An Advanced Computational Approach to System of Systems Analysis & Architecting Using Agent-Based Behavioral Model

Final Technical Report SERC-2012-TR-021

September 30, 2012

Principal Investigator: Dr. Cihan Dagli, Missouri Science & Technology

Team Members

Dr. Nil Ergin, Assistant Professor, Penn State

Dr. John Colombi, Assistant Professor, Air Force Institute of Technology

Dr. George Rebovich, Director, Systems Engineering Practice Office, MITRE

Dr. Kristin Giammarco, Lecturer, Naval Postgraduate School

Paulette Acheson, Khaled Haris, Louis Pape, PhD Students,
Missouri University of Science &Technology

Report Documentation Page		*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **30 SEP 2012**	2. REPORT TYPE	3. DATES COVERED **00-00-2012 to 00-00-2012**
4. TITLE AND SUBTITLE **An Advanced Computational Approach to System of Systems Analysis & Architecting Using Agent-Based Behavioral Model**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Missouri Science & Technology,1870 Miner Circle,Rolla,MO,65409**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**		
13. SUPPLEMENTARY NOTES		
14. ABSTRACT		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**50**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

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

1 EXECUTIVE SUMMARY

A major challenge to the successful planning and evolution of an acknowledged System of Systems (SoS) is the current lack of understanding of the impact that the presence or absence of a set of constituent systems has on the overall SoS capability. Since the candidate elements of an SoS are fully functioning, stand-alone Systems in their own right, they have goals and objectives of their own to satisfy, some of which may compete with those of the overarching SoS. These system-level concerns drive decisions to participate (or not) in the SoS. Individual systems typically must be requested to join the SoS construct, and persuaded to interface and cooperate with other Systems to create the "new" capability of the proposed SoS. Current SoS evolution strategies lack a means for modeling the impact of decisions concerning participation or non-participation of any given set of systems on the overall capability of the SoS construct. Without this capability, it is difficult to optimize the SoS design.

The goal of this research is to model the evolution of the architecture of an acknowledged SoS that accounts for the ability and willingness of constituent systems to support the SoS capability development. Since DoD Systems of Systems (SoS) development efforts do not typically follow the normal program acquisition process described in DoDI 5000.02, the Wave Model proposed by Dahmann and Rebovich is used as the basis for this research on SoS capability evolution. The Wave Process Model provides a framework for an agent-based modeling methodology, which is used to abstract the non-utopian behavioral aspects of the constituent systems and their interactions with the SoS. In particular, the research focuses on the impact of individual system behavior on the SoS capability and architecture evolution processes. A proof of concept agent-based model (ABM) of the system interactions is developed and integrated with a genetic algorithm (GA) to explore the potential architectural design space, using a fuzzy associative memory (FAM) to evaluate candidate architectures for simulating SoS creation and evolution. The model evaluates the capability of the evolving SoS architecture with respect to four attributes: performance, affordability, flexibility and robustness.

The method is applied to an Intelligence Surveillance Reconnaissance (ISR) "acknowledged" SoS as an example domain. The agent-based model represents a System Program Office (SPO) personnel's interactions with the acknowledged SoS manager, and with the other Systems' representatives. An agent models each SPO's decision process and interactions.

Since the SoS is comprised of a subset of the available Systems, the participation of each System is modeled as a binary choice in a "chromosome" genetic algorithm representing the possible subsets of participating systems and their interactions with other Systems within the SoS. The genetic algorithm approach allows a more thorough exploration of the architectural "space" composed of all the possible subsets of Systems and interactions than typical, biased, preconceived human selected subsets.

Finally, the choice of achievable configurations from the various candidate architectures represented by the genetic algorithm generated chromosomes is made by a rule-based fuzzy associative memory. A proposed method is presented for developing 1) desired attribute membership functions, and 2) rules for combining sub-element values to achieve an overall architecture evaluation that can be ranked for selection.

The model elements are integrated into a toolset that can include the environment in which the agents operate, to select better architectures for successive waves of development. These “waves” of development may coincide with annual funding increments and/or major reviews.

Initial integration of the GA and FAM was demonstrated in the ABM framework. Additional effort is planned to improve and modularize the agent models and the interactions among the agents, improve the interfaces among the GA, FAM and ABM components, improve the GA evolution algorithms, add stakeholder participation to creation of the fuzzy evaluations, and to create a better user interface for an executable tool set for future modeling in multiple domains.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

2 INTRODUCTION

The goal of this research is to model System of Systems (SoS) acquisition and architecture evolution based on the Wave Process Model. Agent-based modeling methodology is used to abstract behavioral aspects of the acquisition process. In particular, the research aims to focus on the impact of individual system behavior on SoS architecting and acquisition processes. Details of the model are provided in the following sections.

2.1 PROBLEM/MOTIVATION

2.1.1 PROJECT DESCRIPTION

Develop, validate, and pilot agent-based modeling Methods, Tools and Processes (MTPs) that support predictions about the properties of an acknowledged SoS, applied early in the life cycle when there is high uncertainty and ambiguity about SoS requirements, architecture, DoD Acquisition guidance and implementation technologies based on the Wave Process Model. Tooling should not require extensive knowledge of agent-based modeling and underlying algorithms. The agent based model implements a framework that can include environment, negotiation models, genetic algorithms and fuzzy associative memory for evaluation of candidate architecture evolutions.

2.1.2 VALUE PROPOSITION

Early insight into likely properties of acknowledged SoS meta architecture will inform technological decisions about requirements, architecture, and implementation technologies and aid resulting acquisition decisions.

2.2 RESEARCH OBJECTIVES

2.2.1 OBJECTIVES

The goal of this research is to develop a proof of concept agent based model tool suite for SoS systems simulation for architecture selection and evolution. An Intelligence, Surveillance and Reconnaissance (ISR) SoS, consisting of numerous individual systems is used as a domain example to demonstrate the framework of the agent based model tool suite.

2.2.2 METHODOLOGY

Policies on architecting in the DoD continue to evolve, although perhaps at a slower pace than in the past. As a result, the modeling of architecture development and evolution is not settled science. This is particularly true in SoS settings (ASD(NII), 2009). Existing analysis methodologies and tools narrow the scope of the SoS problem space by invoking the assumption that there is a limited set of solutions, solely or primarily driven by technical performance considerations. However, the SoS problem boundary includes integration of technical systems as well as cognitive and social processes, which alter system behavior (Dauby & Upholzer, 2011). As mentioned before, most system architects assume that SoS participants exhibit nominal behavior (utopian behavior), but deviation from nominal motivation leads to complications and disturbances in systems behavior. It is necessary to capture the behavioral dimension of SoS architecture to be able to represent the full problem space to guide SoS architecting and analysis phase (Dauby & Upholzer, 2011). Evaluation of architectures also lends itself to a fuzzy approach because the criteria are frequently non-quantitative, or subjective, or based on unknowable future conditions, such as "robustness." Finally, since one of the current problems with SoS composition is lack of imagination, the genetic algorithm approach can help to explore the architecture space more fully.

Agent based models (ABMs) consist of a set abstracted entities referred to as agents, and a framework for simulating agent decisions and interactions. Agents may have their own goals and are capable of perceiving changes in the environment (Acheson, 2010). System behavior (global behavior) emerges from the decisions and interactions of the agents. The approach provides insight into complex, interdependent processes. Agent-based modeling methodology has several benefits over other modeling techniques; it captures emergent patterns of system behavior, provides a natural description of a system composed of behavioral entities and is flexible for tuning the complexity of the entities (Bonabeau, 2002). The methodology is used in a wide range of application domains including financial markets [Ergin et al], homeland security applications (Weiss, 2008) and autonomous robots (Dudenhoeffer & Jones, 2000).

The goal of this research is to model the SoS architecture evolution based on the Wave Process Model. Agent-based modeling methodology is used to abstract behavioral aspects of the acquisition process. In this project, it is assumed that the Systems are the agents. The System Agents embody themselves and the people (individual stakeholders) responsible for them. The wave model applies to an acknowledged SoS, thus there is a specific agent responsible for the SoS, and that agent influences the other System Agents. An initial SoS mission is already determined and funds are allocated to the mission with a responsible organizational entity (Director Systems and Software Engineering, 2008). The structure of the wave model is depicted in Figure 1. (Dahmann, Rebovich, Lane, Lowry, & . Baldwin, 2011)

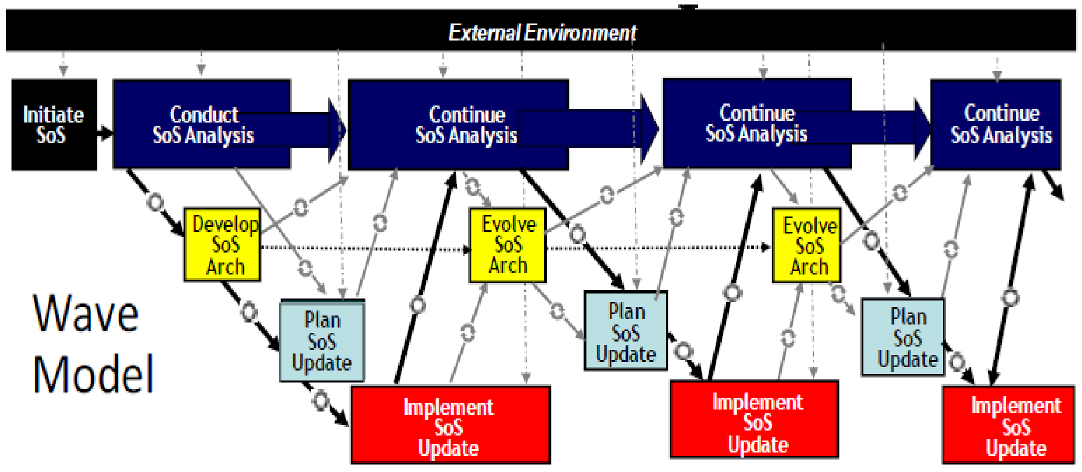


Figure 1. The Wave Model of SoS initiation, engineering, and evolution

2.3 RT Schedule/Plan

Task-1: Analyze Acknowledged SoS domain
Task-2: Environment Model Design
Sub-task 2.1: Based on task 1, identify relevant rules of engagement
Sub-task 2.2: Based on task 1, determine environment attributes (changes in national priorities, changes in SoS funding, changes in individual system funding …)
Task-3: Agent Model Design
Sub-task 3.1: Identification of Agent Types (SoS agent and individual system agents) and Attributes
Sub-task 3.2: Identification of SoS Agent Behavior
Sub-task 3.3: Selection of Agent Architecture
Task-4: Implementation

This phase implements the agent-based SoS model designed in previous tasks using an agent-based simulation toolkit for a specific SoS domain. Once the simulation is built and verified, a series of "what-if" experiments will be conducted by varying the parameters and assumptions of the model.

3 Background: SoS Engineering

System of Systems (SoS) engineering deals with planning, analyzing, organizing and integrating the capabilities of independent systems into a unique set of SoS capabilities (Director Systems and Software Engineering, 2008). Systems of systems differ from traditional systems in ways that require tailoring of the system engineering process. The Department of Defense (DoD) 5000.2 framework for acquiring systems is not suitable for SoS development anymore. This is mainly due to the fact that SoS component systems are independent and have their own functionality, development processes, funding and operational missions. In addition, changes in external environment such as funding, national priorities can alter the dynamics of the acquisition process [Dahmann et al., 2011], [Lane and Dahmann, 2008]. Therefore, SoS engineering needs to consider change as a critical element of the process.
Several acquisition models have been discussed in the literature for SoS development. Evolutionary acquisition models are more suitable to SoS development as they emphasize stakeholder involvement, interim milestones, increased iteration and concurrent development (Creel & Ellison, 2008). The Systems Engineering of SoS (SoS SE) model is developed by the US Department of Defense to identify the core elements of the SoS acquisition (Director Systems and Software Engineering, 2008). The Incremental Commitment Model (Lane & Dahmann, 2008) is a risk driven framework that can be tailored for SoS development.

The Wave Model (Dahmann, Rebovich, Lane, Lowry, & . Baldwin, 2011) maps SoS Systems Engineering (SE) model's core elements to a series of time-sequenced iterative process to guide implementation of the framework for practitioners of SoS development. The wave model or bus stop approach is a development approach that is similar to the effect of periodic waves crashing at the shore or a bus that periodically stops at a specific location. The SoS has specific places in the development where it can accept updates from the individual systems. Individual systems can plan their deliveries to coincide with the SoS 'bus-stops' or can evaluate the effect of missing a planned SoS wave. Figure 1 illustrates the major elements of the Wave Process Model. The steps in the model are briefly introduced below as background information. For further details refer to(Dahmann, Rebovich, Lane, Lowry, & . Baldwin, 2011).

3.1 Steps in the Wave Model

3.1.1 Initiate SoS

This step involves understanding the SoS objectives and operational concept (CONOPS), as well as gathering information on core systems to support desired capabilities.

3.1.2 Conduct SoS Analysis

This step establishes an initial SoS baseline architecture for SoS engineering based on SoS requirements space, performance measures, and relevant planning elements.

3.1.3 Develop and Evolve SoS Architecture

This step evolves the initial SoS baseline and develops the SoS architecture. The SoS architecture includes individual systems, key SoS functions and interdependencies among systems. The architecture identifies necessary changes in contributing systems in terms of interfaces and functionality in order to implement the SoS architecture.

3.1.4 Plan SoS Update

This step plans for the next SoS upgrade cycle based on the changes in external environment, SoS priorities, options and backlogs.

3.1.5 Implement SoS Update

This step establishes a new SoS baseline based on SoS level testing and system level implementation. This step is the end of wave cycle or 'bus-stop' where updates from individual systems can be integrated into the SoS.

3.1.6 Continue SoS Analysis

This step is the beginning of the next wave cycle and continuous to analyze the current SoS architecture for future SoS evolution.

This research builds on the Wave Process Model to abstract behavioral aspects of the SoS acquisition process. An agent-based model of the process is discussed to analyze the impact of individual system behavior on the overall SoS architecture evolution. It is envisioned that this type of model will help us in understanding the intricate dynamics of the SoS development and improve acquisition process.

4 SoS Analytics Methodology

4.1 Acknowledged SoS Agent-Based Architecture

Acknowledged SoS have objectives, management and funding but not complete authority over the constituent systems(Director Systems and Software Engineering, 2008). Therefore, the SoS development depends on collaboration and agreements with individual systems rather than simply a top-down request from the SoS manager. This collaborative nature impacts the evolution of the SoS architecture.

The agent-based methodology is suitable for analyzing the behavioral aspects of the acquisition process as different type of agents can capture various dynamics of the SoS engineering in an integrated analysis framework. In particular, independent systems' behavior can be mapped to a set of agents and SoS manager's engineering activities can be abstracted to another type of agent. The following subsections outline the underlying agent architecture which forms the basis for modeling the SoS Agent and the individual System Agents. Then the proposed agent based model for Acknowledged SoS Engineering is discussed in detail in Section 4.2.

4.1.1 Agent Architecture for Cooperative Problem Solving

In distributed project coordination such as SoS architecting, task coordination between agents is an important issue. A formal cooperation model for multi-agent systems is developed in (Brazier, Dunin-Keplicz, Jennings, & and Treur, 1996) (Brazier, Jonker, & Truer, 1997) and which is a refined agent model of Jenning's model of cooperation(Jennings, 1995). In this model each agent performs several generic tasks to cooperate with other agents; observes the world, manages interaction with the world, maintains information on other agents, manages interaction with other agents, and manages joint activities. For detailed information on the generic model refer to(Brazier, Dunin-Keplicz, Jennings, & and Treur, 1996). This generic model is refined for the SoS acquisition and architecting problem which carries similar coordination and cooperation characteristics. Figure 2 outlines the agent architecture components and flow of information among components.

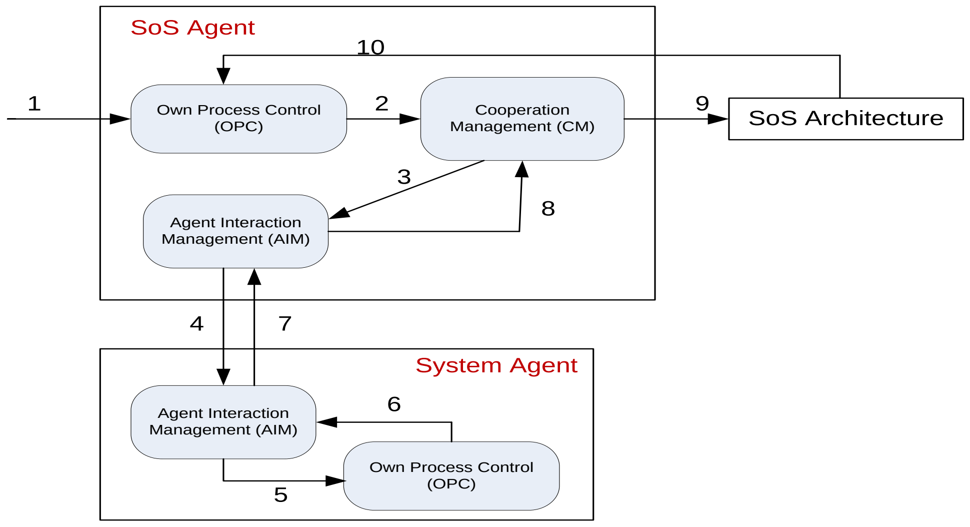

SoS Agent
10
1
Own Process Control (OPC)
2
Cooperation Management (CM)
9
SoS Architecture
3
8
Agent Interaction Management (AIM)
4
7
System Agent
6
Agent Interaction Management (AIM)
Own Process Control (OPC)
5

Figure 2. Agent Architecture

The Own Process Control Module determines the goals of an agent based on its motivations, priorities, and deadlines. Given the desired SoS capabilities, SoS performance parameters and initial baseline SoS Architecture (1), this module formulates the target measures the SoS architecture and passes this information to the Cooperation Management Module (2). Cooperation Management module is responsible for all tasks related to SoS architecting, commitment and cooperation management of individual systems. Figure 3 outlines the details of this module.

Based on the initial SoS architecture baseline, Cooperation Management Module first generates connectivity request for individual systems to the SoS architecture. This request contains interface requirement for individual system to integrate to other systems, performance requirement, deadline to deliver the request, and the funding for the request. This information is passed to Agent Interaction Management Module (3) which is responsible for managing interaction with individual systems. SoS AIM module distributes the requests to individual system alternatives (4). Each alternative system has its own Agent Interaction Module which passes SoS request information to its Own Process Control module (5). Individual system OPC module evaluates the request based on its own motivations and replies back to the SoS Agent through its Agent Interaction Module (6, 7).

The reply contains information on individual system cooperation, performance, capabilities and if the system decides to cooperate, information on probability that the SoS request will be available at the requested deadline. Information received from all alternative systems is passed back to the SoS Cooperation Management module (8) where SoS Agent evaluates the current SoS architecture at time T (9). At wave interval T, the SoS architecture is implemented by integrating individual systems that agree to cooperate and the current SoS architecture is evaluated against the initial SoS baseline architecture. Gaps are identified and feedback is sent to SoS Own Process Control (10) which is also responsible for updating mission, acquisition wave interval, and SoS program parameters based on the Environmental factor changes and Architecture gap analysis.

The process iterates for several cycles of wave. Section 4.2 provides the details of the SoS engineering model in mathematical structure. The analytical tools associated with the SoS engineering model are also discussed in detail in the following subsections.

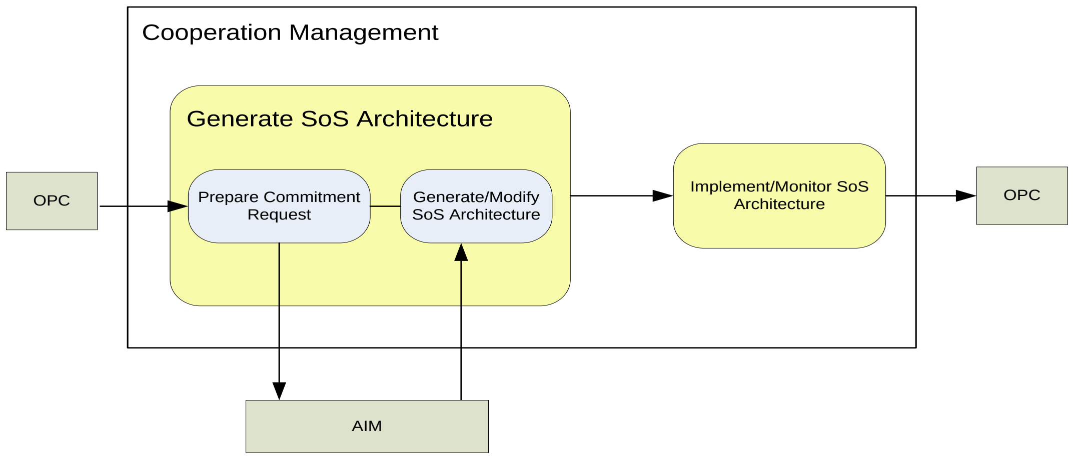

Cooperation Management
Generate SoS Architecture
OPC
Prepare Commitment Request
Generate/Modify SoS Architecture
Implement/Monitor SoS Architecture
OPC
AIM

Figure 3. Cooperation Management

4.2 SoS Agent Analytical Tools and Models

4.2.1 Proposed Agent Based Model

The proposed Agent-Based Model consists of a generic SoS development, genetic algorithm, fuzzy assessor and an executable model, shown in Figure 4. The generic SoS development is based on the Wave Model. The genetic algorithm creates the initial SoS meta-architecture. The fuzzy assessor qualitatively evaluates the possible SoS meta-architectures. Finally, all of these are implemented into an agent-based model that executes and provides the resultant SoS meta-architecture evolution based on the initial inputs.

The model variables and associated variable relationships are outlined in the following mathematical representation. The mathematical representation outlines the elements abstracted from the Wave SoS Engineering model and provides the general structure for the agent-based model The implementation of the model further refines this general framework which will be discussed in detail in Section 4. The mathematical representation is organized based on main agents: SoS Agent and individual System Agents. Following assumptions are used to set the stage:

- Agents represent individual Systems and SoS
- The System Agents embody themselves and the people (individual stakeholders) responsible for them
- The wave model applies to acknowledged SoS, thus there is a specific agent responsible for the SoS and for controlling/influencing the individual System Agents
- Some baseline legacy system is available initially (functional and physical architecture)
- An initial SoS mission is already determined
- SoS architecting is abstracted in the model as an interface link problem of integrating independent systems

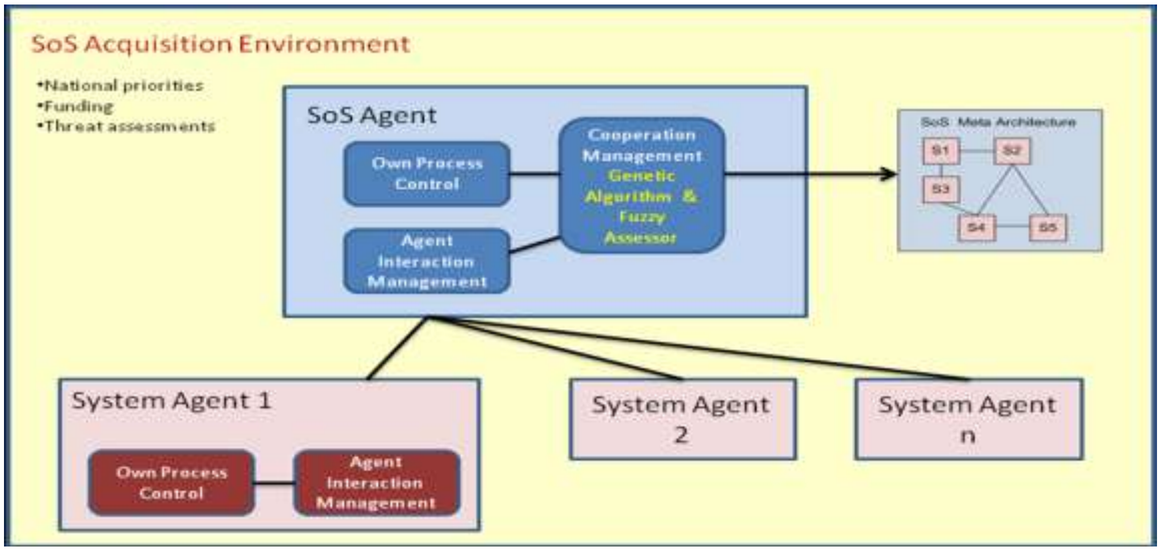
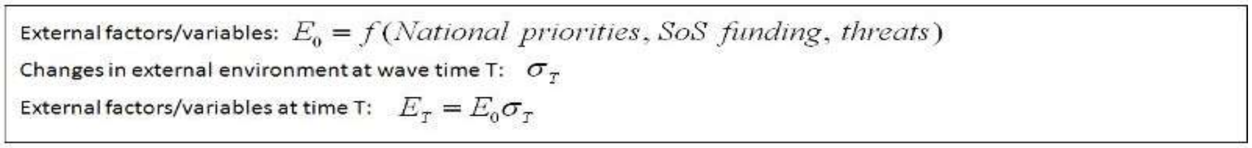


Figure 4. Agent-based SoS Acquisition and Architecting Model

4.2.1.1 SoS Acquisition Environment

The SoS Agent and the individual System Agents are influenced by the changes in the SoS acquisition environment. Thus the environment model includes external factors/variables such as national priorities, threats, and SoS funding. As the SoS acquisition progresses through wave cycles, these variables are updated to reflect acquisition environment changes. Table 1 summarizes the model elements in mathematical notation.

Table 1. SoS Acquisition Environment


External factors/variables: $E_0 = f(National\ priorities, SoS\ funding, threats)$

Changes in external environment at wave time T: σ_T

External factors/variables at time T: $E_T = E_0\sigma_T$

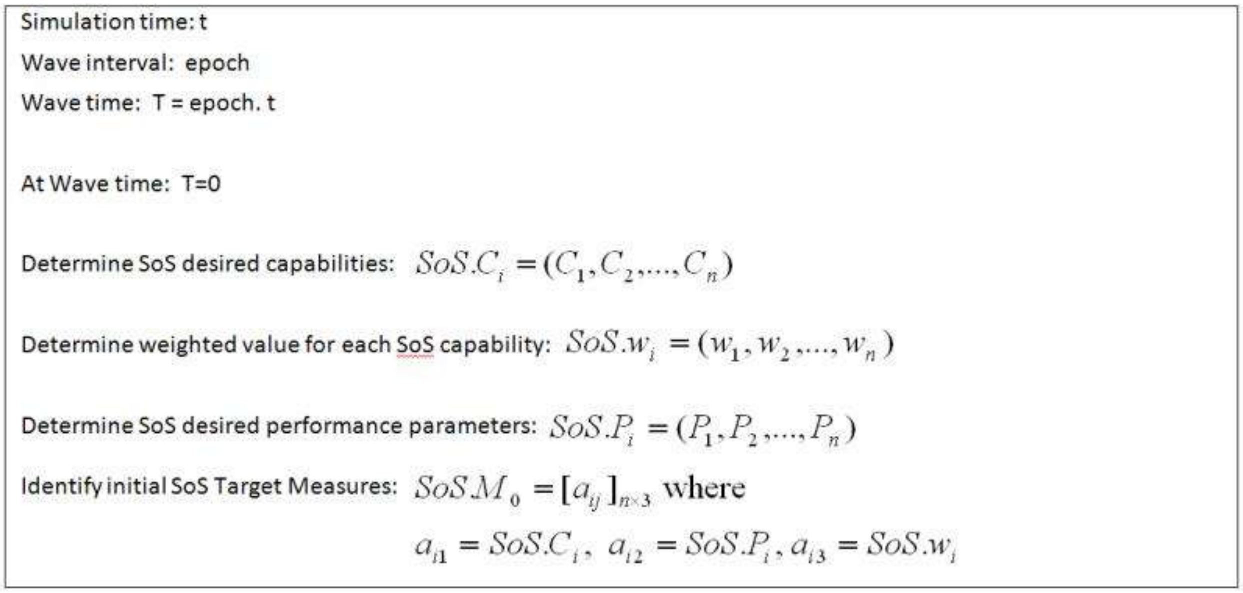
4.2.1.2 SoS Agent Behavior

SoS Agent is responsible for the overall SoS engineering activity and coordinates with individual System Agents to achieve the desired SoS mission. In the model, it is assumed that an initial SoS mission is already determined and initial baseline SoS architecture is available. The SoS Agent follows the six core SoS engineering activities outlined in the Wave Process Model in order to develop the SoS. The SoS architecture evolves based on the behavior of individual systems as well as changes in the external environment.

4.2.1.3 Initiate SoS

During the initialization phase, the wave interval, the time interval from one wave to next, is determined. At each wave interval time, the SoS Agent identifies SoS target measures which comprises desired SoS capabilities and SoS performance parameters for these capabilities in order to meet mission objectives. Since some of the capabilities may have higher priority levels than others, weighted value of each capability is also identified at this phase. Table 2 summarizes the abstracted model elements in mathematical notation.

Table 2. Initiate SoS


Simulation time: t

Wave interval: epoch

Wave time: T = epoch. t

At Wave time: T=0

Determine SoS desired capabilities: $SoS.C_i = (C_1, C_2, ..., C_n)$

Determine weighted value for each SoS capability: $SoS.w_i = (w_1, w_2, ..., w_n)$

Determine SoS desired performance parameters: $SoS.P_i = (P_1, P_2, ..., P_n)$

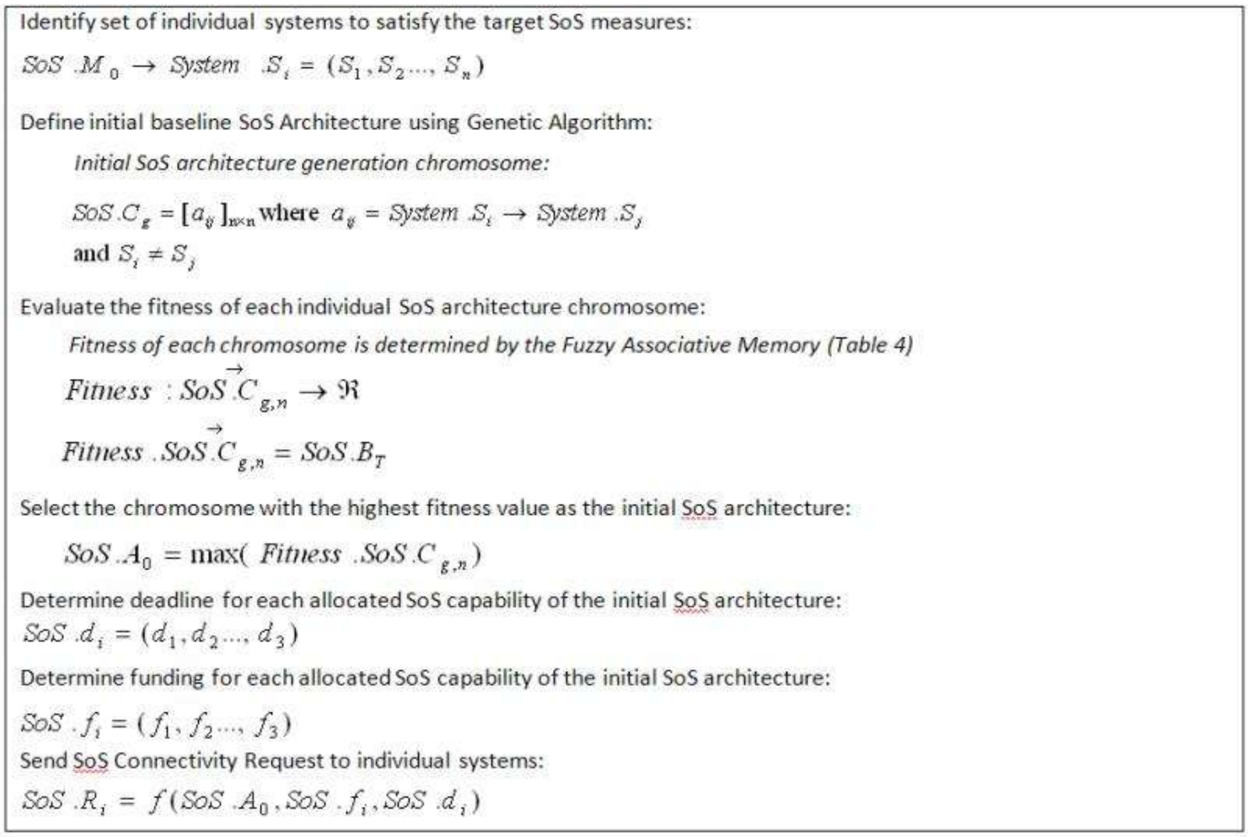
Identify initial SoS Target Measures: $SoS.M_0 = [a_{ij}]_{n\times 3}$ where

$$a_{i1} = SoS.C_i,\ a_{i2} = SoS.P_i, a_{i3} = SoS.w_i$$

4.2.1.4 Conduct SoS Feasibility Analysis

The SoS Agent allocates SoS capabilities to individual systems or group of systems. This allocation defines a baseline SoS architecture which identifies individual systems and interfaces necessary to achieve the SoS target measures. Architecture algorithm which is based on Genetic Algorithms represents alternative SoS architectures as chromosomes. This analytical tool is described in detail in Section 3.2.2. The Fuzzy Associative Memory determines the fitness of each chromosome and the best alternative is selected as the initial SoS baseline architecture for the acquisition wave. The Fuzzy Associative Memory analytical tool is described in detail in Section 3.2.1. Program management measures such as schedule, funding are also identified for the selected SoS architecture. The SoS baseline architecture and program measures information is sent to individual systems as a connectivity request to the SoS architecture. Individual systems should evaluate whether they can develop the requested interface with other systems and capabilities in the given deadline and funding. Table 3 summarizes the abstracted model elements in mathematical notation.

Table 3. Conduct SoS Analysis



Identify set of individual systems to satisfy the target SoS measures:

$SoS.M_0 \rightarrow System.S_i = (S_1, S_2 ..., S_n)$

Define initial baseline SoS Architecture using Genetic Algorithm:

Initial SoS architecture generation chromosome:

$SoS.C_g = [a_{ij}]_{n \times n}$ where $a_{ij} = System.S_i \rightarrow System.S_j$

and $S_i \neq S_j$

Evaluate the fitness of each individual SoS architecture chromosome:

Fitness of each chromosome is determined by the Fuzzy Associative Memory (Table 4)

$Fitness : SoS.\vec{C}_{g,n} \rightarrow \Re$

$Fitness . SoS.\vec{C}_{g,n} = SoS.B_T$

Select the chromosome with the highest fitness value as the initial SoS architecture:

$SoS.A_0 = \max(Fitness . SoS.C_{g,n})$

Determine deadline for each allocated SoS capability of the initial SoS architecture:

$SoS.d_i = (d_1, d_2 ..., d_3)$

Determine funding for each allocated SoS capability of the initial SoS architecture:

$SoS.f_i = (f_1, f_2 ..., f_3)$

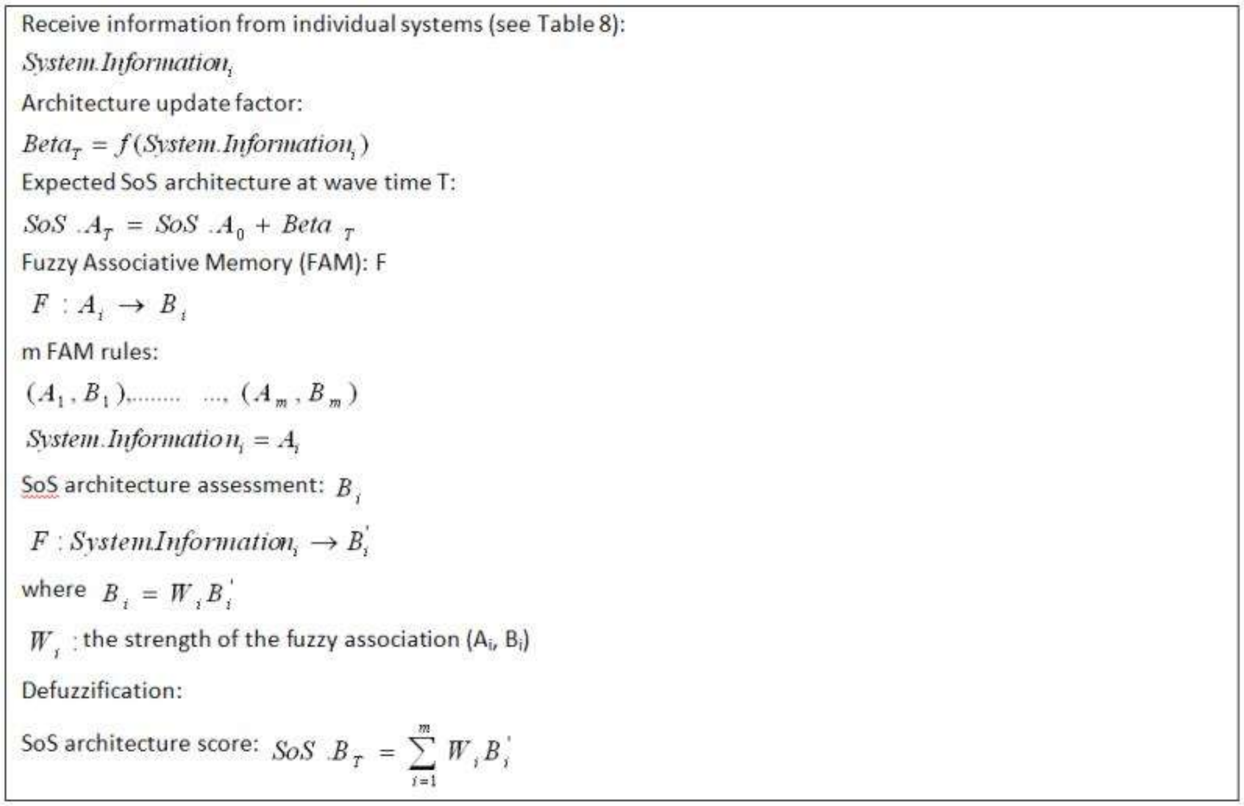
Send SoS Connectivity Request to individual systems:

$SoS.R_i = f(SoS.A_0, SoS.f_i, SoS.d_i)$

4.2.1.5 Develop and Evolve SoS Architecture

The SoS Agent updates the baseline SoS architecture based on information received from individual systems. Individual systems may decide to cooperate at the requested deadline, may decide to cooperate at a later time or may decide to not cooperate at all depending on their motivation. At this step, based on information received from individual systems, the expected SoS architecture at the end of the wave cycle is updated. The SoS Agent has a Fuzzy Assessor or fuzzy associative memory (FAM)which maps desired target measures to SoS architecture score/rating. At this step, the Fuzzy Assessor determines architecture score for the expected SoS architecture at wave time T based on the information received from individual systems. This SoS architecture score is used later in gap analysis to plan for the next SoS architecture update. Table 4 summarizes the abstracted model elements in mathematical notation.

Table 4. Develop and Evolve the SoS Architecture



Receive information from individual systems (see Table 8):

$System.Information_i$

Architecture update factor:

$Beta_T = f(System.Information_i)$

Expected SoS architecture at wave time T:

$SoS.A_T = SoS.A_0 + Beta_T$

Fuzzy Associative Memory (FAM): F

$F : A_i \rightarrow B_i$

m FAM rules:

$(A_1, B_1), \ldots\ldots\ \ldots, (A_m, B_m)$

$System.Information_i = A_i$

SoS architecture assessment: B_i

$F : System.Information_i \rightarrow B_i'$

where $B_i = W_i B_i'$

W_i : the strength of the fuzzy association (A_i, B_i)

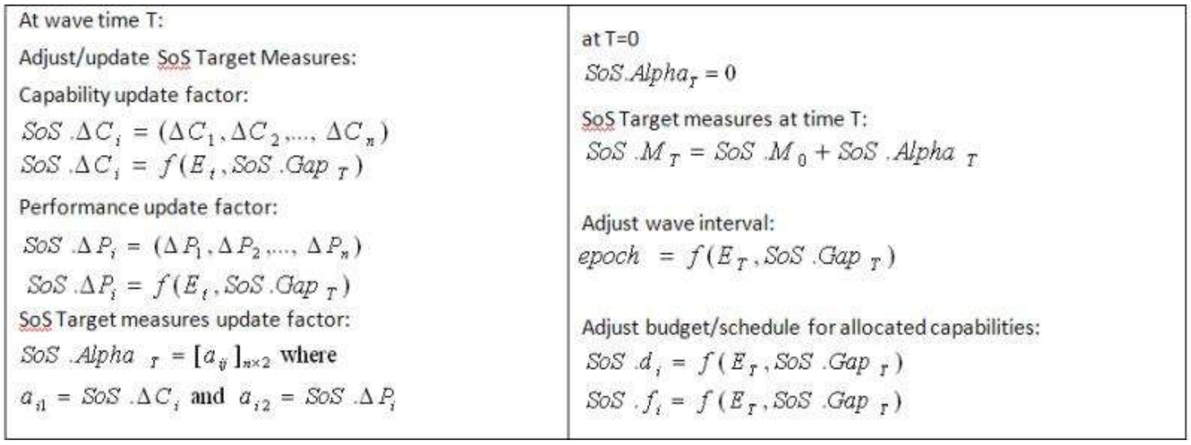
Defuzzification:

SoS architecture score: $SoS.B_T = \sum_{i=1}^{m} W_i B_i'$

4.2.1.6 Plan SoS Update

At the end of the wave cycle, the SoS Agent evaluates changes in the external environment. The SoS target measures and wave interval for the next cycle is updated based on environment changes and architecture gaps analysis. The gap analysis is also conducted at the end of the wave cycle during the SoS implementation step which is described in the following step. Table 5 summarizes the model elements in mathematical notation.

Table 5. Plan SoS Update



At wave time T:

Adjust/update SoS Target Measures:

Capability update factor:

$SoS.\Delta C_i = (\Delta C_1, \Delta C_2, \ldots, \Delta C_n)$
$SoS.\Delta C_i = f(E_t, SoS.Gap_T)$

Performance update factor:

$SoS.\Delta P_i = (\Delta P_1, \Delta P_2, \ldots, \Delta P_n)$

$SoS.\Delta P_i = f(E_t, SoS.Gap_T)$

SoS Target measures update factor:

$SoS.Alpha_T = [a_{ij}]_{n \times 2}$ where

$a_{i1} = SoS.\Delta C_i$ and $a_{i2} = SoS.\Delta P_i$

at T=0

$SoS.Alpha_T = 0$

SoS Target measures at time T:

$SoS.M_T = SoS.M_0 + SoS.Alpha_T$

Adjust wave interval:

$epoch = f(E_T, SoS.Gap_T)$

Adjust budget/schedule for allocated capabilities:

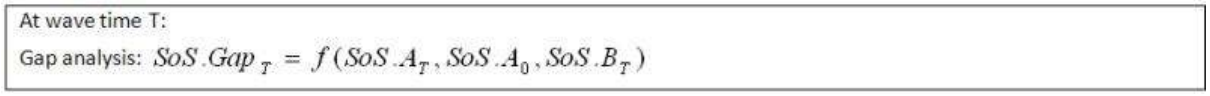
$SoS.d_i = f(E_T, SoS.Gap_T)$

$SoS.f_i = f(E_T, SoS.Gap_T)$

4.2.1.7 *IMPLEMENT SOS*

At the end of the wave cycle, the current SoS architecture is evaluated against initial SoS baseline architecture to identify the functionality gaps. The SoS architecture score determined by the fuzzy assessor is used in the analysis to identify performance gaps. This step is an input to planning SoS update step. Table 6 summarizes model elements in mathematical notation.

Table 6. Implement SoS Architecture



At wave time T:

Gap analysis: $SoS.Gap_T = f(SoS.A_T, SoS.A_0, SoS.B_T)$

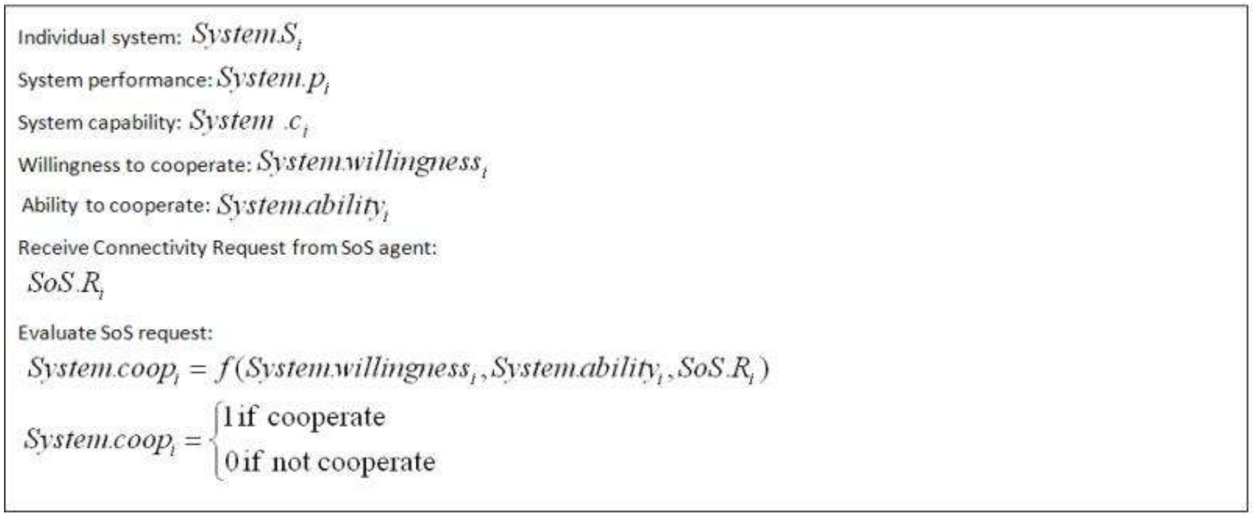
4.2.1.8 $SoS.Gap_T = SoS.M_T - SoS.A_T$ *CONTINUE SOS ANALYSIS*

The next wave cycle of the SoS development starts once the SoS target measures and wave interval time are updated.

4.2.1.9 *INDIVIDUAL SYSTEM BEHAVIOR*

Individual systems receive request for connectivity to SoS architecture. Since each system is independent and has its own goals and motivations, the system has the option to cooperate or not cooperate with the SoS Agent. The decision depends on several factors including system's willingness to cooperate which measures the degree of selfishness of the individual system to be part of the SoS, and system's ability to cooperate which depends on system's resources that will allow the system to be part of the SoS. If individual system decides to cooperate, it sends information to the SoS Agent on the probability of meeting the requested capability at the given deadline. If an individual system decides to not cooperate, it has the option of requesting a later deadline to provide the capability. Table 7 and Table 8 summarize the abstracted model elements in mathematical notation for individual systems.

Table 7. Evaluate SoS Connectivity Request



Individual system: $System.S_i$

System performance: $System.p_i$

System capability: $System.c_i$

Willingness to cooperate: $System.willingness_i$

Ability to cooperate: $System.ability_i$

Receive Connectivity Request from SoS agent:

$SoS.R_i$

Evaluate SoS request:

$$System.coop_i = f(System.willingness_i, System.ability_i, SoS.R_i)$$

$$System.coop_i = \begin{cases} 1 \text{ if cooperate} \\ 0 \text{ if not cooperate} \end{cases}$$

Table 8. Reply back to SoS Agent

If $System.coop_i = 1$

$$System.Information_i = (System.c_i, System.p_i, System.av_i)$$

where $System.av_i = P(SoS.R_i)$

else

Time to cooperate:

$System.cooptime_i = t \text{ where } t > SoS.d_i$

4.2.2 MULTI-STAKEHOLDER, MULTI CRITERIA FUZZY ARCHITECTURE ASSESSMENT

4.2.2.1 MATHEMATICAL BASIS FOR A FUZZY ASSESSMENT

In the baseline, acknowledged SoS architecture model, there are **n** Systems that the SoS Agent asks to participate to provide a new Capability. The universe is bigger than the **n** Systems asked to participate in the SoS. If all the solicited Systems could do everything the SoS Agent might ask, there would be little to be gained from modeling alternative architectures or their development. When some Systems cannot, or refuse to, contribute what is asked, then the model becomes interesting.

4.2.2.2 SYSTEM SOLICITATION AND PARTICIPATION

For whatever System internal reasons, the simplest model comes down to an initialprobability that each individual System will agree (or be able) to participate of **p** $(0 \le p \le 1)$. In the case of a refusal, the System might be having internal problems (technical, managerial, staffing, schedule, etc) that prevent it from being able to take on any additional tasks; they might think the proposed SoS won't work, therefore choose not to divert any attention to it; or the stakeholders for their primary mission may not allow such diversion of resources; or they may want to participate, but not have the capability to do so. For whatever internal reasons, a System can choose *not* to participate; this is represented by a "0" in the SoS chromosome representing the architecture, discussed further in the next section. There are **n** possible Systems$_i$. The expected number of systems to participate on average in each instantiated SoS architecture is **pn** . The decision itself can be determined through complicated, multiparty negotiations, or in this first example of the model framework, by calling a random number generator (evenly distributed between zero and one), to decide if the System will participate when the random number is less than **p** .

The probability that two Systems achieve an interface between them can be independently arrived at through the agent interactions, or simply modeled as **q** $(0 \le q \le 1)$. The two Systems might both be willing to participate, but still only be able to achieve the inter face between them with probability **q** . The successful interface is represented by a "1" at that interface position in the SoS chromosome. Unless both Systems are already participating, q doesn't come into play, because an interface can't exist with a non-participating System (at the very least, it doesn't do any good). If all Systems participate, the maximum number of interfaces is **n(n-1)/2** . If only **pn** Systems participate, and **q** is the likelihood of having an interface with another participating system, then **pqn(n-1)/2** is the *expected* number of contributing interfaces. Placing a "1" in the chromosome represents an interface between the ijth Systems if the Systems both participate, *and* a new random number as described above is less than **q** when generating a chromosome bit, in this simplified first iteration of the model.

4.2.2.3 EVALUATION ATTRIBUTES

Any number of architecture attributes may be used for evaluation, but peoples' attention span, as well as the number of independent attributes, typically cannot sustain large numbers. A program's (System's) current attribute status is often presented as color coded; e.g., plotted on stop light charts, or displayed on Kiviat charts to let management compare alternative architectures, or from one review to the next. Attribute evaluations are ideally suited to fuzzy logic approaches because of the difficult nature of boundaries between evaluation regions. Four attributes are use to evaluate the overall SoS architecture in this model: **Performance**, **Affordability**, Developmental **Flexibility**, and Operational **Robustness** are the attributes chosen for this phase of the model.

It is unlikely that any set of SoS attributes could be completely orthogonal(Dauby & Dagli, 2011). The attributes themselves are fuzzy, and they overlap somewhat in that they frequently evaluate in a correlated way; i.e., good programs are frequently good in many areas, etc. The gradations of the attributes are also "fuzzy," in that an exact boundary between any described gradations from bad to good is difficult to define. Uncertainty about what is meant (at the boundaries) by any of the attributes themselves (or the gradations within them) implies that some observers might evaluate an SoS as Affordable, and some observers might interpret that same SoS as *not* Affordable - based on a perceived but non-quantifiable difference in risk, margins, value, recent expenditure rate, gut feel, sizing up of program personnel personalities, etc. In fact, these differences are always there and managers are employed to evaluate them and manage with them both individually and as a gestalt. The range of uncertainty can be represented by an overlap of each gradation of the attribute evaluation; the demarcation between the evaluations is what is Fuzzy. A Fuzzy associative memory allows operation on the fuzzy concepts in a mathematically precise way. In the demonstration, all the attributes are normalized to a range of goodness from **Unacceptable**, **Marginal**, **Acceptable** and **Exceeds,** shown in Figure 5. One way to look at the membership functions is that some evaluators might call a program that is mathematically in the center of the range from zero to one (worst to best) of "goodness" as Acceptable, some might choose to call the same program Marginal. An equally valid way to consider the fuzziness is that *today* the program might be Marginal, but *next week* it will be Acceptable with only miniscule changes in the various subcomponents that make up its evaluation. It is difficult to precisely define the line separating the gradations, therefore one doesn't try(Dauby & Dagli, 2011). One simply defines the adjacency regions in a fuzzy way.

4.2.2.4 Membership Functions

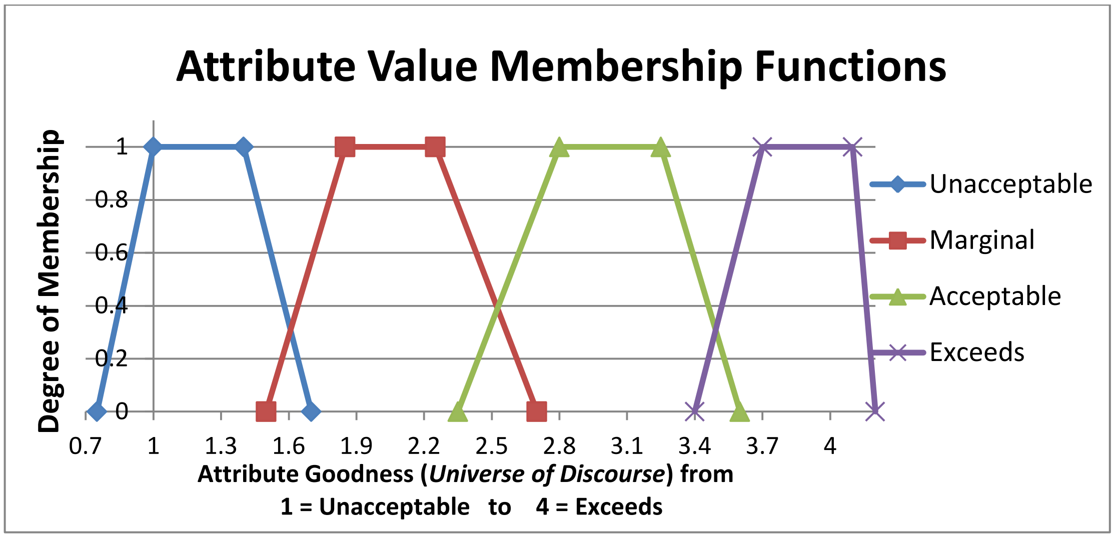


Figure 5. Attribute value membership functions

The membership functions shown in Figure 5 have overlap between each adjacent value, with slightly more overlap between Marginal and Acceptable than for the other boundaries. Slightly more membership opportunity exists in the Marginal and Acceptable categories than in Unacceptable and Exceeds. In this example, there is no overlap between categories that are not adjacent. This selection of membership functions represents a consensus among the RT-37 collaborators. The type of guided discussions among that group, with experience on numerous Programs and SoSs over most of the typical life cycle, that led to the shapes in Figure 5 and guidelines in Table 11, will be embedded in follow on efforts with actual stakeholders. One of the goals of follow on work is to improve the selection process of the membership functions for architecture attribute evaluations.

One purpose of the overarching ABM is to follow SoS development across funding epochs, where the SoS manager can observe progress in the Systems as well as changes in the environment (threat, funding, technology, etc.) and change guidance or participants in the next epoch.

4.2.2.5 Attribute Evaluations

All the SoS attributes were simplified for this task to have the same membership function shapes, labeled with the same four gradations. This is not a requirement; the membership functions can be as complex as it makes sense. The next step is to drive to an overall SoS evaluation through combination of the individual attribute values. The Fuzzy Toolbox in MATLAB allows for different evaluation scoring regions and various membership function shapes quite easily, but it is easier to explain the concepts with all of attribute gradations shaped the same in the demonstration. In this task, there was translation from the MATLAB Fuzzy Toolbox ™ to the ABM through a Java code translation.

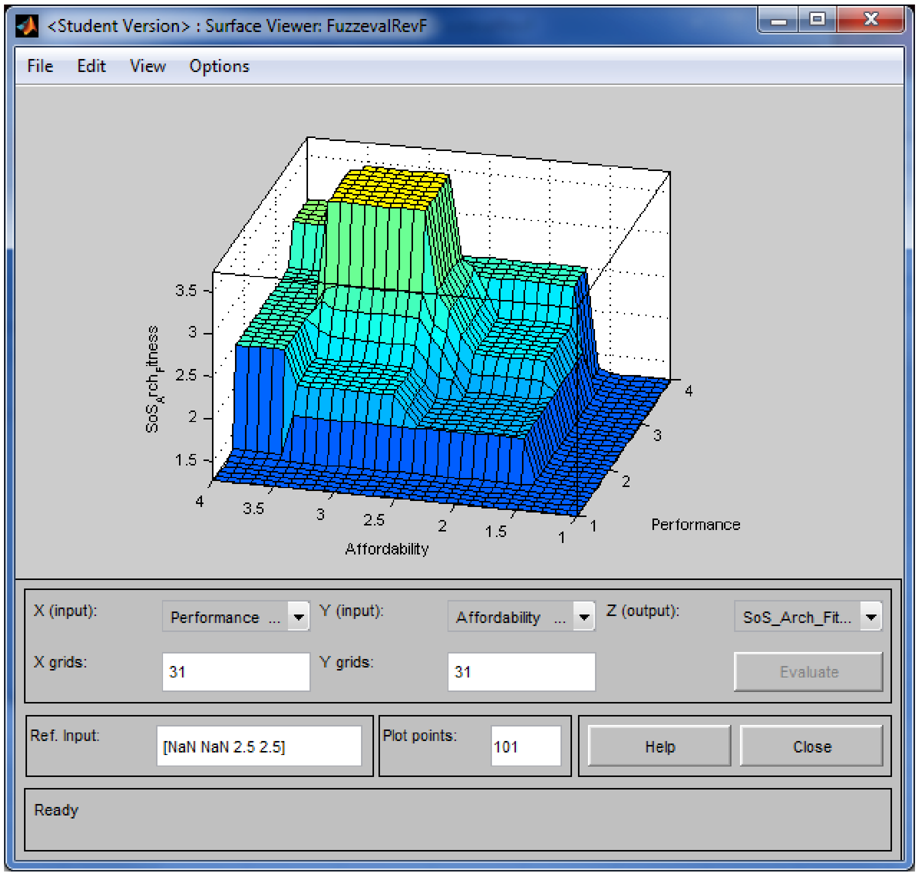
So a set of algorithms, operating in an as yet undefined way on the areas of interest embodied by the attributes, provides an evaluation for each of the attributes of a chromosome representing an architecture. One could retain the mathematical values of the individual attribute evaluations and proceed to arrive at the overall SoS evaluation, but it would require a large number of complicated logical comparisons and manipulations to reach any conclusion. The fuzzy associative memory (FAM) allows one to describe this evaluation in a much simpler, and more general way. The individual attribute values are summarized as Unacceptable, Marginal, Acceptable, and Exceeds (requirements). This is similar to the color coding in Contract Performance Assessment Reports(Department of the Navy, 1997), with red, yellow, green, gold and blue representations across a number of performance areas (see Figure 23, Appendix B). One could use Fuzzy evaluation criteria to reach the conclusions in each of the attributes in a real system, but again, that is for future implementation.

4.2.2.6 COMBINE THE ATTRIBUTE EVALUATIONS TO ARRIVE AT THE SOS EVALUATION

One does not want to write a series of *if – then* statements, because they would need to cover all 256 possible combinations. Changes to this system to handle another case, with a different number of gradations, different weights for different attributes, or a different number of attributes would be horrendously convoluted. A Fuzzy associative memory can make the evaluation process far more general and reusable by being simple to change. By encoding just 6 simple, plain language rules, shown in Table 9 (this becomes 9 rules in the formal language because MATLAB won't mix *and* with *or* inside a single rule), to be used with the Fuzzy Membership Functions. This list of rules, coupled with the membership functions identified above, results in the fairly complicated, non-linear surface shown in Figure 6.

Table 9. Fuzzy Rule Set defining SoS evaluation from our example attribute values

Plain Language Rule	Fuzzy Rule Definitions from MATLAB Fuzzy Toolbox
If ANY attribute is Unaccptable, then SoS is Unacceptable	If (Performance is Unacceptable) or (Affordability is Unacceptable) or (Developmental_Flexibility is Unacceptable) or (Robustness is Unacceptable) then (SoS_Arch_Fitness is Unacceptable) (1)
If ALL the attributes are Exceeds, then the SoS is Exceeds	If (Performance is Exceeds) and (Affordability is Exceeds) and (Developmental_Flexibility is Exceeds) and (Robustness is Exceeds) then (SoS_Arch_Fitness is Exceeds) (1)
If ALL the attributes are Marginal, then the SoS is Unacceptable	If (Performance is Marginal) and (Affordability is Marginal) and (Developmental_Flexibility is Marginal) and (Robustness is Marginal) then (SoS_Arch_Fitness is Unacceptable) (1)
If ALL the attributes are Acceptable, then the SoS is Exceeds	If (Performance is Acceptable) and (Affordability is Acceptable) and (Developmental_Flexibility is Acceptable) and (Robustness is Acceptable) then (SoS_Arch_Fitness is Exceeds) (1)
If (Performance AND Affordability) are Exceeds, but (Dev. Flexibility and Robustness) are Marginal, then the SoS is Acceptable	If (Performance is Exceeds) and (Affordability is Exceeds) and (Developmental_Flexibility is Marginal) and (Robustness is Marginal) then (SoS_Arch_Fitness is Acceptable) (1)
If ALL attributes EXCEPT ONE are Marginal, then the SoS is still Marginal	If (Performance is Marginal) and (Affordability is Marginal) and (Developmental_Flexibility is Marginal) and (Robustness is Acceptable) then (SoS_Arch_Fitness is Marginal) (1)
	If (Performance is Marginal) and (Affordability is Marginal) and (Developmental_Flexibility is Acceptable) and (Robustness is Marginal) then (SoS_Arch_Fitness is Marginal) (1)
	If (Performance is Marginal) and (Affordability is Acceptable) and (Developmental_Flexibility is Marginal) and (Robustness is Marginal) then (SoS_Arch_Fitness is Marginal) (1)
	If (Performance is Acceptable) and (Affordability is Marginal) and (Developmental_Flexibility is Marginal) and (Robustness is Marginal) then (SoS_Arch_Fitness is Marginal) (1)



X (input): Performance ...
Y (input): Affordability ...
Z (output): SoS_Arch_Fit...
X grids: 31
Y grids: 31
Ref. Input: [NaN NaN 2.5 2.5]
Plot points: 101
Ready

Figure 6. Three dimensional representation of the surface of SoS evaluation for all values of Performance and Affordability

The 256 row matrix of all possible input Attribute values produces values from the non-linear combination rules, where 1 = Unacceptable, 2 = Marginal, 3 = Acceptable, and 4 = Exceeds, within each attribute of **P**erformance, **A**ffordability, **F**lexibility, and **R**obustness is shown in Appendix B. Thus far, there is a framework for the FAM to work inside the ABM.

4.2.3 META-ARCHITECTURE GENERATION AND SELECTION

So, to facilitate the prototype progress for phase 1 of the project, a set of initial architectures is generated at random for the Genetic Algorithm (GA). Out of a demonstration universe of ten systems, a portion is selected for participating in the SoS architecture with a probability of about 0.7. In addition, out of 45 possible interfaces between the ten systems, the probability of being selected is 0.8. The interface selection process ensures that the selected interfaces are feasible. This is in the sense that interfaces of non-contributing systems are rejected while, the interface random generation process is repeated until a feasible solution is reached.

The result is a population of ten chromosomes inhabiting the meta-architecture space. Each chromosome represents an alternative SoS architecture. This is done by assuming a vector with binary numbers, where a one stands for a contributing system or interface and a zero for the opposite. The collection of all contributing and non-contributing systems and possible interfaces form a chromosome. Figure 7 illustrates a chromosome representation. The chromosome structure incorporates a number of genes, where a gene S_i refers to a system and S_{ij} to an interface between any two systems *i* and *j*. S_i and S_{ij} can take the value of either one or zero as a genetic alphabet.

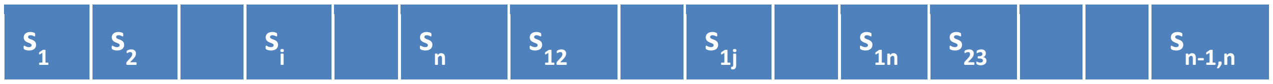


Figure 7. Chromosome representation

In this research, the meta-architecture offers a space that is spanned by all possible SoS architecture configurations with specific information about the set of architectures under consideration. This information is released as outputs for assessment and tradeoff analysis.

Architecture assessment takes place in the fuzzy assessor as explained above, whereas tradeoffs and negotiations among System Agents and with the SoS Agent take place in the agent-based model. Each proposed configuration of a SoS architecture constituted of ten systems and underlying interfaces is evaluated and negotiated as shown in Figure 8. The process is repeated for the set of ten proposed architectures.

The results of the above processes give a score for each of the systems and interfaces under consideration. These scores are provided in the form of crisp numerical values. Together they bring about a vector with ten scores covering the set of ten SoS architectures.

From the meta-architecture perspective, the score vector feeds back an input for rating of the architectures. This also represents chromosome fitness values for the analyzed population. The fitness values would then be used as inputs in the genetic algorithm to determine the next generation of chromosomes. However, only one generation has been implemented during the first phase of the project. Genetic algorithms have been used for the generation and selection of system architectures in conjunction with fuzzy logic as fitness assessor previously (Haris & Dagli, 2011).

In addition, in the process of generating the meta-architectures for the SoS, the future research would employ the evolutionary programming approach to account for individual system architectures while the selection process would consider the key system attributes for the SoS. Therefore, the proposed approach provides a structure to evaluate each meta-architecture based on individual architecture modifications according to interfaces to be imposed upon the meta-architectures generated for the System of Systems.

4.3 Individual System Analytic Tools and Models

4.3.1 System SoS Cooperation Level Assessment Agent Based Model (ABM) framework

The ABM is coded in AnyLogic ™, a commercially available agent modeling system that includes interfaces with external modules in several programming languages and a graphic user interface to illustrate the workings of the ABM.

4.3.2 Negotiation Model

The ABM framework is built with 'hooks' for more complicated negotiation models to be inserted. In the demonstration program, simple selectable probability stubs use imports from Excel ™ spreadsheets to provide a source of variable inputs to check out the integration of the model.

4.3.3 Genetic Algorithm (GA)

The genetic algorithm is a primitive binary chromosome model to show how it can be integrated into the ABM framework. Additional work needs to be done to integrate with feasible modifications, negotiation and the effect of bureaucratic inertial models, the FAM assessor, and show directed evolution over several epochs.

4.3.4 Fuzzy Associative Memory (FAM)

The FAM shows how highly non-linear fitness functions can be included in the ABM framework through simple rules and membership function definitions to produce sophisticated behavior in the SoS objective function. Showing how one can demonstrate highly non-linear outputs even before the introduction of the more complicated negotiations models into the ABM was one of the goals of the research.

5 Implementation/Application of SoS Agent-Based Architecture Framework in AnyLogic

5.1 SoS Domain used in Agent-Based Model Variable Identification

In order to validate the theoretical model described in the previous sections, it is helpful to choose an example domain for the SoS. The selected domain for this research is the DoD Intelligence, Surveillance, and Reconnaissance (ISR) domain. The Agent-Based Model (ABM) implements the theoretical modeling framework described above using an SoS Agent and a System Agent. In terms of ABM, an agent is an abstract entity having unique characteristics or attributes and behavior. An agent is instantiated into a tangible object similar to the way an abstract class in C^{++} is instantiated into a specific class. Just as an abstract C^{++} class can be instantiated several times with different initial settings, an agent can be replicated with or without different initial settings. An agent is implemented in software as an abstract class in order to maintain the independence among agents. This independence of agents and consequently the independent processing, results in a model that more accurately represents the real world situation than is possible in a discrete event simulation or a system dynamic simulation.

That said, the ABM implementation represents the independence among the individual systems that comprise the SoS and upon which the acknowledged SoS depends. This research implements an ABM with one SoS Agent that reflects the characteristics and behavior of an acknowledged SoS. This ABM currently has ten individual System Agents, but future work can have a different number of System Agents. Figure 9 shows the software architecture of the ABM. The ABM has one instantiation of the SoS Agent and ten instantiations of the individual System Agent. The SoS Architecture is a data item implemented as the chromosome depicted in Figure 7.

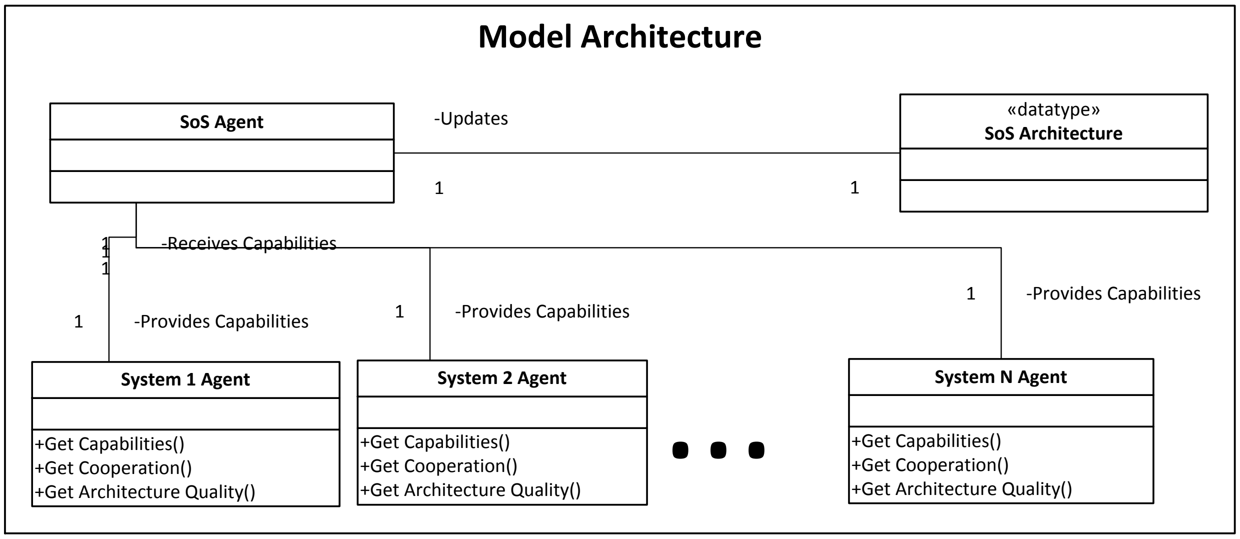


Figure 8. RT-37 Agent-Based Model Architecture

5.1.1 SoS Agent Structure

The SoS Agent starts in the Initialize SoS state and then transitions to the Develop/Evolve SoS Architecture state. From the Develop/Evolve SoS state, the SoS Agent moves to the Plan SoS Update state and then to the Implement SoS Architecture state. The SoS Agent processing follows the states identified in the Wave Model in Figure 1. The SoS Agent state diagram is illustrated in Figure 9.

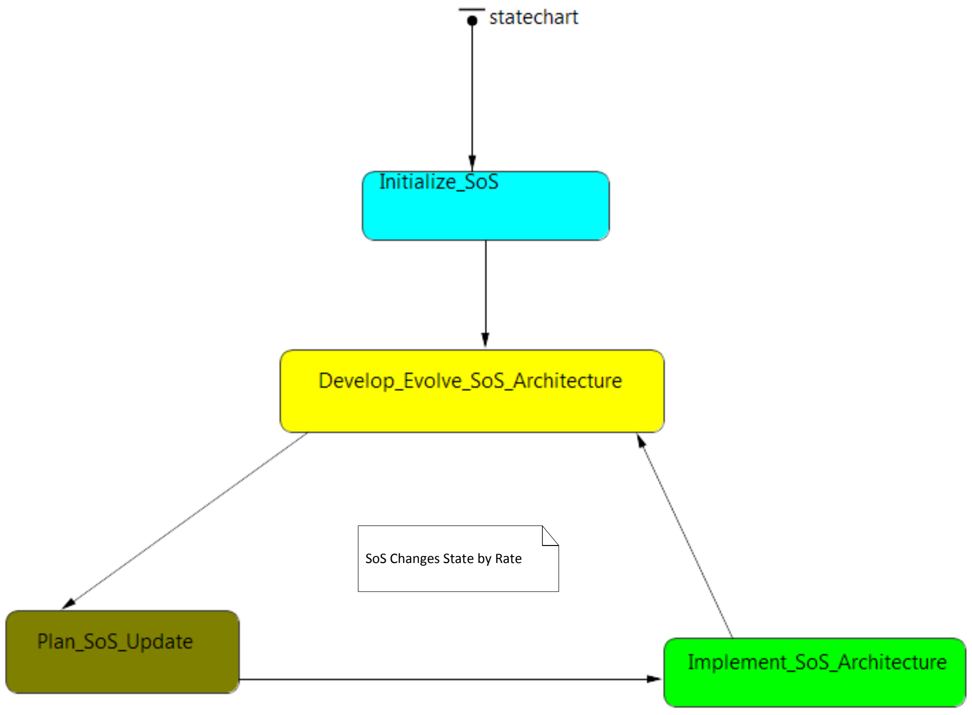

statechart
Initialize_SoS
Develop_Evolve_SoS_Architecture
SoS Changes State by Rate
Plan_SoS_Update
Implement_SoS_Architecture

Figure 9. SoS Agent State Diagram

The SoS Agent reads in the initial chromosome from an Excel spreadsheet, generated from the MATLAB Genetic Algorithm component. Based on which systems will cooperate, the SoS Agent will create an Updated Chromosome that reflects the actual SoS architecture. The Updated Chromosome is written into a second Excel spreadsheet that implements the formulas that evaluate the Updated Chromosome for the architecture attributes of Affordability, Flexibility, Robustness, and Performance as described in Table 11. The ABM reads the attribute values from the second spreadsheet and those values are input to the Fuzzy Assessor. The Fuzzy Assessor has a Fuzzy Inference Engine that is implemented as a Fuzzy Associative Memory (FAM). The output of the Fuzzy Assessor is the overall SoS Architecture Quality, or fitness, or objective function. The structure of the SoS Agent is depicted in Figure 10.

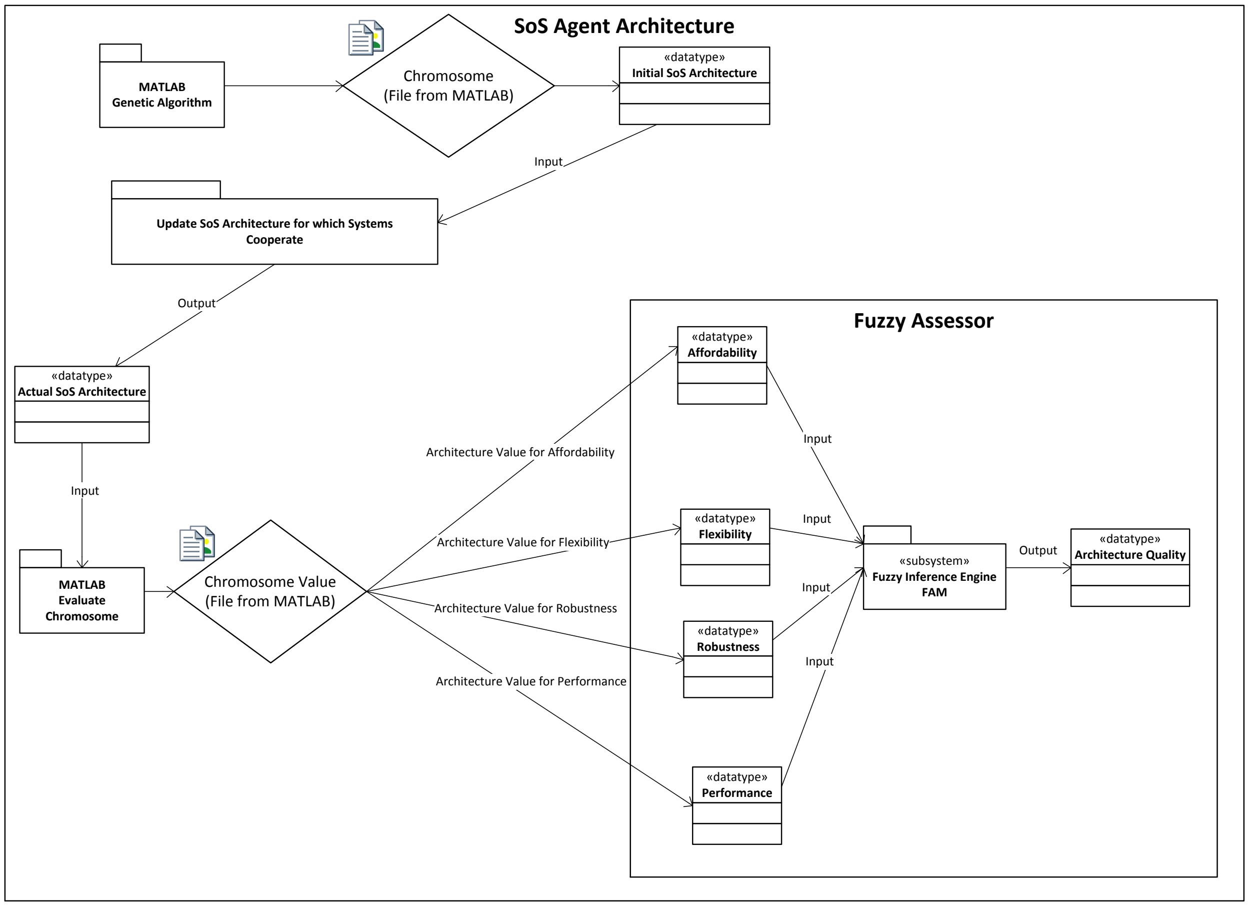

SoS Agent Architecture
MATLAB Genetic Algorithm
Chromosome (File from MATLAB)
«datatype» Initial SoS Architecture
Input
Update SoS Architecture for which Systems Cooperate
Output
«datatype» Actual SoS Architecture
Input
MATLAB Evaluate Chromosome
Chromosome Value (File from MATLAB)
Architecture Value for Affordability
Architecture Value for Flexibility
Architecture Value for Robustness
Architecture Value for Performance
Fuzzy Assessor
«datatype» Affordability
«datatype» Flexibility
«datatype» Robustness
«datatype» Performance
Input
Input
Input
Input
«subsystem» Fuzzy Inference Engine FAM
Output
«datatype» Architecture Quality

Figure 10. SoS Agent Architecture for RT-37 Model

5.1.2 System Agent Structure

The System Agent has a Prep, Initialize System, Cooperate, Non-Cooperate, and Maybe states. The System Agent transitions between states according to the System Agent state diagram in Figure 11 During the Prep state, the System Agent idles, providing time for the SoS Agent to initialize. The System Agent performs its own initialization during the Initialize state. In the Maybe state, the System Agent makes the decision to cooperate or not cooperate based on the probability specified by the user.

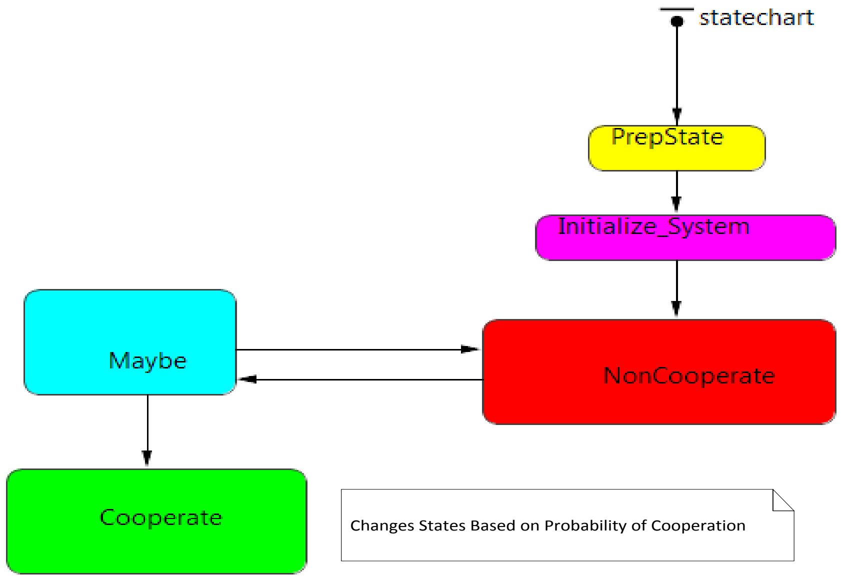


Figure 11. System Agent states for RT-37 model

5.1.3 SEQUENCE DIAGRAM

The sequence diagram in Figure 12shows the sequence of events between the SoS Agent and the System Agent. The sequence diagram represents one wave of the Wave Model for one initial SoS architecture. The ABM starts with an initial SoS architecture (SoS.A_0) as described in Table 3. The initial SoS architecture is the chromosome generated in MATLAB as described in section 3.2.2. The ABM sends the SoS Connectivity Request (SoS.R_i) to each of the individual System Agents. For the initial phase of this research, the SoS Connectivity Request does not include the SoS architecture, funding, or deadlines. These data items are necessary for the System Negotiation that will be included in the next phase of research, so they will be implemented at that time. Each individual system makes a decision whether to cooperate with the SoS request and sends the System Reply (System.Information_i) back to the SoS. For the initial phase of the research, the System Reply only contains the System Cooperation (System.coop_i) and the system cooperation decision is based on a probability that can be set by the user. At this time the System Cooperation is a boolean value (True/False). But the next research phase will incorporate a degree of cooperation for each individual system based on System Willingness and System Ability, which more accurately reflects the real world situation. The next research phase will also incorporate the System Time to Cooperate (System.cooptime_i) and the System Availability Time (System.av_i), also not included in the initial phase.

Since this is an agent-based model and the individual systems execute independently, individual systems process data and change states asynchronously. Thus, the user setting for probability is not received and processed by each individual system at the same time. Therefore, changing the probability of cooperation may cause different probabilities for different systems in the same wave epoch.

The framework for the System Capabilities (System.c_i), System Performance (System.p_i), and System Ability (System.ability_i) is built, but implementation is planned to be in the next research phase.

Model Sequence Diagram (One Chromosome) (One Iteration/Chromosome)

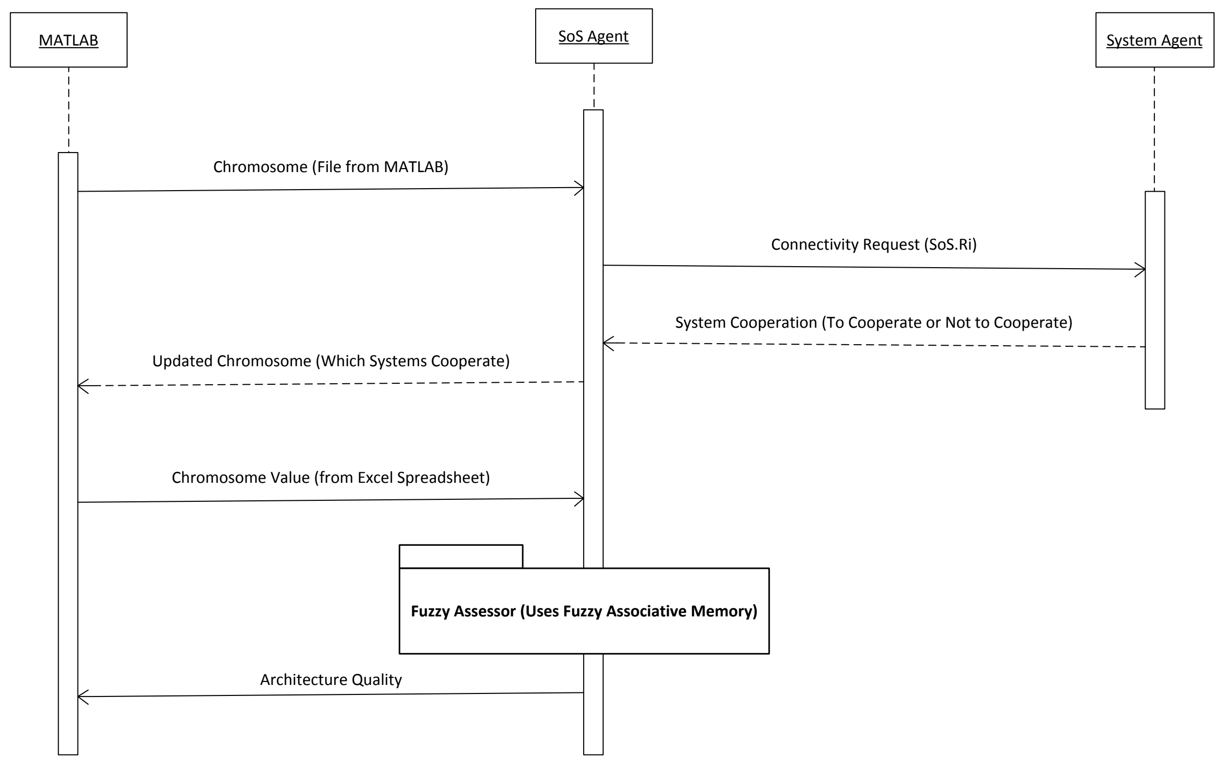

MATLAB
SoS Agent
System Agent
Chromosome (File from MATLAB)
Connectivity Request (SoS.Ri)
System Cooperation (To Cooperate or Not to Cooperate)
Updated Chromosome (Which Systems Cooperate)
Chromosome Value (from Excel Spreadsheet)
Fuzzy Assessor (Uses Fuzzy Associative Memory)
Architecture Quality

Figure 12. Chromosome flow within the model

The present implementation has one iteration of the sequence diagram of Figure 12 for one chromosome. For the initial research phase, ten chromosomes were generated from the MATLAB Genetic Algorithm and the ABM performs one cycle of the sequence diagram for each chromosome. The next research phase ABM will have multiple iterations of the sequence diagram for multiple chromosomes.

5.2 PROCESSES TO BE FOLLOWED IN MODEL USAGE

For a proposed SoS with **n** Systems, the SoS Agent invites the System Agents into the SoS with some offer of resources, usually funding, to provide an extension of their current capability. This extension, whether of basic capability or a changed interface with other System(s) within the proposed SoS, enables the desired, new SoS capability. The SoS Agent proposes an organization and resources to create the new SoS capability.

The Systems contract to accept the resources and deliver their capability increment as a participating System within the SoS, or they choose not to participate. Future improvements to the model may go through more complicated negotiation regimes for joining and allocating resources, but at this stage of development, the model uses a simple random number generator comparison to a selectable threshold and a slightly over budget initial offer of resources to determine which of the Systems join the SoS, and how much that architecture will cost. The agent based model portion may be grown to be able to handle negotiations among Systems and SoS Agent over participation and resources, but currently is only the framework for adjusting the model time and display of participating systems, and producing the fuzzy evaluation at each epoch.

This simplification glosses over the genetic algorithm portion of the model. The intent is to be able to search the potential architecture space through the genetic algorithm. The genetic algorithm will eventually be used to propose new configurations of systems both in the initial and subsequent epochs.

The fuzzy associative memory will grow to discover and document appropriate evaluation attributes, criteria, membership functions, and rules for a domain specific application of the model. At this stage, it is largely pre-selected, simple examples of fuzzy membership functions, rules and simple algorithmic methods of evaluation of an artificially simple four attribute, four gradation metric to evaluate the success of the SoS at each epoch. It is complicated enough to be interesting, but not overwhelming, during the feasibility demonstration phase.

5.3 Early Experiences with the Prototype Model

5.3.1 ISR Domain Choice

ISR was selected to model as a domain with numerous existing SoS. An example problem where a new capability was needed was during the Gulf War, when the previous mere threat of Scud missiles became reality and they were actually launched at our troops and allies from mobile launchers. Existing ISR processes were not finding the launchers in time to take an action against them. Satellites, high altitude-large area aircraft systems, and tactical aircraft were all using observation methods that might have found the launchers, but they were not getting the information to anyone in a timely way to be able to take action. The ISR system of systems also included various methods of data delivery from film to voice communications, then processing, fusion, analysis and interpretation, generation of a target report, delivery of this information back to tactical forces, and finally, physically arriving at the location to deliver an effect. The existing, cumbersome SoS process was not delivering the required capability to stop the Scud launches. Since there were many systems that could be combined to improve the capability to find the mobile launchers, this is a reasonable approximation of what we have modeled. Some ISR systems were invited to help participate in an SoS architecture to achieve a new capability.

The demonstration model is using the number of systems invited in as $\mathbf{n} = 10$, the probability that they choose to participate as $\mathbf{p} \sim 0.7$, and the probability that participating systems interface with any other system as $\mathbf{q} \sim 0.7$.

To provide evaluations directly from the chromosome for the demonstration integration of the ABM with the GA and the FAM, the attributes were evaluated from the chromosome as follows.

5.3.1.1 Performance

To evaluate in the **Performance** attribute, start with an interim definition of $\mathbf{C_{req}}$ as a first iteration estimate of the simple Requested Capability from the SoS. Use the approach of simply adding up the capability contributions of the individual Systems, so that the contribution from each individual system is $\mathbf{C_i = C_{req} /n}$. But, in a reference to Network Centric Systems(David S. Alberts, 1999), one may also assume that there is a gain of *some* Capability through the interfaces between the Systems as well. Using $\mathbf{C_{int\ ij}}$ as the small but distinct Capability contribution from the **ij**th interface; assume the *expected* total contribution from the interfaces is about equal to the *requested* Capability from the Systems themselves: $\mathbf{C_{int\ ij} = C_{req}/(pqn(n-1)/2)}$. Therefore, the expected Capability of the SoS from both Systems and Interface contributions is $\mathbf{C_{SoS} = pnC_i + C_{req} = (1+p)C_{req}}$. The fact that a random selection of Systems and Interfaces with the selection parameters **p** and **q** can provide more or less than the expected numbers of systems and interfaces allows some architecture chromosomes to deliver more or less than either expected or requested capability. Simply normalize the performance by the initial requested capability $\mathbf{C_{req}}$. If one gets exactly the expected contributions, it will be $\mathbf{(1 + p)C_{req}}$.

5.3.1.2 Affordability

B is Total Budget of the SoS Agent; assume for the moment that interfaces cost nothing, but the decision to participate costs each System **(1+p'/2)B/n** . This requires explanation. Without *some* extra percentage offered over the budget, there would *never* be an SoS architecture chromosome that would be over budget in our simple binary participation model, and therefore break the **Affordability** attribute. One expects *some* invited Systems to be unable to comply for various reasons ((**1-p**) will not participate; call that **p'**); those Systems would reject the budget offered, and there would be no spending of those Systems' budget allocation. However, if you use only **p'B** to adjust the budget, then anytime the random number generator produces even one more than the *expected* number of participating Systems, it will already be over budget, so distribute only **Bp'/2** extra total budget to bring a few more chromosomes within the budget, and therefore worth evaluating further. So if the expected number of systems participate, the SoS will be Affordable to a reasonable (Affordability attribute = Acceptable), but not Exceeds, degree.

5.3.1.3 Flexibility

Developmental **Flexibility** can be represented simply by the number of Systems participating. If more Systems participate, there is more room to maneuver among them. So Flexibility can be represented by the simple sum of the System_i components of the chromosome. One could also argue that more systems contribute to Robustness as well, because flexibility and robustness are strongly correlated. But don't add that complication for the feasibility demonstration. To keep the values normalized, divide by the possible maximum, **n** .

5.3.1.4 Robustness

Use the number of interfaces as the measure of Operational **Robustness**, with the maximum number of interfaces as the normalization. This will allow chromosomes that evaluate better and worse than the *expected* number of interfaces. Mathematically, the sum of the interface chromosome positions (with a one contribute), divided by the normalization factor of the maximum possible **n (n-1)/2** .

Evaluation algorithms for each SoS attribute can then be summarized in terms of the Chromosome itself as shown in Table 10, with the example chromosome in Figure 13.

Table 10. Evaluation of Attributes from chromosome

SoS Attribute	Equation	Unaccep table	Marginal	Accept able	Exceeds
Performance	**$(\text{Sum} (C_i) + \text{Sum} (C_{int\ ij})) / C_{req}$**	< .8	.8 - 1	1- 1.5	< 1.5
Affordability	**$(\text{Sum} (\text{System}_i * (1+(1-p)/2)B/n)) / B$**	> 1	.9 - 1	.8 - .9	< .8
Developmental Flexibility	**$\text{Sum} (\text{System}_i) / n$**	< .5	.5 - .8	.8 - .9	.9 - 1
Robustness	**$\text{Sum} (\text{Interface}_{ij})/ (.5n(n-1)))$**	< .5	.5 - .7	.7 - .9	.9 - 1

Future work may bring the fuzzy evaluation process down a step to this level, for the attribute evaluations, but in the demonstration of tool integration phase, we held the fuzzy process to combining these four attributes evaluated to these four gradations to an overall SoS evaluation.

Table 11. Example four attribute evaluation guidelines

\Evaluation Attribute \	**Unacceptable**	**Marginal**	**Acceptable**	**Exceeds performance**
Performance (KPPs for ISR SoS) Coverage (sq km/hr) Resolution # of channels Timeliness (sensor on target, and processing, dissemination) Adaptability (sensor type match to target)	Fails to meet multiple key performance parameters (KPPs)	Fails to meet at least one key performance parameter (KPPs)	Meets or exceeds all KPPs.	Exceeds performance in one or more KPPs by 20% or more.
Affordability A measure of the projected total ownership cost versus budget (acquisition cost plus O&M cost) and delivered capability.	Projected total ownership cost exceeds 120% of budget. Large mismatch in annual estimates.	Projected total ownership cost exceeds 100% of budget.	Projected total ownership cost is between 85% and 99% of budget.	Projected total ownership cost is less than 85% of budget.
Robustness (in the field) Ability of the SoS to continue proper functioning despite external disturbances.	More than 30% degradation in one or more KPPs due to external disturbances.	Between 10% and 30% degradation on one or more KPPs due to projected external disturbances.	Between 5% and 10% degradation in one or more KPPs due to projected external disturbances.	Not more than 5% degradation in any KPP due to estimated external disturbances.
(Developmental) **Flexibility** Ease with which the SoS can be repurposed to support other missions. Ease with which individual system contributions can be traded	Architecture is monolithic and key SoS capability applications are tightly coupled. 0-25% of key functionality is allocated to software.	Several different Architectures are possible with varying degrees of cooperation among systems. 25-50% of key functionality is allocated to software.	Architecture is layered and most key SoS capability applications are loosely coupled. 50-75% of key functionality is allocated to software.	Architecture is fluid and all key SoS capability applications are loosely coupled. > 75% of key functionality is allocated to software.

5.3.2 Sample Fuzzy Evaluations

To evaluate the fuzzy evaluation part, sample Excel spreadsheets with random chromosome definitions were used to test the above evaluation approaches.

s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	1-2	1-3	1-4	1-5	1-6	1-7	1-8	1-9	1-10	2-3	2-4	2-5	2-6	2-7	2-8	2-9	2-10	3-4	3-5
1	1	1	0	1	1	1	1	1	1	1	1	0	0	1	0	1	1	0	1	0	1	0	1	1	1	1	0	1

…

Figure 13. Sample (partial) binary chromosome for 10 Systems

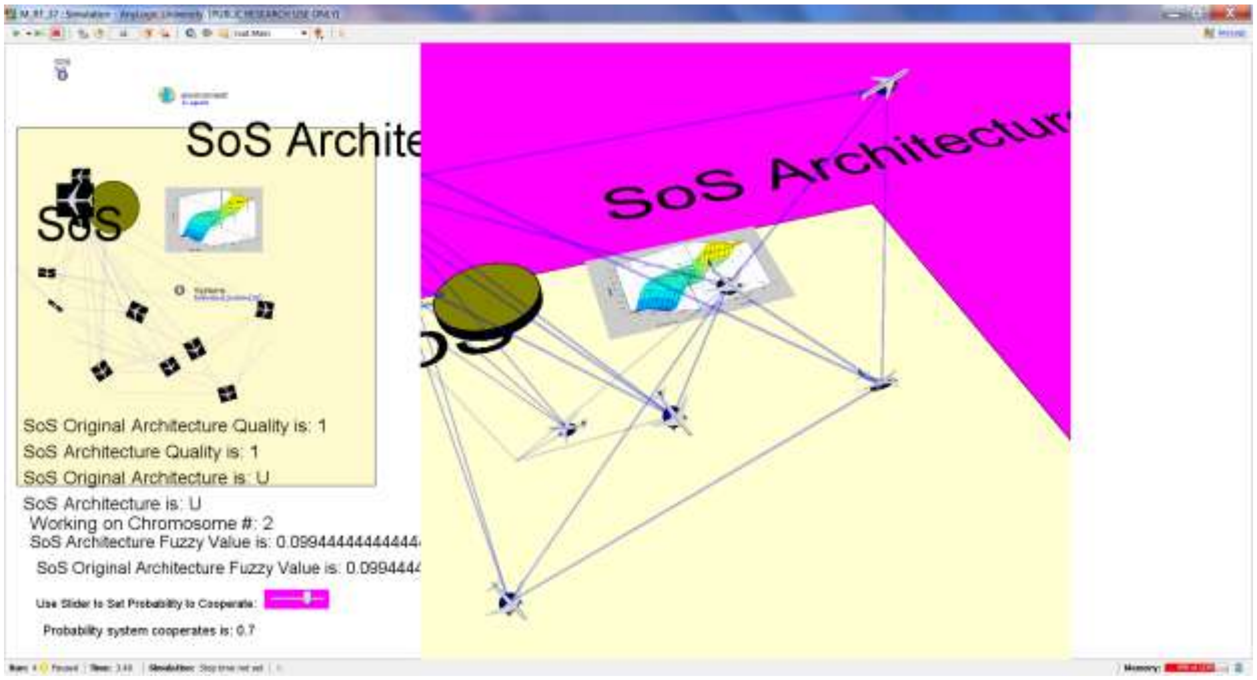
Table 12. Spreadsheet evaluation of sample chromosome

		evaluation
SoS Performance	2.23	E
SoS Affordability	1.03	U
SoS Flexibility	0.9	E
SoS Robustness	0.77	A

By creating sample, random chromosomes in an Excel spreadsheet, one can easily check the evaluation criteria for a variety of test architecture chromosomes. A quick check evaluation of the chromosome representation of the SoS in the four chosen attribute areas is possible. The combination of the four attributes through the rules is done in the Fuzzy associative memory in MATLAB, then exported to the ABM to be implemented in Java. The GA creates sample chromosome sets in Excel, which the ABM reads, and returns the fuzzy evaluation for each chromosome so the GA can select new chromosomes for the next trial.

5.3.3 Agent-Based Prototype Model Results and Future Research

Using the initial ten chromosomes, the ABM provides the SoS architecture quality based on which systems cooperate. The ABM provides the actual SoS architecture quality and the initial SoS architecture quality. A snapshot of the simulation window is provided in Figure 14.



```
SoS Original Architecture Quality is: 1
SoS Architecture Quality is: 1
SoS Original Architecture is: U
SoS Architecture is: U
 Working on Chromosome #: 2
 SoS Architecture Fuzzy Value is: 0.09944444444444
  SoS Original Architecture Fuzzy Value is: 0.0994444
  Use Slider to Set Probability to Cooperate:
   Probability system cooperates is: 0.7
```

Figure 14. Simulation window in Anylogic

The current prototype simulation window displays the architecture quality (value ranging from 1-4) for the ten original architectures generated by the GA. It also displays the architecture quality at the end of the wave based on the cooperation response of individual systems. The fuzzy quality values (U, M, A, E) for original and current SoS architecture are also displayed. The probability of individual systems cooperating (0-1) can be varied during simulation experiments.

The preliminary results indicate that multiple iterations of the wave cycle result in an actual SoS architecture that matches the initial SoS architecture. This is due to the fact with each additional iteration of the wave, more and more systems cooperate. With the probability of cooperation left at 70% and using the ten chromosomes from the MATLAB Genetic Algorithm, the actual SoS architecture matches the initial SoS architecture after the first iteration or first wave cycle. The difference between the initial SoS architecture and the actual SoS architecture for the first wave cycle is typically only in one system which results in a change in one bit in the chromosome. This reflects the fact that the formulas used to calculate Affordability, Flexibility, Robustness, and Performance are not significantly affected if only one system does not cooperate. That might not be the case in the real world SoS development, depending on the significance of the capability that could have been provided if that one system had cooperated. If a necessary capability could only be provided by that one system, then the quality of the actual SoS architecture would be significantly affected when that one system does not cooperate. Capability increments were equally distributed, and not prioritized in the initial implementation to show feasibility of the integration approach, but the framework is present.

These results are consistent with the assumption that the weights ($SoS.w_i$) in Table 2 are not implemented in this phase, but will be implemented in the next phase of research. The next phase of research will also include the SoS desired performance parameters ($SoS.P_i$), initial SoS target measures ($SoS.M_0$).

SoS desired capabilities ($SoS.C_i$) are partially implemented in terms of the systems in the initial SoS architectures (chromosomes). Each system in the initial SoS architecture represents a capability that is desired in the SoS. In that way, SoS desired capabilities are partially implemented in the initial research phase and will be fully implemented in the next research phase. External factors (E_0 and E_T) which are a function of National priorities, SoS funding, and threats are also partially implemented in the current ABM. The ABM includes these variables in the Environment but the relationship between these external factors and elements in the SoS development process is to be defined in the next research phase. Gap analysis ($SoS.Gap_T$) in Table 6 and the processing that leads up to Gap analysis in Table 5 are not yet implemented but are reserved for the next phase of research.

5.4 SoS Analytics Workbench Integration

The AnyLogic ABM was chosen as the container or wrapper for the GA and FAM components. It already has a graphic user interface and display capability, and the ability to read in data and embed code from the other components. Finally, the ABM created within AnyLogic can be exported as a Java Applet, so it can be run on other computers without license The agents were thought to have a growth path to be the more complicated parts of the model framework as well, so AnyLogic seemed the natural place for the overall model to reside. There were some difficulties in getting the AnyLogic software to run on the collaborators different computers. That seems to be mostly resolved now, but it was exceedingly time consuming. The FAM is created in MATLAB and exported to a file embedded in the Java code that the AnyLogic ABM uses to do the fuzzy evaluation of the chromosome. Reading the chromosomes from the GA into AnyLogic is now working for one at a time, but the interface needs to be improved significantly. The GA also needs to be upgraded to do the epoch to epoch evolution. We demonstrated export of an ABM as an applet which runs on other computers, although not yet with the Wave Model ABM.

Work being performed at Purdue University is proceeding along complementary lines leading to an SoS Analytics Workbench. It was suggested that a collaboration with them during Phase II of this research task to integrate our joint efforts would be beneficial.

6 Conclusion/ Recommendation for Future Phases

As stated in the executive summary, the goal of this research is to model the evolution of the architecture of an acknowledged SoS that accounts for the ability and willingness of constituent systems to support the SoS capability development. The Wave Process Model for SoS is used as the basis for SoS capability evolution.

An agent-based modeling framework is developed to abstract the non-utopian behavioral aspects of the constituent systems and their interactions with the SoS. Basic prototype structure of the frame work is implemented within AnyLogic software illustrates the impact of individual system behavior on the SoS capability and architecture evolution processes. It also provides a structure for modular development of various subordinate mathematical models that can be developed independently to be integrated to the general model represented with the prototype.

RDTE funding requested in FY13 will be used to complete development, validation, and piloting of MPTs and transition. The basic objective of the second phase is to develop this prototype further to enable its use in pre milestone A of the acquisition phase as a decision making tool without extensive knowledge of Agent based modeling and fuzzy logic and architecture assessment algorithms. The architecture frame work developed in the first phase will be used in this phase to create next model spiral.

This necessitates additional two basic research avenues besides agent based modeling namely:

- Development of a mathematical model that can handle ambiguous information and data within Agent based model to assess the SoS architecture measure of effectiveness and implementation of the genetic algorithm developed into the agent based model structured developed as a result of phase I. (Khaled Haris is a potential PhD student with extensive industrial experience who is interested in working on this research problem).
- Development of a methodology in collecting data, ambiguous information and rules for Fuzzy Assessor which is a part of mathematical model assessing measure of effectiveness in the agent based model. (Lou Pape is a potential PhD student who works for The Boeing Company with extensive industrial experience who is interested in working on this research problem).

In

Figure 15, the model architecture proposed is given

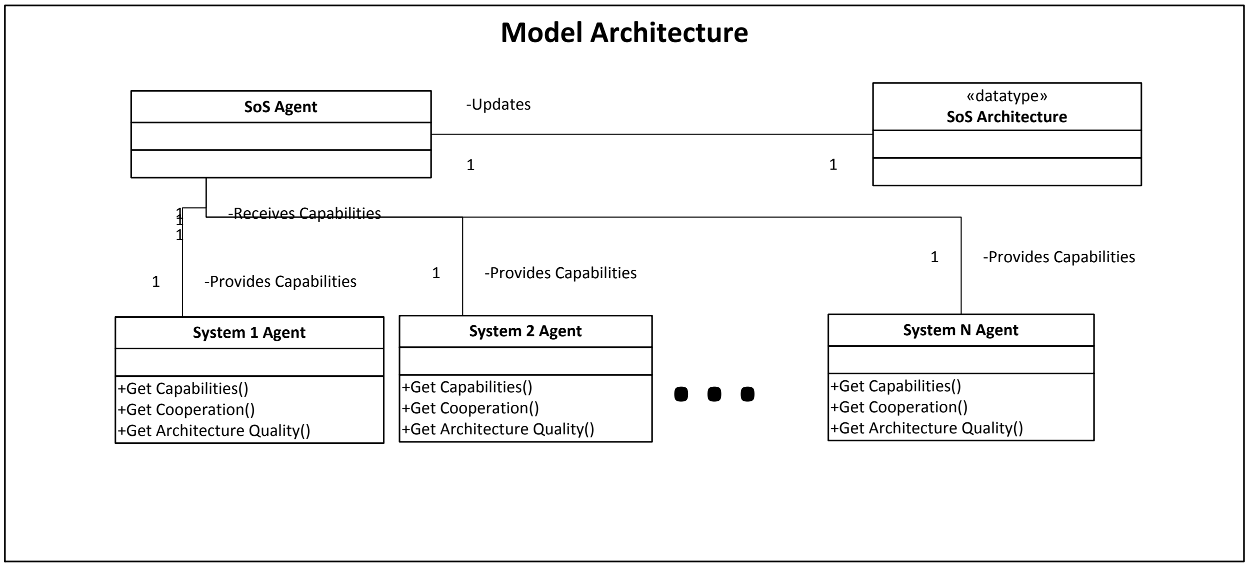


Figure 15. Model Architecture of Phase II

Project description requirement states; "Tooling should not require extensive knowledge of agent-based modeling and underlying algorithms". This necessitates the integration of the genetic algorithm into Anylogic and elimination of text file transfer that is being used in current framework, and improved genetic algorithm to iterate and converge towards optimum meta-architecture. In Figure 16 the Model Sequence diagram is depicted for phase II. The research tasks to be performed are related to genetic algorithms and development of necessary mathematical models in meta architecture generation and evaluation for SoS Agent and systems agent modified architecture generation and assessment will be a part of PhD dissertation of Khaled Haris.

Evaluation of acknowledged SoS meta-architectures alternatives represented as chromosomes and elimination of infeasible architectures is an important part of the model. Fuzzy Associative Memories are demonstrated in phase I for this purpose. There is a need to develop a procedure and a model to be able to capture this information from stake holders of participating systems in acknowledged SoS and the SoS owner. The development of this general model procedure will be a part of PhD dissertation of Lou Pape. Dr. John Columbi's MS students will also contribute to this effort. SoS Agent and System Agent architectures are given in Figure 16, Figure 17, and Figure 18.

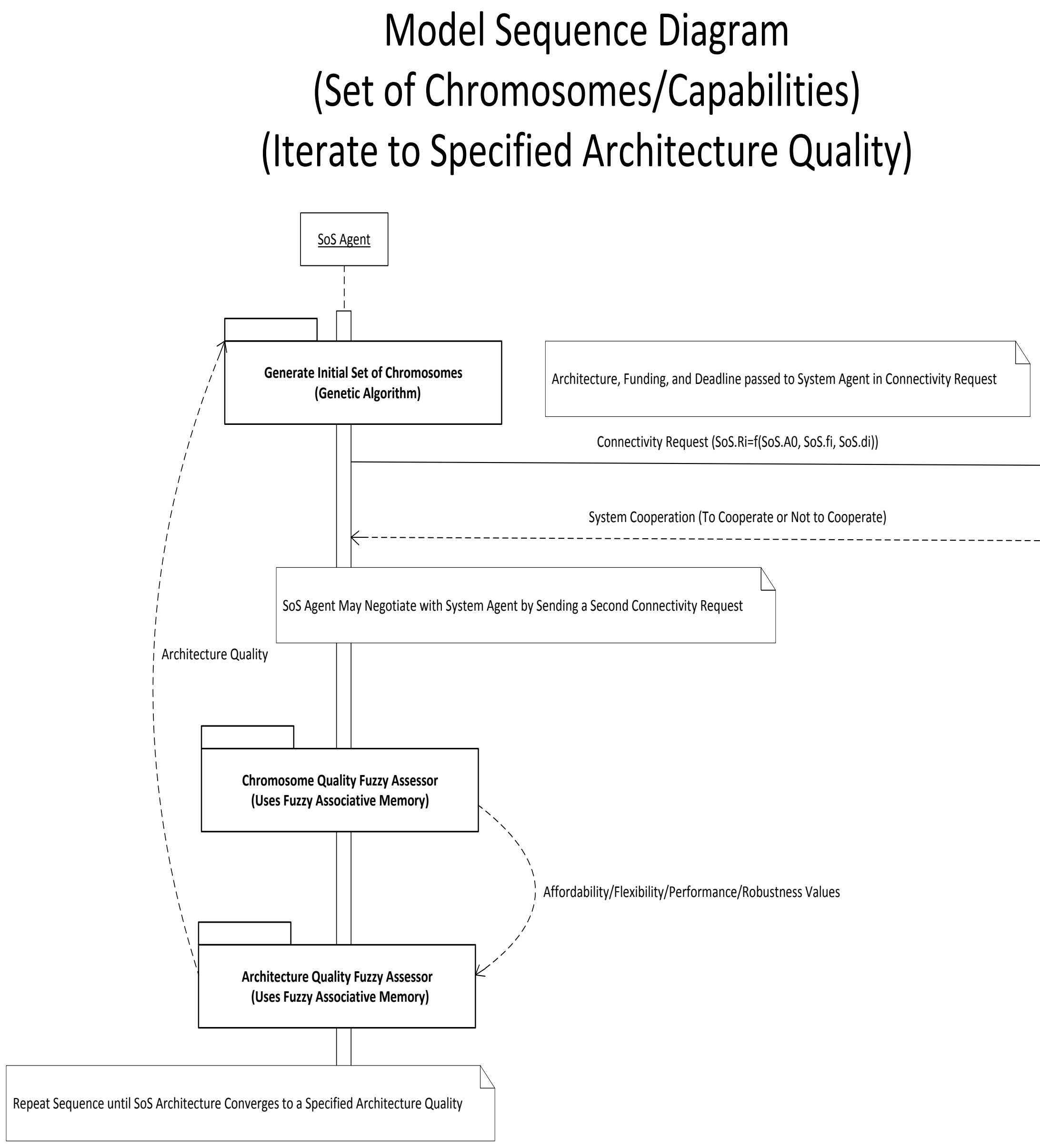

Model Sequence Diagram
(Set of Chromosomes/Capabilities)
(Iterate to Specified Architecture Quality)
SoS Agent
System Agent
Generate Initial Set of Chromosomes (Genetic Algorithm)
Architecture, Funding, and Deadline passed to System Agent in Connectivity Request
Connectivity Request (SoS.Ri=f(SoS.A0, SoS.fi, SoS.di))
System Cooperation (To Cooperate or Not to Cooperate)
SoS Agent May Negotiate with System Agent by Sending a Second Connectivity Request
Architecture Quality
Chromosome Quality Fuzzy Assessor (Uses Fuzzy Associative Memory)
Affordability/Flexibility/Performance/Robustness Values
Architecture Quality Fuzzy Assessor (Uses Fuzzy Associative Memory)
Repeat Sequence until SoS Architecture Converges to a Specified Architecture Quality

Figure 16. Model Sequence Diagram for Phase II

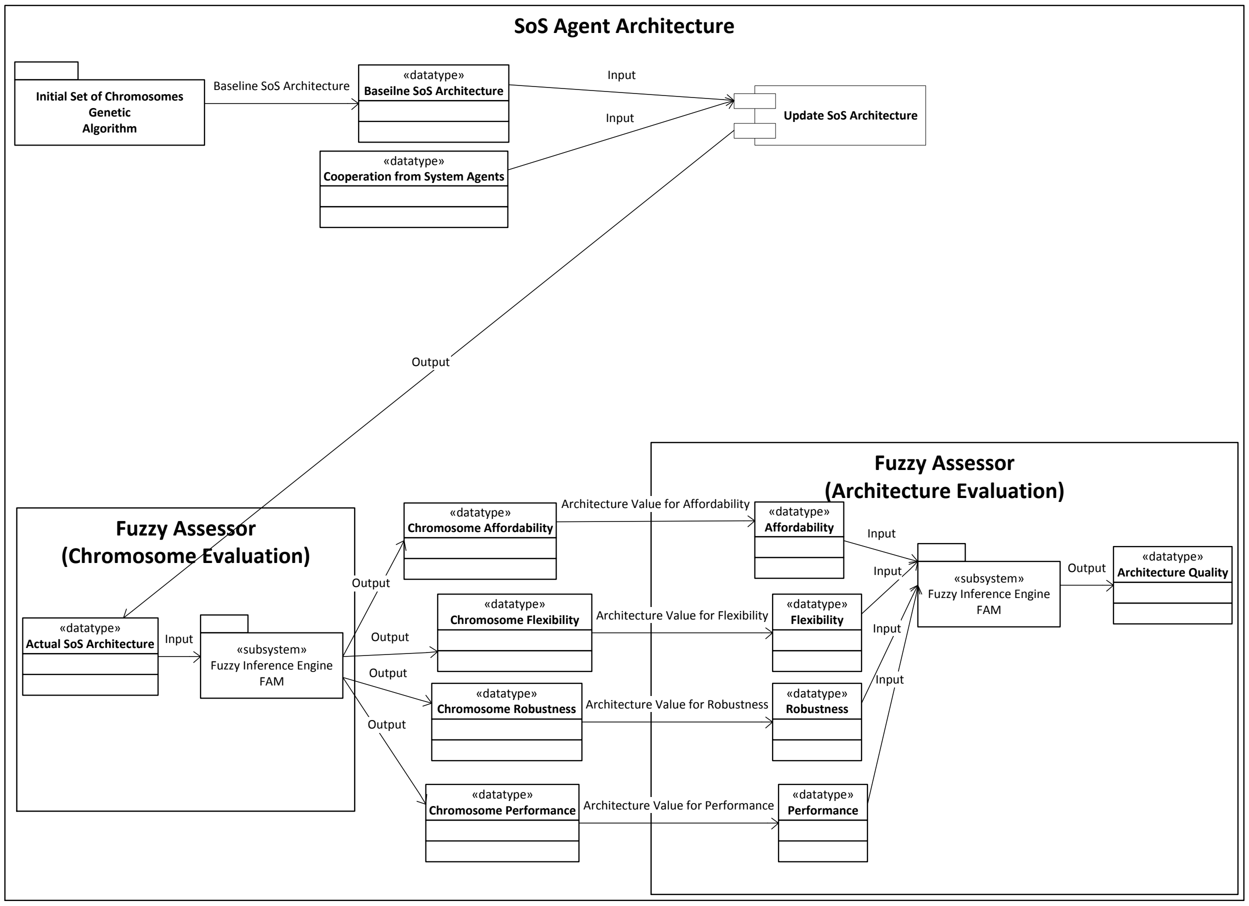

SoS Agent Architecture
Initial Set of Chromosomes Genetic Algorithm
Baseline SoS Architecture
«datatype» Baseilne SoS Architecture
Input
Input
Update SoS Architecture
«datatype» Cooperation from System Agents
Output
Fuzzy Assessor (Chromosome Evaluation)
«datatype» Actual SoS Architecture
Input
«subsystem» Fuzzy Inference Engine FAM
Output
Output
Output
Output
«datatype» Chromosome Affordability
«datatype» Chromosome Flexibility
«datatype» Chromosome Robustness
«datatype» Chromosome Performance
Architecture Value for Affordability
Architecture Value for Flexibility
Architecture Value for Robustness
Architecture Value for Performance
Fuzzy Assessor (Architecture Evaluation)
«datatype» Affordability
«datatype» Flexibility
«datatype» Robustness
«datatype» Performance
Input
Input
Input
Input
«subsystem» Fuzzy Inference Engine FAM
Output
«datatype» Architecture Quality

Figure 17. SoS Agent Architecture for Phase II

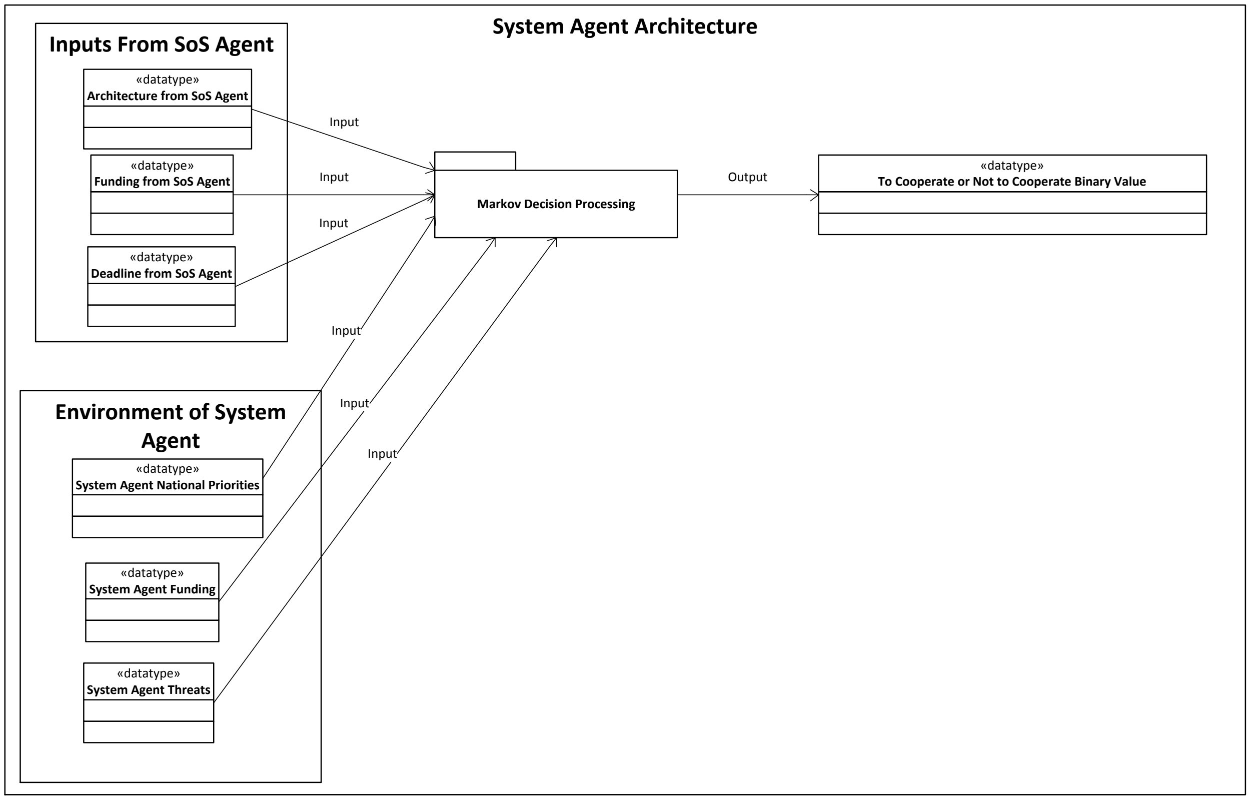

System Agent Architecture
Inputs From SoS Agent
«datatype» Architecture from SoS Agent
«datatype» Funding from SoS Agent
«datatype» Deadline from SoS Agent
Environment of System Agent
«datatype» System Agent National Priorities
«datatype» System Agent Funding
«datatype» System Agent Threats
Input
Markov Decision Processing
Output
«datatype» To Cooperate or Not to Cooperate Binary Value

Figure 18. System Agent Architecture for Phase II

Project description states "Develop, validate, and pilot agent-based modeling MPTs that support predictions about the properties of an Acknowledged SoS when applied early in the life cycle when there is high uncertainty and ambiguity about SoS requirements, architecture, and implementation technologies based on the Wave Process Model". This necessitates development of decision models for individual systems to better capture individual system dynamics and include negotiation mechanisms between individual systems that need to interface with each other and also include negotiation mechanisms between individual system and SoS Agent. The tasks to be performed to achieve this capability in the model will be a part of PhD dissertation of Paulette Acheson. Paulette will also be responsible to generate user friendly Java template where users of the tool can play with various scenario inputs and analyze the overall dynamics.

In order to create non-linear dynamics of SoS systems, there is a need to build several mathematical models to represent systems agents behavior based on each system requested capability by SoS agent and their own performance measures and funds allocated to them by acknowledged SoS. It is essential that these models to be developed independently for each system. It is estimated that three more researchers will join the team from Missouri S&T to develop these models.

During Phase II there will be periodic meetings with Purdue University to make sure that the model can be integrated to SoS Analytic Workbench.

7 BIBLIOGRAPHY

Acheson, P. (2010). Methodology for Object Oriented System Architecture Development. *IEEE Systems Conference.*

ASD(NII), D. (2009). *DoD Architecture Framework Version 2.0 (DoDAF V2.0).* Washington DC: Department of Defense.

Bonabeau, E. (2002). Agent based Modeling: Methods and Techniques for Simulating Human Systems,. *Proceedings of the National Academy of Sciences, Vol. 99* , 7280-7287.

Brazier, F., Dunin-Keplicz, B., Jennings, N. R., & and Treur, J. (1996). DESIRE: modeling multi-agent systems in a compositional framework. *International Journal of Cooperative Information Systems, M Huhns, M Singh (eds), special issue of Formal Methods in Cooperative Information Systems* .

Brazier, F., Jonker, C., & Truer, J. (1997). Formalization of a cooperation model based on joint intentions. In M. W. J.P. Muller, *Intelligent Agents III (Proc. Of the Third International Workshop on Agent Theories, Architectures and Languages, ATAL'96), Lecture Notes in AI, Vol. 1193* (pp. 141-155). Springer Verlag.

Creel, R., & Ellison, B. (2008). *System-of-Systems Influences on Acquisition Strategy Development.* Carnegie Mellon University.

Dagli, C. (. (2011). *Complex Adaptive Systems, Procedia Computer Science Volume 6.* Elsevier.

Dahmann, J., Rebovich, G., Lane, J. A., Lowry, R., & . Baldwin, K. (2011). An Implementers' View of Systems Engineering for Systems of Systems. *Proceedings of IEEE International Systems Conference.* Montreal.

Dauby, j. P., & Dagli, C. H. (2011). The canonical decomposition fuzzy comparative methodology for assessing architectures. *IEEE Systems Journal, vol. 5, no. 2* , 244-255.

Dauby, J. P., & Upholzer, S. (2011). Exploring Behavioral Dynamics in Systems of Systems. In C. (. Dagli, *Complex Adaptive Systems, Procedia Computer Science, vol 6* (pp. 34-39). Elsevier.

David S. Alberts, J. J. (1999). *Network Centric Warfare: Developing and Leveraging Information Superiority, 2nd Edition.* Washington DC: C4ISR Cooperative Research Program.

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

Director Systems and Software Engineering, O. (. (2008). *Systems Engineering Guide for Systems of Systems.* available from http://www.acq.osd.mil/se/docs/SE-Guide-for-SoS.pdf.

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

Dudenhoeffer, D., & Jones, M. (2000). A Formation Behavior for Large Scale Micro-Robot Force Deployment. *Proceedings of the 32nd Conference on Winter Simulation.*

Haris, K., & Dagli, C. (2011). Architecture Trade-off Analysis and Reconfiguration. *Proceedings Conference on Systems Engineering Research.* ISBN -978-0-9814980-1-0.

Jennings, N. R. (1995). Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence* .

Kilicay Ergin, N., & Dagli, C. H. (2008). In M. (. Jamshidi, *System of Systems: Innovations for the 21st Century.* Wiley & Sons, Inc.

Kilicay-Ergin, N., Enke, D., & Dagli, C. (2012). Biased trader model and analysis of financial market dynamics. *International Journal of Knowledge-based Intelligent Engineering Systems, Vol. 16* , 99-116.

Lane, J., & Dahmann, J. (2008). "Process evolution to support system of systems engineering. *Ultra-Large-Scale Software-Intensive Systems (ULSSIS).* Leipzig.

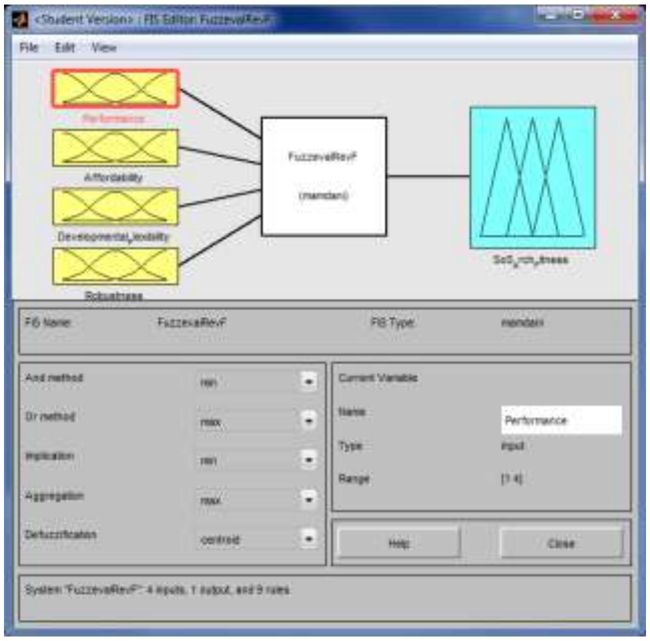
NDIA. (11 October 2011). *Best Practices Model for SoS Systems Engineering (SE) and Test & Evaluation (T&E), Draft for NDIA Strategic Initiative: Best Practices Model for SoS T&E.*

Weiss, W. (2008). Dynamic Security: An Agent-based Model for Airport Defense. *Proceedings of the Winter Simulation Conference.*

APPENDIX A ACRONYMS

Acronym	Definition
ABM	Agent based model
AIM	Agent interaction management
ASD(NII)	Assistant Secretary of Defense for Networks and Information Integration (disestablished Jan 2012)
CM	Cooperation management
CONOPS	Concept of Operations
CPAR	Contractor Performance Assessment Report
DoD	Department of Defense
DoDI	Department of Defense Instruction
FAM	Fuzzy Associative Memory
FY	Fiscal Year
GA	Genetic Algorithm
ISR	Intelligence, Surveillance and Reconnaissance
KPP	Key Performance Parameter
MS	Master of Science
MTPs	Methods, Tools and Processs
NDIA	National Defense Industrial Association
OPC	Own Process Control
RDT&E	Research, Development, Test and Evaluation
RT	Research Task
SE	Systems Engineering
SERC	Systems Engineering Research Center
SoS	System of Systems
SPO	System Program Office
US	United States

APPENDIX B FUZZY ASSOCIATIVE MEMORY



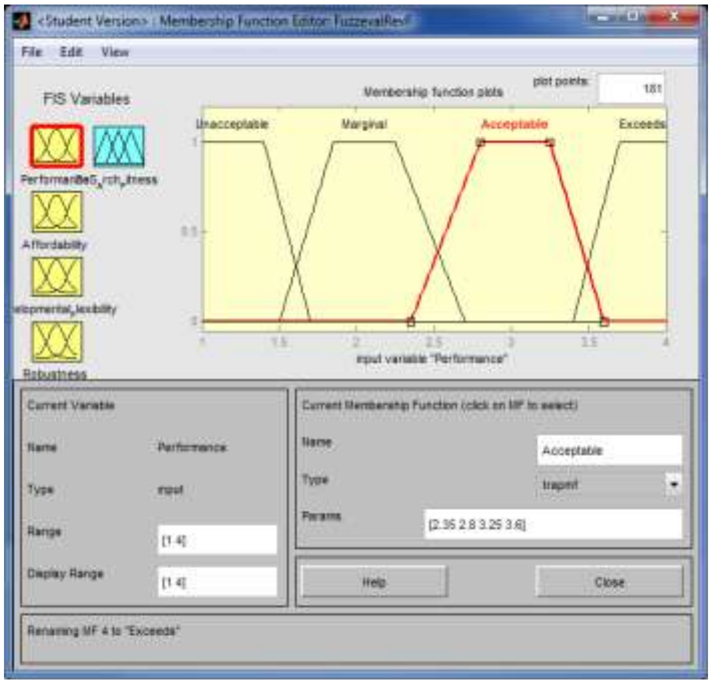
FIS Name: FuzzevalRevF
FIS Type: mamdani

And method: min
Or method: max
Implication: min
Aggregation: max
Defuzzification: centroid

Current Variable
Name: Performance
Type: input
Range: [1 4]

System "FuzzevalRevF": 4 inputs, 1 output, and 9 rules

Figure 19. MATLAB Fuzzy Toolbox.



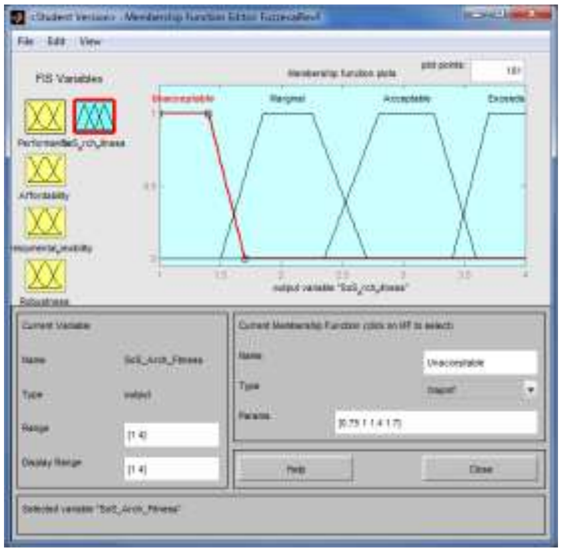
plot points: 181

Current Variable
Name: Performance
Type: input
Range: [1 4]
Display Range: [1 4]

Current Membership Function (click on MF to select)
Name: Acceptable
Type: trapmf
Params: [2.35 2.8 3.25 3.6]

Renaming MF 4 to "Exceeds"

Figure 20. Membership functions (same for all input attributes)

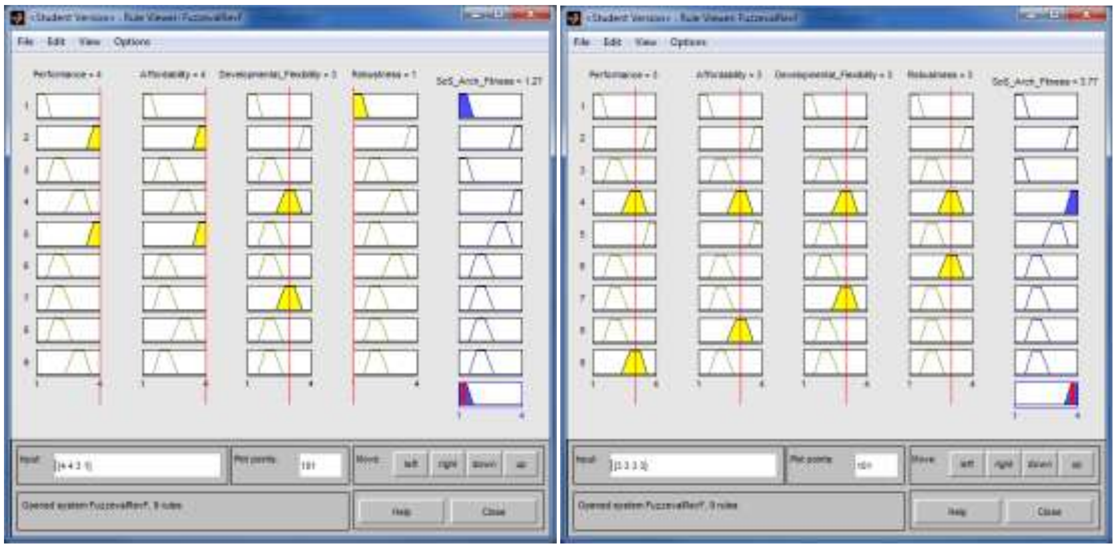


Current Variable
Name SoS_Arch_Fitness
Type output
Range [1 4]
Display Range [1 4]

Current Membership Function (click on MF to select)
Name Unacceptable
Type trapmf
Params [0.75 1 1.4 1.7]

Selected variable "SoS_Arch_Fitness"

Figure 21. Membership function for SoS Architecture evaluation (output)



Input: [4 4 3 1]
Opened system FuzzevalRev1, 9 rules

Input: [3 3 3 3]
Opened system FuzzevalRev1, 9 rules

Figure 22. Example fuzzy rule contribution from each Attribute

The scaling of "goodness," also termed the "universe of discourse," is between 1 = Unacceptable and 4 = Exceeds for plotting purposes within the MATLAB Fuzzy Toolbox.

Table 13. All possible evaluations of 4 Attributes; 1=Unacceptable, 2=Marginal, 3=Acceptable, 4=Exceeds

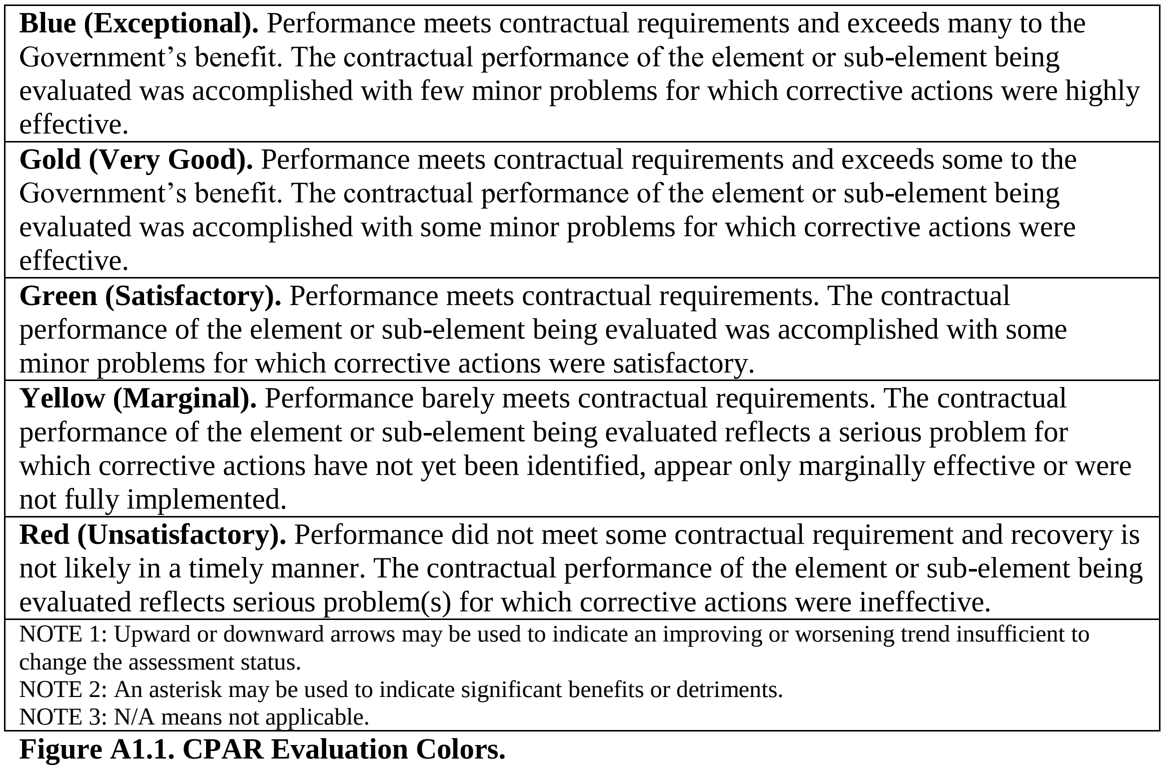
P	A	F	R	SoS
1	1	1	1	1
1	1	1	2	1
1	1	1	3	1

P	A	F	R	SoS
1	1	1	4	1
1	1	2	1	1
1	1	2	2	1
1	1	2	3	1
1	1	2	4	1
1	1	3	1	1
1	1	3	2	1
1	1	3	3	1
1	1	3	4	1
1	1	4	1	1
1	1	4	2	1
1	1	4	3	1
1	1	4	4	1
1	2	1	1	1
1	2	1	2	1
1	2	1	3	1
1	2	1	4	1
1	2	2	1	1
1	2	2	2	1
1	2	2	3	1
1	2	2	4	1
1	2	3	1	1
1	2	3	2	1
1	2	3	3	1
1	2	3	4	1
1	2	4	1	1
1	2	4	2	1
1	2	4	3	1
1	2	4	4	1
1	3	1	1	1
1	3	1	2	1
1	3	1	3	1
1	3	1	4	1
1	3	2	1	1
1	3	2	2	1
1	3	2	3	1
1	3	2	4	1
1	3	3	1	1
1	3	3	2	1
1	3	3	3	1
1	3	3	4	1
1	3	4	1	1
1	3	4	2	1
1	3	4	3	1
1	3	4	4	1
1	4	1	1	1
1	4	1	2	1
1	4	1	3	1
1	4	1	4	1
1	4	2	1	1
1	4	2	2	1
1	4	2	3	1
1	4	2	4	1
1	4	3	1	1
1	4	3	2	1
1	4	3	3	1
1	4	3	4	1
1	4	4	1	1
1	4	4	2	1
1	4	4	3	1
1	4	4	4	1
2	1	1	1	1
2	1	1	2	1
2	1	1	3	1
2	1	1	4	1
2	1	2	1	1
2	1	2	2	1
2	1	2	3	1
2	1	2	4	1
2	1	3	1	1
2	1	3	2	1
2	1	3	3	1
2	1	3	4	1
2	1	4	1	1
2	1	4	2	1
2	1	4	3	1
2	1	4	4	1
2	2	1	1	1
2	2	1	2	1

P	A	F	R	SoS
2	2	1	3	1
2	2	1	4	1
2	2	2	1	1
2	2	2	2	1
2	2	2	3	2
2	2	2	4	3
2	2	3	1	1
2	2	3	2	2
2	2	3	3	3
2	2	3	4	3
2	2	4	1	1
2	2	4	2	3
2	2	4	3	3
2	2	4	4	3
2	3	1	1	1
2	3	1	2	1
2	3	1	3	1
2	3	1	4	1
2	3	2	1	1
2	3	2	2	2
2	3	2	3	3
2	3	2	4	3
2	3	3	1	1
2	3	3	2	3
2	3	3	3	3
2	3	3	4	3
2	3	4	1	1
2	3	4	2	3
2	3	4	3	3
2	3	4	4	3
2	4	1	1	1
2	4	1	2	1
2	4	1	3	1
2	4	1	4	1
2	4	2	1	1
2	4	2	2	3
2	4	2	3	3
2	4	2	4	3
2	4	3	1	1
2	4	3	2	3
2	4	3	3	3
2	4	3	4	3
2	4	4	1	1
2	4	4	2	3
2	4	4	3	3
2	4	4	4	3
3	1	1	1	1
3	1	1	2	1
3	1	1	3	1
3	1	1	4	1
3	1	2	1	1
3	1	2	2	1
3	1	2	3	1
3	1	2	4	1
3	1	3	1	1
3	1	3	2	1
3	1	3	3	1
3	1	3	4	1
3	1	4	1	1
3	1	4	2	1
3	1	4	3	1
3	1	4	4	1
3	2	1	1	1
3	2	1	2	1
3	2	1	3	1
3	2	1	4	1
3	2	2	1	1
3	2	2	2	2
3	2	2	3	3
3	2	2	4	3
3	2	3	1	1
3	2	3	2	3
3	2	3	3	3
3	2	3	4	3
3	2	4	1	1
3	2	4	2	3
3	2	4	3	3
3	2	4	4	3
3	3	1	1	1

P	A	F	R	SoS
3	3	1	2	1
3	3	1	3	1
3	3	1	4	1
3	3	2	1	1
3	3	2	2	3
3	3	2	3	3
3	3	2	4	3
3	3	3	1	1
3	3	3	2	3
3	3	3	3	4
3	3	3	4	3
3	3	4	1	1
3	3	4	2	3
3	3	4	3	3
3	3	4	4	3
3	4	1	1	1
3	4	1	2	1
3	4	1	3	1
3	4	1	4	1
3	4	2	1	1
3	4	2	2	3
3	4	2	3	3
3	4	2	4	3
3	4	3	1	1
3	4	3	2	3
3	4	3	3	3
3	4	3	4	3
3	4	4	1	1
3	4	4	2	3
3	4	4	3	3
3	4	4	4	3
4	1	1	1	1
4	1	1	2	1
4	1	1	3	1
4	1	1	4	1
4	1	2	1	1
4	1	2	2	1
4	1	2	3	1
4	1	2	4	1
4	1	3	1	1
4	1	3	2	1
4	1	3	3	1
4	1	3	4	1
4	1	4	1	1
4	1	4	2	1
4	1	4	3	1
4	1	4	4	1
4	2	1	1	1
4	2	1	2	1
4	2	1	3	1
4	2	1	4	1
4	2	2	1	1
4	2	2	2	3
4	2	2	3	3
4	2	2	4	3
4	2	3	1	1
4	2	3	2	3
4	2	3	3	3
4	2	3	4	3
4	2	4	1	1
4	2	4	2	3
4	2	4	3	3
4	2	4	4	3
4	3	1	1	1
4	3	1	2	1
4	3	1	3	1
4	3	1	4	1
4	3	2	1	1
4	3	2	2	3
4	3	2	3	3
4	3	2	4	3
4	3	3	1	1
4	3	3	2	3
4	3	3	3	3
4	3	3	4	3
4	3	4	1	1
4	3	4	2	3
4	3	4	3	3
4	3	4	4	3

P	A	F	R	SoS
4	4	1	1	1
4	4	1	2	1
4	4	1	3	1
4	4	1	4	1
4	4	2	1	1
4	4	2	2	3
4	4	2	3	3
4	4	2	4	3
4	4	3	1	1
4	4	3	2	3
4	4	3	3	3
4	4	3	4	3
4	4	4	1	1
4	4	4	2	3
4	4	4	3	3
4	4	4	4	4



Blue (Exceptional). Performance meets contractual requirements and exceeds many to the Government's benefit. The contractual performance of the element or sub-element being evaluated was accomplished with few minor problems for which corrective actions were highly effective.
Gold (Very Good). Performance meets contractual requirements and exceeds some to the Government's benefit. The contractual performance of the element or sub-element being evaluated was accomplished with some minor problems for which corrective actions were effective.
Green (Satisfactory). Performance meets contractual requirements. The contractual performance of the element or sub-element being evaluated was accomplished with some minor problems for which corrective actions were satisfactory.
Yellow (Marginal). Performance barely meets contractual requirements. The contractual performance of the element or sub-element being evaluated reflects a serious problem for which corrective actions have not yet been identified, appear only marginally effective or were not fully implemented.
Red (Unsatisfactory). Performance did not meet some contractual requirement and recovery is not likely in a timely manner. The contractual performance of the element or sub-element being evaluated reflects serious problem(s) for which corrective actions were ineffective.
NOTE 1: Upward or downward arrows may be used to indicate an improving or worsening trend insufficient to change the assessment status. NOTE 2: An asterisk may be used to indicate significant benefits or detriments. NOTE 3: N/A means not applicable.

Figure A1.1. CPAR Evaluation Colors.

Figure 23. Example CPAR evaluation gradations (Department of the Navy, 1997)

APPENDIX C WAVE MODEL DEFINITIONS

- System – a system is one of the individual entities (systems) that comprise the overall System of Systems (SoS)
- SoS – the SoS is the overarching entity that is comprised of the individual systems and their interactions. "A set or arrangement of systems that results when independent and useful systems are integrated into a larger system that delivers unique capabilities" .
  - Directed Type of SoS – is type of SoS where the individual systems are subordinated to the SoS. This type of SoS has SoS objectives, management, funding, and authority. This type of SoS exhibits the highest influence on the individual systems.
  - Acknowledged Type of SoS – is a type of SoS where the individual systems retain their own management, funding, and authority in parallel with the SoS development. This type of SoS has SoS objectives, management, funding, and authority but does not exhibit as much influence over the individual systems as does the directed type of SoS. (Dahmann, Rebovich, Lane, Lowry, & . Baldwin, 2011)
  - Collaborative Type of SoS – is a type of SoS where the individual systems "voluntarily work together to address shared or common interest"(NDIA, 11 October 2011). This type of SoS does not have any top down objectives, management, authority, or funding. This type of SoS must depend on the good-will of each individual system to support SoS common goals. The SoS basically gets whatever the individual systems provide.
  - Virtual Type of SoS – is a type of SoS similar to the collaborative SoS, but the individual systems are not aware of each other (NDIA, 11 October 2011). This type of SoS has the least amount of influence over the individual systems and must accept whatever the individual systems provide.
- SoS Domain – is a specific area in which the SoS exits and whose objectives are specific to that domain.
  - SoS Mission Domain – is a set of individual systems "working together to provide a broader capability or mission" (NDIA, 11 October 2011). The SoS objectives are focused on a specific mission.
  - SoS Platform Domain – is a "military platform (e.g. ship, aircraft, satellite, ground vehicle) equipped with additional independent systems (e.g. sensor, weapons, communication) needed to meet platform objectives" (NDIA, 11 October 2011). In this case, the SoS objectives are specific to the platform.
  - SoS IT-Based Domain – is a set of systems that are networked to provide information "to support operations within or across platforms or systems to meet mission or platform objectives" (NDIA, 11 October 2011). The objectives of this domain are on the information passing across the individual systems.
- Capability (Capability Objectives) – a capability is a functionality of an individual system or SoS. It can be high level as in a SoS or more detailed functionality as in an individual system.
- Performance – performance defines and quantifies how well SoS or individual system must accomplish a particular capability.
- Mission Plan – a mission plan is the big picture that a set of capability objectives supports. It is a function of capabilities and associated performance.

- Wave – a wave (or “bus-stop”) is a development approach where the capabilities of the individual systems are incorporated into the larger SoS on a periodic basis similar to the way that waves periodically flow on the shore or the way a bus periodically stops at a bus-stop. For each capability, the individual system decides which SoS wave will receive this capability. If an individual system cannot provide a capability for a specific wave, then the SoS and the individual system can evaluate the risk of missing a wave. In the SoS, there are certain time-based delivery points and the individual systems target their deliveries for these points (Dahmann, Rebovich, Lane, Lowry, & . Baldwin, 2011). Figure 1 shows the first wave in a wave model development approach.
- Epoch – the time from one wave to the next. The period of time between each capability delivered to the SoS from an individual system. It is a Wave interval.
- Environment – outside the scope of the SoS (add definition); context of the SoS
  - National Priority – a national priority is something that is of utmost importance to the nation as a whole. There could be multiple national priorities and they could affect each other and even be in the opposite direction of effect thus cancelling each other out.
  - Threat Assessment – a threat assessment is a decision that a specific entity can negatively impact the security of the nation. This negative impact could be in the form of loss of life, loss of property, of loss of sensitive information. There could be multiple threat assessments which could impact each other and other agents in the model.
  - Changes in SoS funding
- System fitness – Fitness of a system is an objective function used to evaluate how close a given system alternative is to achieving the SoS mission
- System motivation – System motivation is the intent of an individual system which can exhibit tendencies anywhere on a spectrum of sociological attitudes ranging from
  - Hostile intentions can be exhibited toward specific individuals or toward the collective. This translates into a SoS participant actively seeking retribution against another participant or seeking to disturb the functionality of the SoS.
  - Selfish individuals pursue the most efficient realization of their goals without regard for their impact on others
  - Utopian participants are those who place the operation of the collective SoS ahead of their individual goals

- Cooperation – Individual system agrees to be part of the SoS by providing certain capabilities and associated performance values.
- System’s willingness to cooperate – System’s willingness to cooperate is a variable which measures the degree of willingness of the individual system to be part of the SoS.
- System’s ability to cooperate – System’s ability to cooperate is a variable which depends on system resources (i.e, funding) that will allow the individual system to be part of the SoS.
- Funding – Percentage of discretionary funding available for implementing the SoS.