

Min, B. K., & Chang, S. H. (1991). System Complexity Measure in the Aspect of Operational Difficulty. *Nuclear Science, IEEE Transactions on*, *38*(5), 1035-1039.

Mordecai, Y., & Dori, D. (2013). I5: A Model-Based Framework for Architecting System-of-Systems Interoperability, Interconnectivity, Interfacing, Integration, and Interaction. In *International Symposium of the International Council on Systems Engineering (INCOSE)*.

Mostafavi, A., Abraham, D., Noureldin, S., Pankow, G., Novak, J., Walker, R., ... & George, B. (2012). Risk-Based Protocol for Inspection of Transportation Construction Projects Undertaken by State Departments of Transportation. *Journal of Construction Engineering and Management*, *139*(8), 977-986.

Osman, M. S., Abo-Sinna, M. A., Amer, A. H., & Emam, O. E. (2004). A Multi-level Non-linear Multi-objective Decision-making under Fuzziness. *Applied Mathematics and Computation*, *153*(1), 239-252.

Owens, W. A. (1996). *The emerging US system-of-systems (No. 63).* National Defense University, Washington D.C., Instruction for National Strategic Studies.

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.

Pedrycz, W., Ekel, P., & Parreiras, R. (2011). *Fuzzy Multi-criteria Decision-making: Models, Methods and Applications*. John Wiley & Sons.

Pernin, C. G., Axelband, E., Drezner, J. A., Dille, B. B., Gordon, I. V., Held, B. J., ... & Shah, A. R. (2012). *Lessons from the Army's Future Combat Systems Program*. Rand Arroyo Center Santa Monica CA.

Pieume, C. O., Marcotte, P., Fotso, L. P., & Siarry, P. (2011). Solving Bi-level Linear Multi-objective Programming Problems. *American Journal of Operations Research*, *1*, 214.

Pitsko, R., & Verma, D. (2012). Principles for Architecting Adaptable Command and Control Systems. *Procedia Computer Science*, *8*, 135-140.

Ravindran, A., Reklaitis, G. V., & Ragsdell, K. M. (2006). *Engineering Optimization: Methods and Applications*. John Wiley & Sons.

Räihä, O. (2008). Applying Genetic Algorithms in Software Architecture Design.

Rela, L. (2004). Evolutionary Computing in Search-based Software Engineering.

Rios, L. M., & Sahinidis, N. V. (2013). Derivative-free Optimization: a Review of Algorithms and Comparison of Software Implementations. *Journal of Global Optimization*, *56*(3), 1247-1293.

Ricci, N., Ross, A. M., Rhodes, D. H., & Fitzgerald, M. E. (2013). Considering Alternative Strategies for Value Sustainment in Systems-of-Systems (Draft). *Systems Engineering Advancement Research Initiative, Cambridge MA*.

Rosenau, W. (1991). Coalition Scud Hunting in Iraq, 1991. *RAND Corporation*.

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*, 246 – 262.

Ross, S. M. (2014). *Introduction to Probability Models, 8th Edition*. Academic Press.

Rostker, B. (200, July 25). *Iraq's Scud Ballistic Missiles*. Retrieved Sep 12, 2013, from Iraq Watch: http://www.iraqwatch.org/government/US/Pentagon/dodscud.htm

Runarsson, T. P., & Yao, X. (2000). Stochastic Ranking for Constrained Evolutionary Optimization. *Evolutionary Computation, IEEE Transactions on*, *4*(3), 284-294.

Russell, S. (2009). Artificial Intelligence: A Modern Approach Author: Stuart Russell, Peter Norvig, Publisher: Prentice Hall Pa.

Saleh, J. H., Hastings, D. E., & Newman, D. J. (2001). Extracting the essence of flexibility in system design. In *Evolvable Hardware, 2001. Proceedings. The Third NASA/DoD Workshop on* (pp. 59-72). IEEE.

Saleh, J. H., Mark, G., & Jordan, N. C. (2009). Flexibility: a multi-disciplinary literature review and a research agenda for designing flexible engineering systems. *Journal of Engineering Design*, 20(3), 307-323.

Sanz, J. A., Fernández, A., Bustince, H., & Herrera, F. (2013). IVTURS: A Linguistic Fuzzy Rule-Based Classification System Based On a New Interval-Valued Fuzzy Reasoning Method With Tuning and Rule Selection. *IEEE T. Fuzzy Systems*, *21*(3), 399-411.

Schäfer, R. (2001, October). Rules for using Multi-attribute Utility Theory for Estimating a User's Interests. In *Ninth Workshop Adaptivität und Benutzenmodellierung in Interaktiven Softwaresystemen* (pp. 8-10).

Schreiner, M. W., & Wirthlin, J. R. (2012). Challenges Using Modeling and Simulation in Architecture Development. *Procedia Computer Science*, *8*, 153-158.

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.

Selva, D., & Crawley, E. F. (2013). VASSAR: Value Assessment of System Architectures using Rules. *IEEE Aerospace Conference* (pp. 1-21). Big Sky MT: IEEE.

Shi, X., & Xia, H. S. (2001). Model and Interactive Algorithm of Bi-level Multi-objective Decision-making with Multiple Interconnected Decision Makers. *Journal of Multi-Criteria Decision Analysis*, *10*(1), 27-34.

Shtub, A., Bard, J. F., & Globerson, S. (1994). *Project Management: Engineering, Technology, and Implementation*. Prentice-Hall, Inc..

Singer, Y. (2006, November). Dynamic Measure of Network Robustness. In *Electrical and Electronics Engineers in Israel, 2006 IEEE 24th convention of* (pp. 366-370). IEEE.

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).

Simpson, T. W., & D'souza, B. S. (2002). Assessing Variable Levels of Platform Commonality within a Product Family using a Multi-objective Genetic Algorithm. In *Proceeding of the 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, AIAA*.

Singh, A., & Dagli, C. H. (2009, March). Multi-objective Stochastic Heuristic Methodology for Tradespace Exploration of a Network Centric System of Systems. In *Systems Conference, 2009 3rd Annual IEEE* (pp. 218-223). IEEE.

Smartt, C., & Ferreira, S. (2012). Constructing a General Framework for Systems Engineering Strategy. *Systems Engineering*, *15*(2), 140-152.

Sommerer, S., Guevara, M. D., Landis, M. A., Rizzuto, J. M., Sheppard, J. M., & Grant, C. J. (2012). *Systems-of-Systems Engineering in Air and Missile Defense*. Johns Hopkins APL Technical Digest, 31(1), 5-20.

SPEC Innovations. (2014). *Model-Based Systems Engineering Tools*. Retrieved August 20,2014, from https://www.innoslate.com/systems-engineering.

Suarez, R. (2004, Dec 9). *Troops Question Secretary of Defense Donald Rumsfeld about Armor*. Retrieved Apr 14, 2014, from PBS NewsHour: http://www.pbs.org/newshour/bb/military-july-dec04-armor_12-9/

Sumathi, S., & Paneerselvam, S. (2010). *Computational Intelligence Paradigms: Theory & Applications using MATLAB*. CRC Press.

Trans-Atlantic Research and Education Agenda in Systems of Systems (T-AREA-SOS) Project, "The Systems of Systems Engineering Strategic Research Agenda," Loughborough University, Loughborough, 2013.

Talbot, F. B. (1982). Resource-constrained Project Scheduling with Time-resource Tradeoffs: The Non-preemptive Case. *Management Science*, *28*(10), 1197-1210.

Taleb, N. N. (2004). *Fooled by Randomness*. New York: Random House Trace Paperbacks.

Thompson, M. (2002). Iraq: The Great Scud Hunt. *Time Magazine*, *23*.

Valerdi, R., Axelband, E., Baehren, T., Boehm, B., Dorenbos, D., Jackson, S., ... & Settles, S. (2008). A research agenda for systems of systems architecting. *International Journal of System of Systems Engineering*, 1(1), 171-188.

Wall, M. B. (1996). *A Genetic Algorithm for Resource-constrained Scheduling* (Doctoral dissertation, Massachusetts Institute of Technology).

Wang, J. Q., & Zhang, H. Y. (2013). Multi-criteria Decision-making Approach based on Atanassov's Intuitionistic Fuzzy Sets with Incomplete Certain Information on Weights. Fuzzy Systems, IEEE Transactions on, 21(3), 510-515.

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., & Dagli, C. H. (2013). Developing a Holistic Modeling Approach for Search-based System Architecting. *Procedia Computer Science*, *16*, 206-215.

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, Y., & Cai, Z. (2012). Combining Multi-objective Optimization with Differential Evolution to Solve Constrained Optimization Problems. *Evolutionary Computation, IEEE Transactions on*, *16*(1), 117-134.

Wang, Y., Cai, Z., Guo, G., & Zhou, Y. (2007). Multi-objective Optimization and Hybrid Evolutionary Algorithm to Solve Constrained Optimization Problems. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, *37*(3), 560-575.

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.

Wappler, S. (2007). *Automatic Generation of Object-oriented Unit Tests using Genetic Programming* (Doctoral dissertation, Universitätsbibliothek).

Wappler, S., & Wegener, J. (2006, July). Evolutionary Unit Testing of Object-oriented Software using Strongly-typed Genetic Programming. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation* (pp. 1925-1932). ACM.

Warfield, J. N. (1973). Binary Matrices in Systems Modeling. *IEEE Transactions on Systems, Man, and Cybernetics, SMC-3* (No. 5, September), 441-449.

Weiss, W. E. (2008, December). Dynamic Security: An Agent-based Model for Airport Defense. In *Simulation Conference, 2008. WSC 2008. Winter* (pp. 1320-1325). IEEE.

Welby, S. P. Systems Engineering FY 2012 Annual Report.

Woldesenbet, Y. G., Yen, G. G., & Tessema, B. G. (2009). Constraint Handling in Multi-objective Evolutionary Optimization. *Evolutionary Computation, IEEE Transactions on*, *13*(3), 514-525.

Wooldridge, M., & Parsons, S. (2000, August). Languages for Negotiation. In *ECAI* (pp. 393-400).

Wu, D., & Mendel, J. M. (2007). Uncertainty Measures for Interval Type-2 Fuzzy Sets. *Information Sciences*, *177*(23), 5378-5393.

Yi, J. S., Kang, Y., Stasko, J. T., & Jacko, J. A. (2007), Nov/Dec). Toward a Deeper Understanding of the Role of Interaction in Information Visualization. *IEEE Transactions on Visualization and Computer Graphics, 13(6),* 1224-1231.

Yu, O. Y., Guikema, S. D., Briaud, J. L., & Burnett, D. (2012). Sensitivity Analysis for Multi-attribute System Selection Problems in Onshore Environmentally Friendly Drilling (EFD). *Systems Engineering*, *15*(2), 153-171.

Yu, T. L., Yassine, A. A., & Goldberg, D. E. (2007). An Information Theoretic Method for Developing Modular Architectures using Genetic Algorithms. *Research in Engineering Design*, *18*(2), 91-109.

Zadeh, L. A. (1975). Fuzzy Logic and Approximate Reasoning. *Synthese*, *30*(3-4), 407-428.

Zhang, T., Hu, T., Zheng, Y., & Guo, X. (2012). An Improved Particle Swarm Optimization for Solving Bi-level Multi-objective Programming Problem. *Journal of Applied Mathematics*, *2012*.

Zhang, T., Hu, T., Chen, J. W., Wan, Z., & Guo, X. (2012, November). Solving Bi-level Multi-objective Programming Problem by Elite Quantum Behaved Particle Swarm Optimization. In *Abstract and Applied Analysis* (Vol. 2012). Hindawi Publishing Corporation.

Zhang, G., Lu, J., & Gao, Y. (2008). An Algorithm for Fuzzy Multi-objective Multi-follower Partial Cooperative Bi-level Programming. *Journal of Intelligent and Fuzzy Systems*, *19*(4), 303-319.

Zhang, G., Lu, J., & Gao, Y. (2008). Fuzzy Bi-level Programming: Multi-objective and Multi-follower with Shared Variables. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *16*(supp02), 105-133.

Zhang, G., & Lu, J. (2010). Fuzzy Bi-level Programming with Multiple Objectives and Cooperative Multiple Followers. *Journal of Global Optimization*, *47*(3), 403-419.

Zheng, Y., Wan, Z., & Wang, G. (2011). A Fuzzy Interactive Method for a Class of Bi-level Multi-objective Programming Problem. *Expert Systems with Applications*, *38*(8), 10384-10388.

Zhou, A., Qu, B. Y., Li, H., Zhao, S. Z., Suganthan, P. N., & Zhang, Q. (2011). Multi-objective Evolutionary Algorithms: A Survey of the State of the Art. *Swarm and Evolutionary Computation*, *1*(1), 32-49.