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

Volume 9 – Building Executable Architecture on Notational SoS

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
# Table of Contents

Executive Summary .......................................................................................................................... 1  
Introduction .................................................................................................................................. 7    
  Motivation for Research ................................................................................................................ 7  
  System of System Challenges ...................................................................................................... 9  
  How Does FILA-SoS Address SoS Pain Points ......................................................................... 12  
Overview of the FILA-SoS integrated model .............................................................................. 16  
  Definition of Variables for SoS ................................................................................................. 16  
  Independent modules of FILA-SoS ............................................................................................ 19  
Building Executable Architecture on Notational SoS ................................................................. 23  
  Problem Definition ...................................................................................................................... 24  
  System Model ............................................................................................................................... 26  
  Performance Assessment ............................................................................................................. 31  
  Concluding Remarks .................................................................................................................. 33  
Appendix A: List of Publicas Resulted and Papers Submitted from FILA-SoS Research .......... 34  
Appendix B: Cited and Related References ................................................................................. 36

# List of Figures

Figure 1 Schematic Drawing of Four Classical Types of SoS Based on Degree of Control and Degree of Complexity .......................................................................................................................... 8  
Figure 2 ISR System-of-Systems for Testing FILA-SoS ................................................................ 14  
Figure 3 SAR System-of-Systems for Testing FILA-SoS ................................................................. 14  
Figure 4 Aircraft Carrier Performance Assessment for Testing FILA-SoS ..................................... 15  
Figure 5 The Wave Model of SoS initiation, Engineering, and Evolution ..................................... 17  
Figure 6 Integrated modules within FILA-SoS ............................................................................ 20  
Figure 7 Generative class model of the SoS represented using OPM ........................................... 27  
Figure 8 Unfolding the Reconnaissance System ........................................................................... 27  
Figure 9 CPN model converted from OPM .................................................................................. 29  
Figure 10 CPN substitution transition representing data collection activity .................................. 30  
Figure 11 A sample CPN function declaration ............................................................................. 30  
Figure 12 Coverage area as a function of simulation time ............................................................ 32
LIST OF TABLES

Table 1 System of Systems and Enterprise Architecture Activity ................................................. 10
Table 2 Parameters of participating systems (capability providers) .................................................. 25
Table 3 An example of token values and the corresponding information encoded for systems 30
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 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:

**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.

---

*Figure 1 Schematic Drawing of Four Classical Types of SoS Based on Degree of Control and Degree of Complexity*
This research is based on Acknowledged SoS. The major objectives of the research 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 of 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

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

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-SO 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.
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.

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.
These examples 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:

\[
\begin{align*}
C & : \text{Overall capability (the overall goal to be achieved by combining sub-capabilities)} \\
c_j & : j \in J = \{1, 2, \ldots, M\} \\
& \quad \text{Constituent system capabilities required} \\
s_i & : i \in I = \{1, 2, \ldots, N\} \\
& \quad \text{Total number of systems present in the SoS problem} \\
A & : \text{A} \text{ matrix of } a_{ij} \text{ where} \\
a_{ij} & = 1 \text{ if capability } j \text{ is possessed by system } i \\
a_{ij} & = 0 \text{ otherwise} \\
P_i & : \text{Performance of system } i \text{ for delivering all capabilities } \sum_j a_{ij} \\
F_i & : \text{Funding of system } i \text{ for delivering all capabilities } \sum_j a_{ij} \\
D_i & : \text{Deadline to participate in this round of mission development for system } i \\
IF_{ik} & : \text{Interface between systems } i \text{ and } k \text{ s.t. } s \neq k, k \in I \\
IC_i & : \text{The cost for development of interface for system } i \\
OC_i & : \text{The cost of operations for system } i \\
KP_r & : r \in R, R = \{1, 2, \ldots, Z\} \\
& \quad \text{The key performance attributes of the SoS} \\
FA & : \text{Funding allocated to SoS Manager} \\
p & = \{1, 2, \ldots, P\} \\
& \quad \text{Number of negotiation attributes for bilateral negotiation} \\
t_{\max} & : \text{Total round of negotiations possible}
\end{align*}
\]
\( t \): Current round of negotiation (epochs)
\( t_{\text{max}} \): Total round of negotiations possible
\( V^\text{SoS}_{pi}(t) \): The value of the attribute \( p \) for SoS manager at time \( t \) for system \( i \)
\( V^S_{pi}(t) \): 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.

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 \( t_{\text{max}} \) 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 types 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.
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 is 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 a new 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.
While a substantial amount of information about a system can be derived from analyzing its stationary properties, more detailed assessment of system performance entails the consideration of the interactions between system components or between participating systems of a SoS. For examine, by analyzing such interactions, system architect can have a much better understanding regarding whether participating systems can collaborate with each other in delivering the desired capabilities and how well such collaboration might be when the SoS is in operation. This in turn requires examining the workflow that is needed to achieve the overall SoS functionalities, the dependency between functions and capabilities, the information or resources flow between related activities, and parameters that define these interactions.

To achieve such capabilities, an executable architecture model is imperative. In this research, a modeling approach that combines the capabilities of Object Process Methodology (OPM) and Colored Petri Net (CPN) is proposed. OPM developed by Dori (Dori, 2002) is a general-purpose modeling language with a single model formalism. The building blocks of OPM are entities (things and states) and links that connect them. Objects are things that exist and they may have states. At any particular point in time, an object can be exactly in one state, and object states are changed through processes (Reinhartz-Berger & Dori, 2004). Processes are things that transform objects. Links can be structural or procedural. Structural links express static relations between pairs of objects or process (Dori & Reinhartz-Berger, 2003) while procedural links describe the behavior of a system (Dori & Reinhartz-Berger, 2003), which can be summarized as the following three ways: (1) processes transform (generating, consuming or affecting) objects; (2) objects can enable process without being transformed by them; and (3) things can trigger events that invoke processes (Reinhartz-Berger & Dori, 2005).

A Petri net (Wikarski, 1997; Peterson, 1981) is a mathematical modeling language manifests as a directed bi-partite graph that is consists of places and transitions and directed arcs that connect a place to a transition or vice versa. Place can store tokens which represent objects in the system. The distribution of tokens over the places collectively marks the state of the system. With the use of tokens to mark the state of a system, Petri nets can captures the dynamic aspects of a system. Transitions represent the actions of a system. When certain conditions hold, a transition will fire, causing a change in the placement of tokens and thus the change of system states. Therefore, Petri Nets can model the causalities between interacting entities and thus often used for discrete event system modeling and simulation. There are several extensions to basic Petri Nets, such as CPNs, which allow tokens to be typed, timed Petri nets, which introduce time concepts into transition, stochastic Petri nets, which add nondeterministic time through adjustable randomness of the transitions, and Object-oriented Petri nets, which support object-oriented modeling, to name a few.

In the approach proposed for this research, 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. OPM modeling supports both object-oriented and process-oriented paradigm. Object-oriented
modeling, as one of the most popular modeling paradigms, can capture a variety of systems, at various levels of abstraction, from various types of perspective. Process-oriented modeling allows defining processes independently of objects. CPN supplements OPM with state-transition-based execution semantics supporting discrete-event system simulation, which is an indispensable means to derive certain performance metrics and to conduct precise behavior analyses. The incorporation of CPN also allows the developed system model to be doubled as an analysis model. A large collection of analysis methods and tools developed for CPN can be utilized for strong model analysis, verification, and validation. Such integration of a system model with an analysis model not only avoids the loss of fidelity during model transformation but also eliminate the need to develop a new analysis model when the system model changes. The mapping between OPM and CPN follows the algorithms developed in (Wang, 2012; Wang & Dagli 2013). Such hybrid modeling approach is explained here using a sample project, the architecture development of an ISR system.

- Toy Problem for aircraft carrier performance assessment
- Intelligence, Surveillance and Reconnaissance Problem

**Problem Definition**

Suppose an ISR system to be developed is required to work under either of the following two scenarios in carrying out its designated mission: Scenario 1: Find concentration and movement of irregular troops that might be a warning of impending attack; and Scenario 2: Provide surveillance support to friend troops during an attack. The performance metrics to be considered include (Refer to Table 2) (1) total covered area under the surveillance of the system and (2) latency, i.e., time between the observations being collected and the decision being made in response to the observations obtained. Other metrics like the cost of the system and the time needed to build the system is omitted for simplicity.

Based on such requirements, the capabilities that the system needs to possess and can be provided by existing systems include (1) terrain image capture capability provided by ElectroOptic and Infrared (EO/IR) with repeat coverage of large area and high resolution and radar signal provided SAR, which has better capabilities in identifying moving vehicles or concentration of equipment under all weather conditions; (2) Communication system for data transmission; (3) Exploitation center for data analysis; (4) Command and control station (C&C) for decision making. The systems providing the above capabilities need to collaborate with each other in carrying out the mission required by the SOS. Such interactions can be described by the following work flow:

- The C&C selects and dispatches required number of systems that carry EO, IR or SAR capabilities to the target terrain.
- The deployed systems carrying EO, IR or SAR capabilities watch the designated terrain and collect the observations with the carried equipment.
- The collected data are then transmitted over wide-band data links (provided by communication systems) to the exploitation centers in real time.
• Upon receiving the data, the exploitation centers analyze the data and provide recommendations.
• The recommendations are sent to the C&C through the communication systems.
• C&C makes decisions based on the data received from the exploitation center. Such decisions may include adjusting the deployment of the EO, IR and SAR capabilities by sending command through communication system to those deployed systems.

Based on the above design, both of these two scenarios employ the same set of capabilities and follow the same workflow. However, the systems involved will have different operating parameters based on the task performed. The result is a different system deployment. For example, scenario 1 will require the system to have a broad range of coverage so as to monitor a large area. The latency should be as small as the mission requires. The second scenario will require the system to watch a much smaller area providing higher resolution information. Most importantly, it requires very low latency (near real time response). On the other hand, less communication infrastructure is needed and the exploitation center only need to filter the information, no extensive analysis is needed or can be afforded due to time limit. Based on the capabilities required, the systems selected as capability providers and related parameters are summarized in Table 2.

Table 2: Parameters of participating systems (capability providers)

<table>
<thead>
<tr>
<th>Systems</th>
<th>Shadow</th>
<th>Gray</th>
<th>apache</th>
<th>C&amp;C</th>
<th>Exp.</th>
<th>Atl</th>
<th>Voice/Chat</th>
<th>LOS</th>
<th>BLOS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>G</td>
<td>H</td>
<td>I</td>
</tr>
<tr>
<td>ID</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operating time (hour)</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>25</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>Time between Operations</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>Coverage</td>
<td>85</td>
<td>85</td>
<td>85</td>
<td>250</td>
<td>200</td>
<td>200</td>
<td>200</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>200</td>
<td>200</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capacity</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Capability</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>h</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>EO</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>IR</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Radar</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Command, Control</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Data Analysis</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Fusion</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Strike</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LOS</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>BLOS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

* "1" means possession of capability and "0" means absent of capability
Based on the analysis presented in last section, an OPM model can be developed to model the system as shown in Figure 7 and Figure 8. Figure 7 presents the overall system model whereas Figure 8 presents an unfolding of the ReconnaissanceSys object to reveal its detailed characters. Similar unfolding for other objects in Figure 7 is omitted here for simplicity. The systems are grouped into four types of systems: Reconnaissance, Communication, Exploitation center, and Command & Control systems. They are modeled as OPM objects (represented as rectangular boxes) in Figure 8. Reconnaissance systems have two states, Operation and OffDuty. They are represented by smoothed rectangles encapsulated in the ReconnaissanceSys object. Two process (oval shaped) GoOffDuty and GoOnDuty can change a reconnaissance system from Operation state to OffDuty state or vice versa. Observations are collected by reconnaissance systems, and then transmit to Exploitation center for analysis, and finally to Command & Control systems for decision-making. The OPM object, Data, represents all types of data to be processed by the SoS. It has five states (Raw, ReceivedByExpl, Recommendations, ReceivedByCC, and ActionPlan), each of which represents a stage of the information being processed during the operation of the SoS. Four processes (Transmit1, Analyze, Transmit2, and DecisionMaking) cause the change of states of data object. The execution of each of these processes requires the support of one or more objects as indicated by instrument links of OPM (circle end connectors between objects and processes). Such an OPM is a class model, from which instance models can be created representing a particular system configuration to be used.

Due to the dynamic of the operations of the SoS, the performance to be measured (coverage area and latency) won’t stay constant. For example, a particular reconnaissance system needs to have a break after operating for the maximum allowed period. With systems going on and off duty, the total coverage area by these systems might fluctuate. Furthermore, the observations may not arrive at fixed intervals and, therefore, the capacity of the data processing systems may be overloaded or under loaded during the operation resulting variant length of latency. Such variations resulting from the interactions of the constituent systems are rather difficult to estimate statically. Simulations, such as the capability offered by CPN can be very helpful in such cases.
An OPM model such as the one developed above can be mapped to CPN using the approach developed in (Wang, 2012; Wang & Dagli 2013). The essence of such mapping is summarized as follows: Map OPM processes to CPN transitions. Map OPM attribute objects (objects connected to their parent object using exhibition-characterization links) to CPN color sets. Such color set thus defines the set of class attributes for the OPM objects being connected by those attribute objects. Map non-attribute objects that have no states and object states of OPM to CPN places. Map the value(s) of an OPM object to CPN token(s). One or a set of tokens on a CPN place represents either the existence of an object or an object being at the state represented by that place. The former results from a CPN place mapped from an OPM object with no state and the token(s) on that place represents alternative objects. The latter results from a CPN place mapped from an OPM state. An object, as in object-oriented modeling, is defined by three parts, states, attributes and services (or methods, functions, or processes). By following the mapping scheme discussed above, a CPN token can capture the attribute and state part of an object definition explicitly. The service part of an object definition can be inferred if the CPN model created from the OPM follows certain naming convention. For example, an object’s service can be modeled as
an OPM process connected to its owning OPM object through an exhibition-characterization link. When such process is mapped to a CPN transition, the transition can be named by prefixing the OPM process name with the corresponding OPM object name to express the ownership of the object to the process. OPM structural links that have no effect on the system dynamics are not mapped to CPN. The details of the procedures for mapping an OPM/H model to a CPN model are as follows.

By following such rules, an executable CPN model can be developed as the one shown in Figure 9 through Figure 11. In this model, tokens are used to represent the instance of a system in SoS. Particularly, tokens at CPN place $S_{dn}$ represent the available reconnaissance systems and, before the operation of the SoS, they are all at off duty state. Such tokens represent the object instances using a list of values that encode the set of attributes that define their owning objects. Tokens at place Comm represent the communication systems and they are defined with similar list values. The number of tokens with a particular system ID at that place represents the capacity of that communication system. Similarly tokens and their numbers at place Exp represent the exploitation center and its capacity, respectively. Examples of token values and the information they encoded are presented in Table 3. The processing time required by Exploitation center and C&C is simulated by time delays added to the output arc of the Process and Decision transition, respectively. Such time delays are assumed to follow exponential distribution, the value of which is computed through a function (as shown in the left part of Figure 11). The CPN page “Ltran” models is a substitution transition that represents the data collection activity of the reconnaissance systems. The arrival of data items is assumed to follow an exponential distribution too.
Figure 9 CPN model converted from OPM

```plaintext
C_C
Uact
Dact
S_dn
S_wk
Ltran
Comm
Exp
Process

Actions
DATA

input (sid, dtl, col, stt, opt, dur)
output (upd_odt, proctime)
action
startProcC(sid, dtl, col, stt, opt, dur)

1'(1, A, [a], [g], 0, 540, 300)++
1'(3, A, [b], [g], 0, 540, 300)++
1'(4, A, [b], [g], 0, 540, 300)++
1'(5, A, [b], [g], 0, 540, 300)++
1'(6, B, [a, c], [g,h], 0, 1500, 1200)++
1'(7, B, [a, c], [g,h], 0, 1500, 1200)++
1'(8, C, [a, b, c], [h], 0, 120, 300)++
1'(9, C, [a, b, c], [h], 0, 120, 300+)

C_C
RTP
(input, dtl, col, stt, opt, dur)
output (upd_odt, proctime)
action
startProcC (sid, dtl, col, stt, opt, dur)

1'(10, C, [a, b, c], [h], 0, 120, 300+)

C_C
RTP
(input, dtl, col, stt, opt, dur)
output (upd_odt, proctime)
action
startProcC (sid, dtl, col, stt, opt, dur)

1'(11, D, [d,e,f], [g,h])++

Exp
GSTPT
GSFT

(exp_cp'(12, E, [e,f], [g,h])@0++)
exp_cp'(13, E, [e,f], [g,h])@0

C_C
RTP
(input, dtl, col, stt, opt, dur)
output (upd_odt, proctime)
action
startProcC (sid, dtl, col, stt, opt, dur)

1'(10, C, [a, b, c], [h], 0, 120, 300+)

Stran
DATA

Recved
DATA

Arrived
DATA

Processed
DATA

Figure 9 CPN model converted from OPM
fun expTime (mean: int) =
let
  val realMean = Real.fromInt mean
  val rv = exponential((1.0/realMean))
in
  floor (rv+0.5)
end;

fun startProcC (sid, dt, tsi, at, wt, pt) =
let
  val proc_time = expTime(cc_pt)
  val time_stamp = ModelTime.add(time(), ModelTime.fromInt(proc_time))
  val new_tsi = IntInf.toInt(time_stamp)
  val new_wt = wt + (intTime() - tsi)
  val new_pt = pt + proc_time
in
  ((sid, dt, new_tsi, at, new_wt, new_pt), proc_time)
end

Table 3 An example of token values and the corresponding information encoded for systems

<table>
<thead>
<tr>
<th>Token Value</th>
<th>ID</th>
<th>Duplicity</th>
<th>System Type</th>
<th>Capability Type</th>
<th>Communication on Duty Time</th>
<th>Break Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Place S_dn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1'(1, A, [a], [g], 0, 540, 300)</td>
<td>1</td>
<td>1</td>
<td>Shadow</td>
<td>EO</td>
<td>LOS</td>
<td>0</td>
</tr>
<tr>
<td>1'(3, A, [b], [g], 0, 540, 300)</td>
<td>3</td>
<td>1</td>
<td>Shadow</td>
<td>IR</td>
<td>LOS</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1'(6, B, [a, c], [g,h], 0, 1500, 1200)</td>
<td>6</td>
<td>1</td>
<td>Gray Eagle</td>
<td>EO, SAR</td>
<td>LOS, BLOS</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The system types are encoded in upper case letters according to the mapping shown in Table 2. Capabilities are encoded in lower case letters according to the mapping shown in Table 3. On duty time is the simulation time at which a system goes on duty.

Performance Assessment

The performance metrics, i.e., coverage area and latency, can be assessed through the simulation of the CPN model. The total coverage area (Cr) of the SoS is calculated from the simple equation below:

\[ Cr = \sum_{i=0}^{t} cr_i n_i \]

where \( cr_i \) is the coverage of a system of type \( i \), \( n_i \) is the number of systems of type \( i \) on duty, and \( t \) is the total types of systems on duty.

To get the information regarding what system (s) is on duty at a particular time, a CPN place, \( U_{act} \), is added to the CPN model shown in Figure 9, as an output place of transition Rd. As a result, each time the transition Rd fires, corresponding a system goes on duty, a duplicate token representing the system goes on duty is added to place \( U_{act} \). Since the token value contains information about the exact time the system goes on duty and the system is sure to go off duty after a pre-defined operation time, by collecting the tokens recorded at place \( U_{act} \), it can be...
known what systems are on duty at a particular time and therefore, the total coverage area of the SoS as a function of time. Figure 12 shows a plot of the coverage area as a function of simulated time based on a CPN simulation run. As can be seen the coverage area varies with time due to systems leaving (going off duty) and joining in (going on duty) the reconnaissance mission. The average coverage area and minimum coverage area during the entire operation time can also be obtained from the data.

The Data token has a list type of value, (sid, dt, tsi, at, wt, pt), where sid is the system type identifier, dt is the data type identifier (each data type binds to a particular capability, i.e., EO, IR, or SAR), tsi is the system simulation time, at is the initial arrival time of a data item, i.e., the time that the data is collected by a reconnaissance system, wt is the total waiting time, up to the current system time, that a data item spent, and pt is the total processing time, up to the current system time, that a data item spent. The difference between the current system simulation time tsi and the initial arrival time at is the latency (equals to wt + pt). The right part of Figure 11 shows the CPN function declaration that is used in transition Process and Decision for updating wt and pt. When a simulation run is completed all tokens representing data will land at place ActionPlan, representing that they are ready for decision makers. The feedback loop (adjusting action plans) are omitted in this model.

The total simulation time is controlled by the token value at place compTime. It has an initial value of 10080, which represents the period of a week length. Therefore, the average latency, average coverage area, and minimum coverage are all calculated based on the events that happened during a week length, which gives a reasonable estimation of the performance of the SoS in operation. In summary, by changing the set of tokens and their values at place S_dn, it can be determined what types of reconnaissance systems to be included in the system, the number of them to be included, and what capabilities they carry. By changing the number of tokens at place Exp and place Comm, the capacity of the exploitation center and communication system can be set, respectively. By changing the mean parameters of the respective exponential
distribution functions, the processing time of the exploitation center and C&C can be defined. The same holds for the rate that data is collected. Finally, by changing the token value at place compTime, the total simulation period can be set. Those are all the parameters to be set before a simulation run. A different set of these parameters represents a different SoS configuration. The performance of all SoS alternatives can be accessed in this way.

Concluding Remarks

The proposed modeling approach enhances the FILA-SoS meta-architecture generation framework with behavior analysis and performance assessment capabilities using OPM in conjunction with CPN. The key part of the methodology is to transform OPM to CPN models through well-defined algorithms. Simulation is an indispensable means to understand the interactions between system components and assess the system performance resulting from such interactions. Such behavior analysis capability greatly enhanced the capability of SoS domain manager in pre-evaluation of SoS and help them not only determine the feasibility of integrating chosen system to achieve desired capability but also can estimate how well the overall SoS will perform when deployed. Future work includes adding Markov Chains, graph theory, and Bayesian networks for further enhancing the analysis capability of the proposed executable modeling approach.
APPENDIX A: LIST OF PUBLICATIONS RESULTED AND PAPERS SUBMITTED FROM FILA-SoS 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)


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


