Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 16 (2013) 11 – 19

Procedia Computer Science



Conference on Systems Engineering Research (CSER'13)
Eds.: C.J.J. Paredis, C. Bishop, D. Bodner, Georgia Institute of Technology, Atlanta, GA, March 19-22, 2013.

# Model Based Systems Engineering for System of Systems Using Agent-Based Modeling

Paulette Acheson[a], Cihan Dagli[a]*, Nil Kilicay-Ergin[b]

[a]*Missouri University of Science & Technology, Rolla, MO,65409,USA*
[b]*Penn State University, Malvern, PA, 19355, USA*

**Abstract**

Model Based Systems Engineering (MBSE) follows a model centric approach in contrast to the traditional document centric approach. The complexity of SoS development lends itself nicely to a model centric approach, especially a model that can represent the independence of the systems that comprise the SoS. An agent-based model provides a framework where the SoS and each system are independent entities with individual dynamics and interactions. The System of Systems (SoS) development depends on contributions from the individual systems each having their own agenda and priorities. System-to-system interactions are often necessary to accomplish the overall objectives and capabilities of the SoS. This research investigates the SoS development of an Acknowledged SoS with its associated individual systems and represents this development in an Agent-Based Model (ABM). The ABM includes decision models for the individual system agents that capture system dynamics and system-to-system negotiations as well as system to SoS negotiations. The ABM incorporates the key factors that influence SoS and individual systems' dynamics and enables systems engineers to try different scenario inputs and analyze the overall dynamics.


Selection and/or peer-review under responsibility of Georgia Institute of Technology

*Keywords*: system of systems; model based systems engineering; agent-based model

## 1. Introduction

Model Based Systems Engineering (MBSE) is becoming a more popular approach to system development as it provides a communication and verification that transcends the levels of development. MBSE uses a model or set of models to document and communicate from the system requirements level down to the software implementation level. When a set of models is used, the models are connected and dependent on each other so that changes in one model automatically require the update of the set of models. This interdependence among the models provides the extra level of verification that is not present in document-centered system development.

SoS development does not follow the normal system development process. SoS capability is based on the contributions of the individual systems that comprise the SoS. This interdependence between the SoS and the individual systems makes a document-centric development impractical as an exorbitant effort is required to maintain

* Corresponding author. Tel.: +0-310-336-3789; fax: +0-310-336-4070.
*E-mail address*: pbatk5@mail.mst.edu.

1877-0509 © 2013 The Authors. Published by Elsevier B.V.
Selection and/or peer-review under responsibility of Georgia Institute of Technology
doi:10.1016/j.procs.2013.01.002

the SoS development documentation. On the other hand, SoS development being characterized by the individual systems' capabilities, can be expressed in terms of models and approached using MBSE.

This research provides an Agent-Based Model (ABM) of SoS development. The ABM is based on the Wave Model [1] and can be executed using different initial SoS architectures and different input parameters. This being an agent-based model the result is not a simple output as is found with discrete event models. But rather the ABM can be used to determine how different behaviors in different systems along with external parameters such as funding, priorities, performance, etc. can affect the actual final SoS capabilities, parameters, funding, etc..

## 2. Background

System of Systems development for the DoD does not follow the normal DoD acquisition process as defined in DoDI 5000.02. Instead SoS development has been shown to follow a Wave Model process [1] as shown in Figure 1. The Wave Model of SoS development is like a bus-stop where the bus picks up passengers at a specific time interval. A passenger arriving at the bus-stop after the bus has left, must wait for the next time the bus arrives. Similarly, individual systems provide capabilities to the SoS but those capabilities must arrive at the height of the wave in order for the system capability to integrate into the overall SoS. A system that can provide a capability that arrives after the SoS integration point (or height of the wave), must wait until the next wave for incorporation into the SoS.

There are multiple types of SoS, such as Directed or Acknowledged [1], and the type of SoS determines the role and authority of the SoS in the SoS development. An Acknowledged SoS is by definition dependent on the individual systems to provide the capabilities that the SoS requires [1]. The SoS has objectives, management, and funding but does not have the strong influence over the individual systems that a Directed SoS would have [1]. So the actual capabilities and performance of the SoS is determined by the cooperation of the individual systems with the SoS and with each other. In this research, an Acknowledged SoS is chosen as the overarching focus of the model based systems engineering product.

In this research, the SoS Architecture is a key to the SoS development. The importance of architecture to MBSE was shown in [3]. The ABM begins with an initial proposed SoS Architecture and this architecture reflects the systems providing capabilities as well as the system-to-system interfaces. This SoS Architecture structure is based on the chromosome structure described in [7].

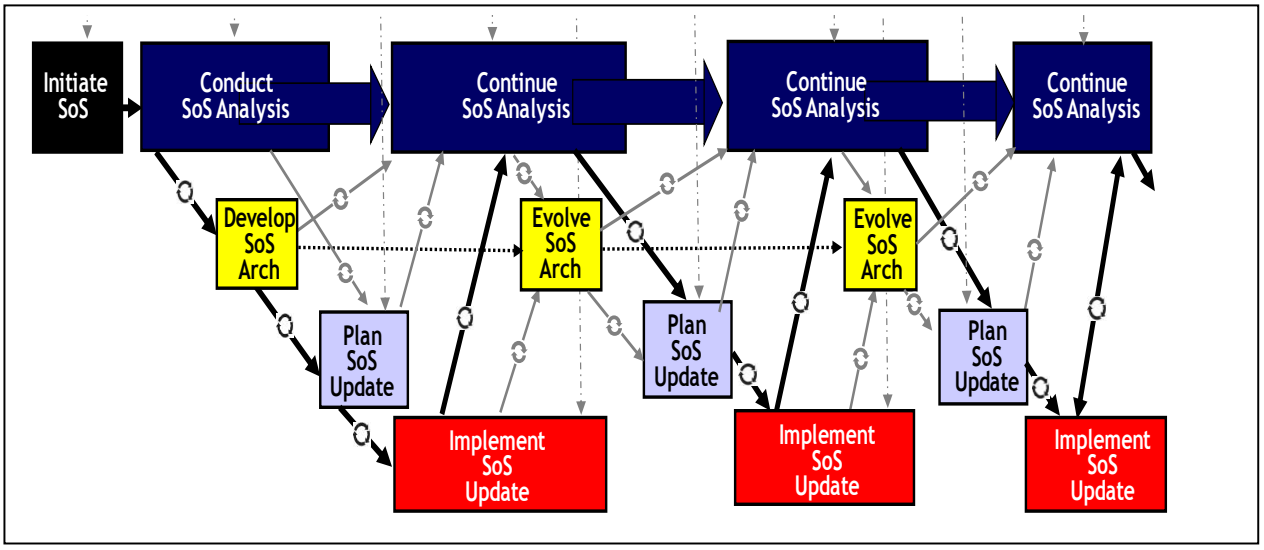


Figure 1. Wave Model of SoS Development

## 3. Approach

The essence of the Acknowledged SoS development is the independence of the individual systems. This independence invokes the idea of an Object-Oriented software development approach where each Object stands on its own and has a specified interface to the other objects in the system. Because of this similarity between SoS development and Object-Oriented software development, the Object-Oriented Systems Approach (OOSA) as specified in [6] was used to design the model structure.

The independence of the individual systems also leads to the choice of an agent-based model over the discrete event or system dynamics models. The agent-based model implementation is done using an Object-Oriented programming language [5], such as C++ or Java, in which each agent or object becomes its own independent entity. Thus the selection of an agent-based model more accurately represents the real world systems which each have their own agenda, funding, and priorities. In terms of SoS development, the objects become the agents in the ABM. Figure 3 shows the Agent–Based Model architecture for SoS development.

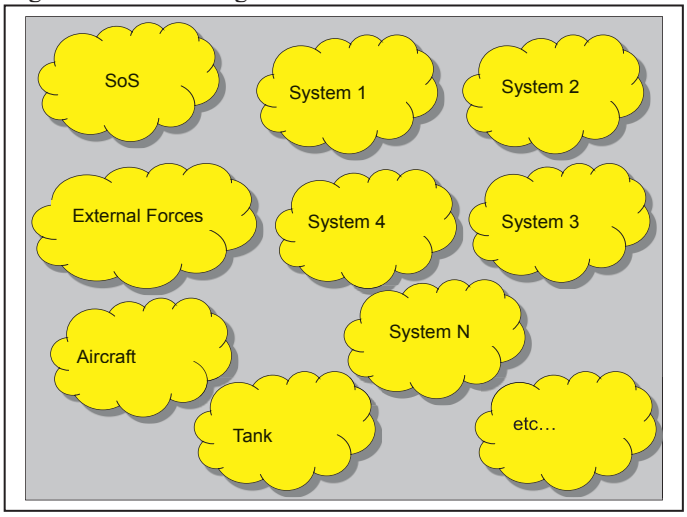


Figure 2. Objects in the Object-Oriented System Approach for SoS Development

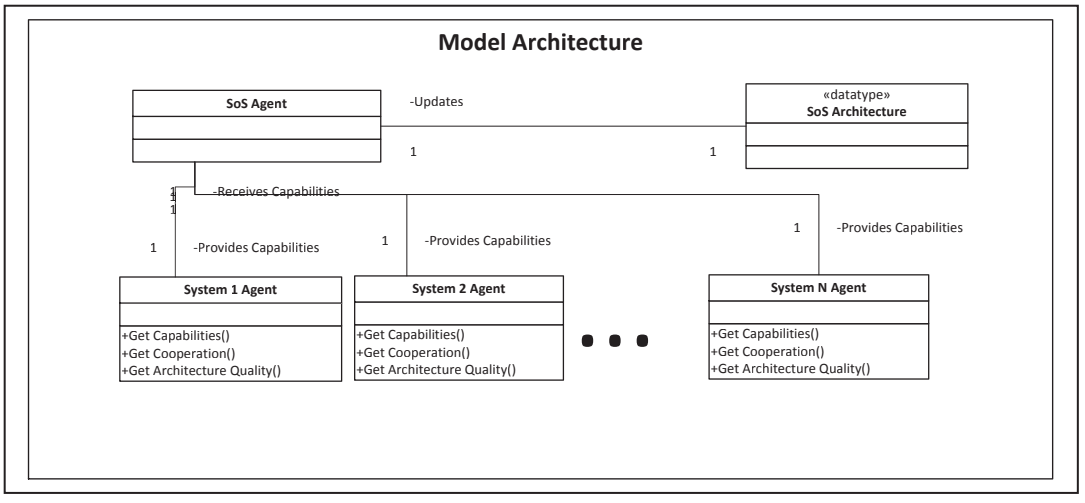


Figure 3. Model Architecture for SoS Development

## 4. Agent-Based Model

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.

The ABM has one SoS agent that reflects the characteristics and behavior of an acknowledged SoS and a System agent that embodies the funding, priorities, capabilities, and behavior of the individual systems. This ABM currently has ten individual System Agents, but future work can have a different number of System Agents. The capabilities in the SoS are derived from the capabilities provided by the individual systems. The SoS Architecture is a data item implemented as the chromosome described in [7]. The SoS capabilities, funding, performance, and priorities are implemented as variables within the SoS agent. The individual system capabilities, funding, performance, priorities are implemented within the System agent.

Figure 4 shows the sequence diagram for the ABM. The SoS sends a request to the Systems for specific capabilities. The individual Systems provide a response that indicates level of cooperation and when it will provide the capabilities. The SoS creates the Updated SoS Architecture and evaluates the architecture for architecture attributes of Affordability, Flexibility, Robustness, and Performance. Based on the responses from the individual systems and the SoS Architecture quality, the SoS makes a decision to proceed with what the systems provided or to negotiate with the systems for different capabilities. The sequence can iterate until a specific architecture quality is achieved or can cycle as the user collects data on the sensitivity of the architecture to SoS and/or System parameters such as funding, capabilities, or performance.

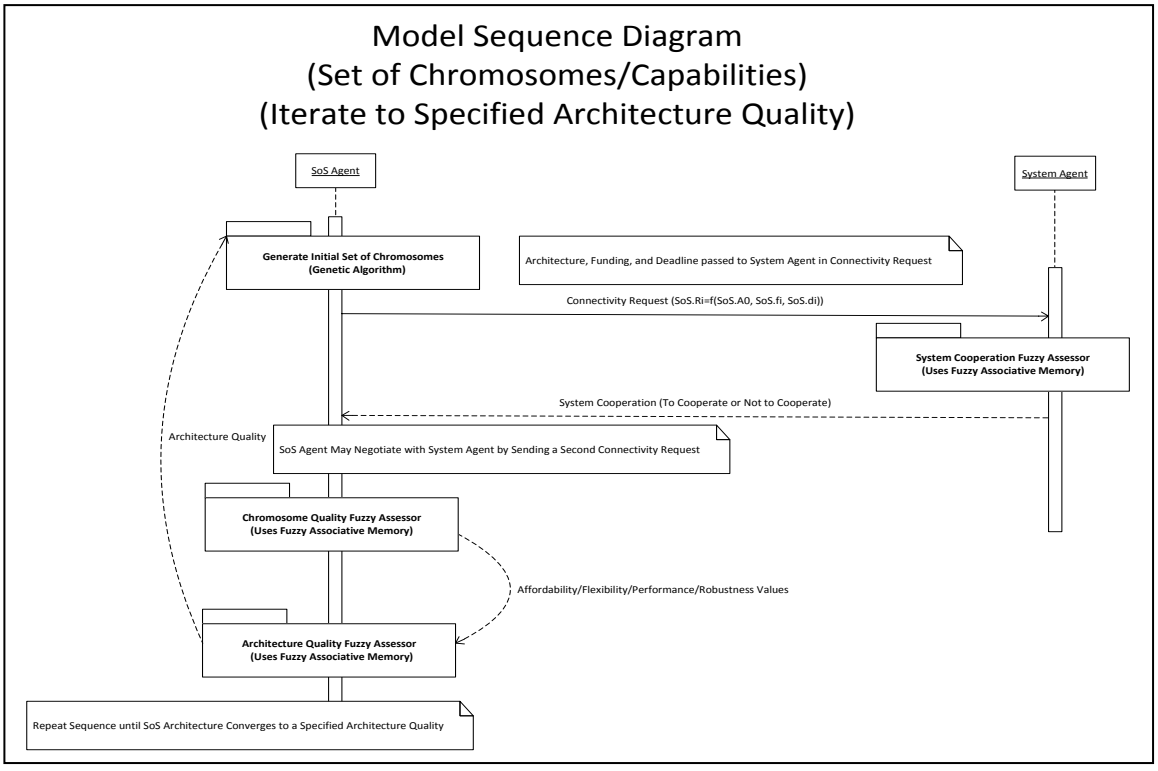


Figure 4. Sequence Diagram for the Agent-Based Model

### 4.1. SoS Agent Structure

The SoS agent follows the SoS development as described in the Wave Model [1] in Figure 1 [1]. 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. Figure 5 depicts the states within the SoS agent.

The SoS agent starts with a proposed SoS Architecture and set of desired capabilities that can be specified by the user. There are multiple methods of determining the initial architecture such as using a genetic algorithm to provide an optimal initial architecture [8] but that is beyond the scope of this research. Having the initial SoS Architecture, the SoS agent requests capabilities from the individual system agents. Based on the responses from the individual systems, the SoS agent creates the Updated SoS Architecture that reflects which systems can cooperate with the SoS agent and provide the requested capabilities at the first "wave". The SoS agent can negotiate with systems to provide capabilities at subsequent waves when the system cannot provide the capability at the requested wave. The Actual SoS Architecture is input into the Architecture Roll Up which uses priorities and participation weights specified by the user and determines the Actual SoS Architecture values for Affordability, Flexibility, Robustness, and Performance. The Architecture Roll Up considers the architecture values from the individual systems and is a Fuzzy Inference System. The structure of the SoS agent is shown in Figure 6. Figure 7 has the structure of the Architecture Roll Up.

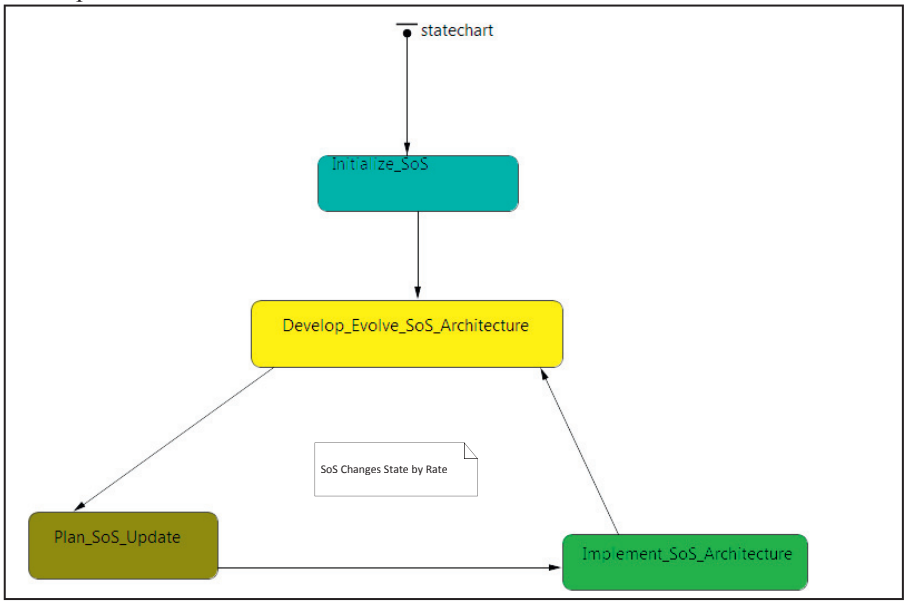


Figure 5. SoS Agent State Diagram

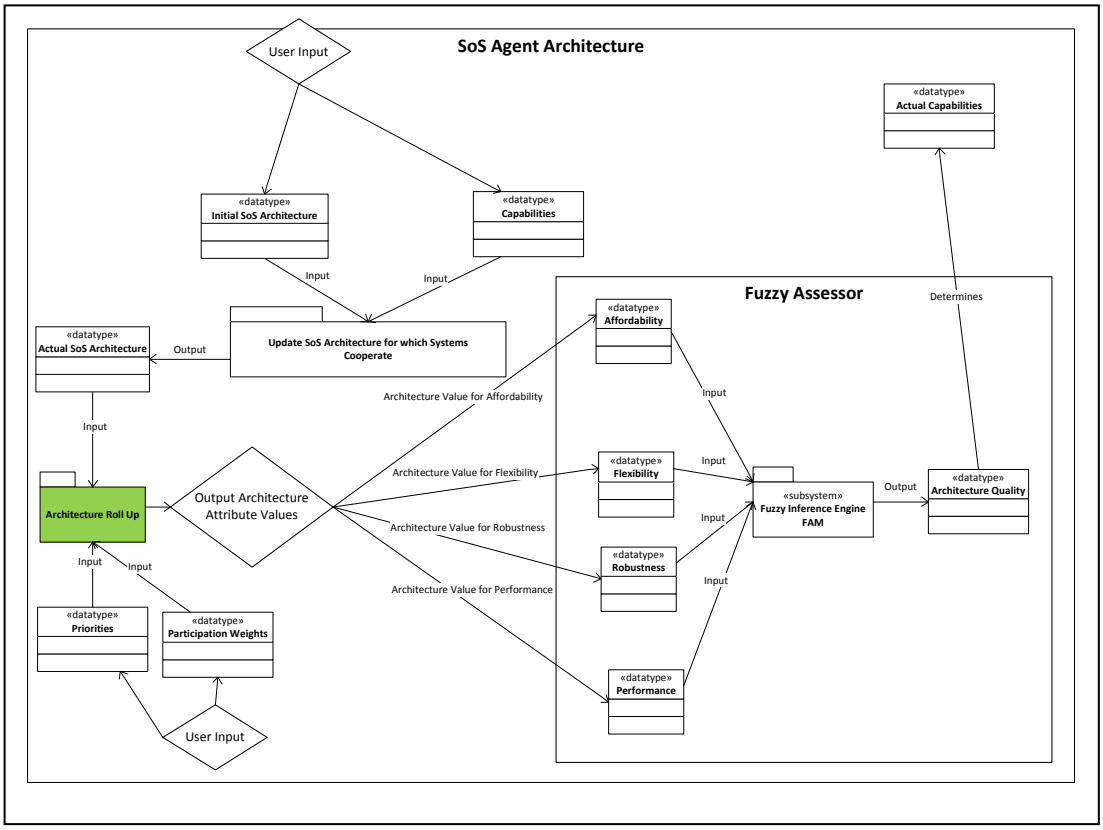


Figure 6. SoS Agent Architecture

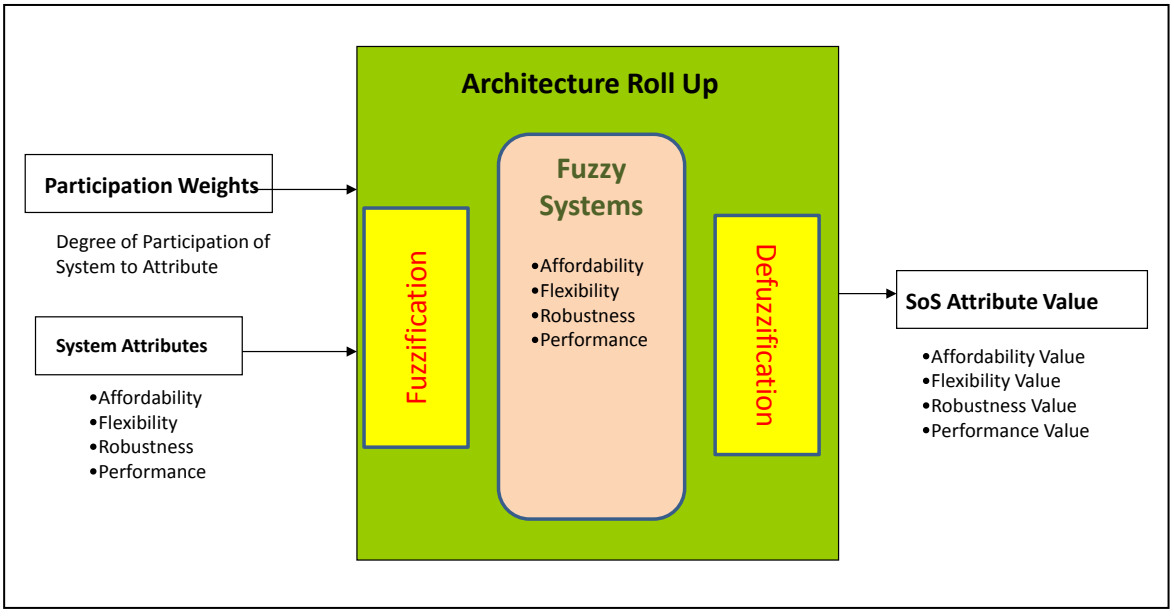


Figure 7. Architecture Roll Up Structure

*4.2. System Agent Structure*

The System agent embodies all the behavior and attributes of the individual systems. Systems either cooperate with the SoS request, do not cooperate with the SoS request, or are trying to decide whether to cooperate with the SoS request. The System agent reflects these states and also includes an initialization state. The System state diagram is in Figure 8. The Prep state exists so that the SoS agent can initialize before any systems begin processing. During the Maybe state the System agent is making the decision whether to cooperate with the SoS request. This decision is based on a inputs set by the user.

The System agent uses a Fuzzy Inference Engine (FIE) to determine cooperation with SoS. The FIE is based on user inputs as well as Fuzzy Systems that represent the following aspects of a system:

- Willingness to Evolve
- Acceptance of Future Vision
- Ability (resources) to Change
- Ability to Influence Change

In addition, the System agent can negotiate with other systems to provide the capabilities requested by the SoS. Figure 9 shows the architecture of the System agent.

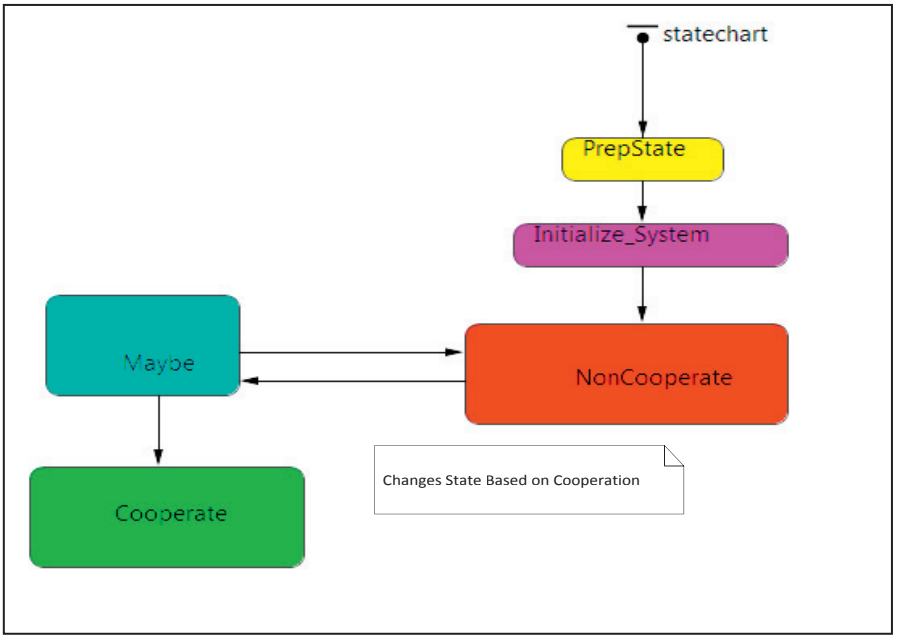


Figure 8. System Agent State Diagram

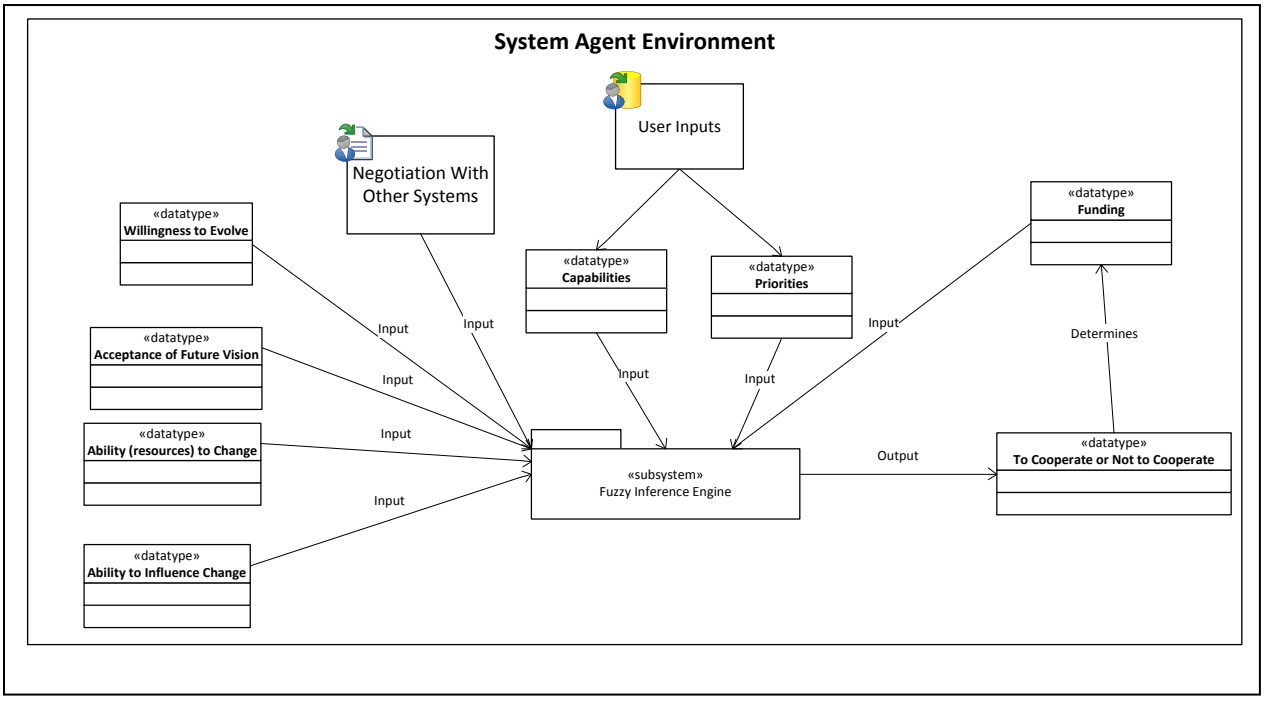


Figure 9. System Agent Architecture

## 5. Conclusion

An Agent-Based Model representing SoS development was presented. The ABM is flexible enough to represent any type of SoS development, however the details of the Fuzzy Inference Engine and Fuzzy Assessor along with the Fuzzy Membership functions, are dependent on the characteristics of the SoS domain. The ABM was executed using initial architectures generated by a genetic algorithm and an Intelligence Surveillance Reconnaissance (ISR) SoS as the domain. The results showed that multiple waves were necessary to produce a meaningful SoS Architecture. Running only one epic or one wave did not produce any architecture as the individual systems needed more time before they cooperated with the SoS.

## Acknowledgement

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.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense.

## References


1. 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.
2. Kilicay-Ergin, N., Acheson, P., Colombi, J., & Dagli, C. H. (2012). Modeling System of Systems Acquisition. 2012 IEEE International Conference on System of Systems Engineering. Genoa, Italy.
3. Butterfield, M.L.; Pearlman, J.S.; Vickroy, S.C.; , "A System-of-Systems Engineering GEOSS: Architectural Approach," *Systems Journal, IEEE* , vol.2, no.3, pp.321-332, Sept. 2008.
4. Gaudiano, P.; Bonabeau, E.; Shargel, B.; , "Evolving behaviors for a swarm of unmanned air vehicles," Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE , vol., no., pp. 317- 324, 8-10 June 2005.
5. Macal, C.M. & North, M.J. (2006). Tutorial on Agent-Based Modeling and Simulation Part 2: How to Model with Agents. Proceedings of the IEEE 2006 Winter Simulation Conference.
6. Acheson, P. (2010). Methodology for Object Oriented System Architecture Development. *IEEE Systems Conference.*
7. Acheson, P., Pape, L., Dagli, C., Kilicay-Ergin, N., Columbi, J., & Haris, K. (2012). Understanding System of Systems Development Using an Agent-Based Wave Model. *Procedia Computer Science* 2012.
8. Khaled Haris, Cihan H Dagli, Adaptive Reconfiguration of Complex System Architecture, Procedia Computer Science, Volume 6, 2011, Pages 147-152, ISSN 1877-0509, 10.1016/j.procs.2011.08.029.