11-12-2012

Understanding System of Systems Development Using an Agent-Based Wave Model

Paulette Acheson
Missouri University of Science and Technology

Louis Pape
Missouri University of Science and Technology

Cihan Dagli
Missouri University of Science and Technology

Nil Kilicay-Ergin
The Pennsylvania State University

John M. Colombi
Air Force Institute of Technology

See next page for additional authors

Recommended Citation

Acheson, P., Pape, L., Dagli, C. H., Kilicay-Ergin, N., Colombi, J. M., & Haris, K. (2012). Understanding System of Systems Development Using an Agent- Based Wave Model. Procedia Computer Science, 12(0), 21–30. https://doi.org/10.1016/j.procs.2012.09.024

Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 12 (2012) 21 – 30

Procedia Computer Science

Complex Adaptive Systems, Publication 2
Cihan H. Dagli, Editor in Chief
Conference Organized by Missouri University of Science and Technology
2012 - Washington D.C.

Understanding System of Systems Development Using an Agent-Based Wave Model

Paulette Acheson[a]*, Louis Pape[a], Cihan Dagli[a], Nil Kilicay-Ergin[b], John Columbi[c], Khaled Haris[a]

[a]*Missouri University of Science and Technoloy, Rolla, MO USA*
bPenn State University, Malvern, PA USA
[c]*Air Force Institute of Technology, Dayton, OH USA*

Abstract

System of Systems (SoS) development is a complex process that depends on the cooperation of various independent Systems [1]. SoS acquisition and development differs from that typical for a single System; it has been shown to follow a wave paradigm known as the Wave Model [2]. Agent based models (ABMs) consist of a set of abstracted entities referred to as agents, and a framework using simplified rules for simulating agent decisions and interactions. Agents have their own goals and are capable of perceiving changes in the environment. Systemic (global) behavior emerges from the decisions and interactions of the agents. This research provides a generic model of SoS development with a genetic algorithm and fuzzy assessor implemented in an agent based model. The generic SoS development follows the Wave Model. The genetic algorithm provides an initial SoS meta-architecture. The fuzzy assessor qualitatively evaluates SoS meta-architectures. The agent-based model implements the generic SoS development, the genetic algorithm, the fuzzy assessor, and independent SoS and system agents and shows the SoS development based on an initial set of conditions. A prototype model is developed to test the concept on a sample from the DoD Intelligence, Surveillance, and Reconnaissance (ISR) domain.

Keywords: agent based model; system of systems; SoS; human behavior; genetic algorithm; fuzzy systems

1. Introduction

System of Systems (SoS) architecting poses challenges, as the solution space of the design is much more open compared to a standalone system [3]. Existing analysis methodologies and tools scope the SoS problem space by assuming that there is a limited set of solutions [4][5]. However, the SoS problem boundary includes integration of technical systems as well as cognitive and social processes, which alter system behavior [6]. As mentioned before

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

 doi:10.1016/j.procs.2012.09.024

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 analysis and architecting phase [7].

Agent based models (ABM) consist of a set abstracted entities referred to as agents, and a framework for simulating agent decisions and interactions [8][9]. Agents have their own goals and are capable of perceiving changes in the environment. Simplified agent interaction rules may result in interesting group behavior. 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, such as Discrete Event modeling or System Dynamic modeling: 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 [10]. A key characteristic of an SoS is the independence of the individual systems that comprise the SoS [2]. The ABM has agents implemented as independent processes that more accurately reflects real world SoS development. The methodology is used in a wide range of application domains including financial markets [11], homeland security applications [12] and autonomous robots [13].

The goal of this research is to model SoS architecture evolution and acquisition based on the Wave Process Model and test the concept on the DoD Intelligence, Surveillance, and Reconnaissance (ISR) domain. The idea of Wave Planning was developed by Dombkins [1] and applied to the Trapeze Model of SoS Systems Engineering in order to illustrate the incremental and iterative process that characterizes SoS development [2]. Agent based modeling methodology is well suited to abstract behavioral aspects of the acquisition process in the special case of SoS. In this project, the SoS and the individual Systems are embodied in agents. The System agents represent themselves (e.g., Program Manager) as well as any other individual stakeholders. The wave model applies to acknowledged [14] SoS, thus there is a specific agent responsible for the SoS; that agent influences the cooperation of other System agents. An initial SoS mission is already determined and funds are allocated to the mission through a responsible organizational entity. The structure of the wave model is depicted in Figure 1 [2].

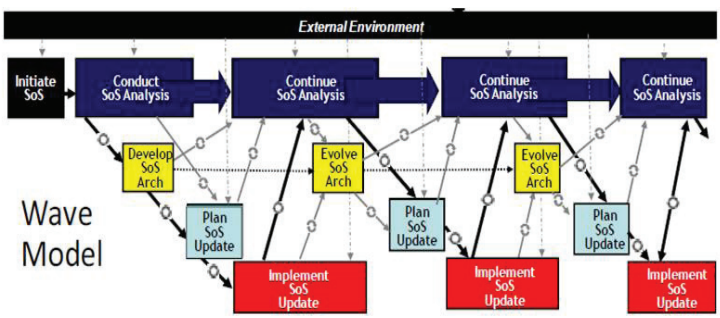

Figure 1. Wave Process Model [2]

The ABM in this paper consists of the SoS proposed development with the genetic algorithm, the fuzzy assessor applied in several places, and the actual implementation agreed among the System agents. The following sections describe in further detail these aspects of the model.

2. Proposed Agent Based Model

The proposed ABM consists of a generic SoS development, genetic algorithm, fuzzy assessor and an executable model. The generic SoS development is based on the Wave Model shown in Fig 1. The genetic algorithm creates an initial set of SoS meta-architectures, to initiate the SoS. The fuzzy assessor qualitatively evaluates the possible SoS meta-architectures in the SoS analysis step. The ABM operates on the proposed meta-architecture to develop an agreed SoS architecture. The SoS agent plans and the System agents implement the agreed update. Finally, the genetic algorithm and fuzzy assessor operate on the result again to evolve the SoS Architecture in successive updates.

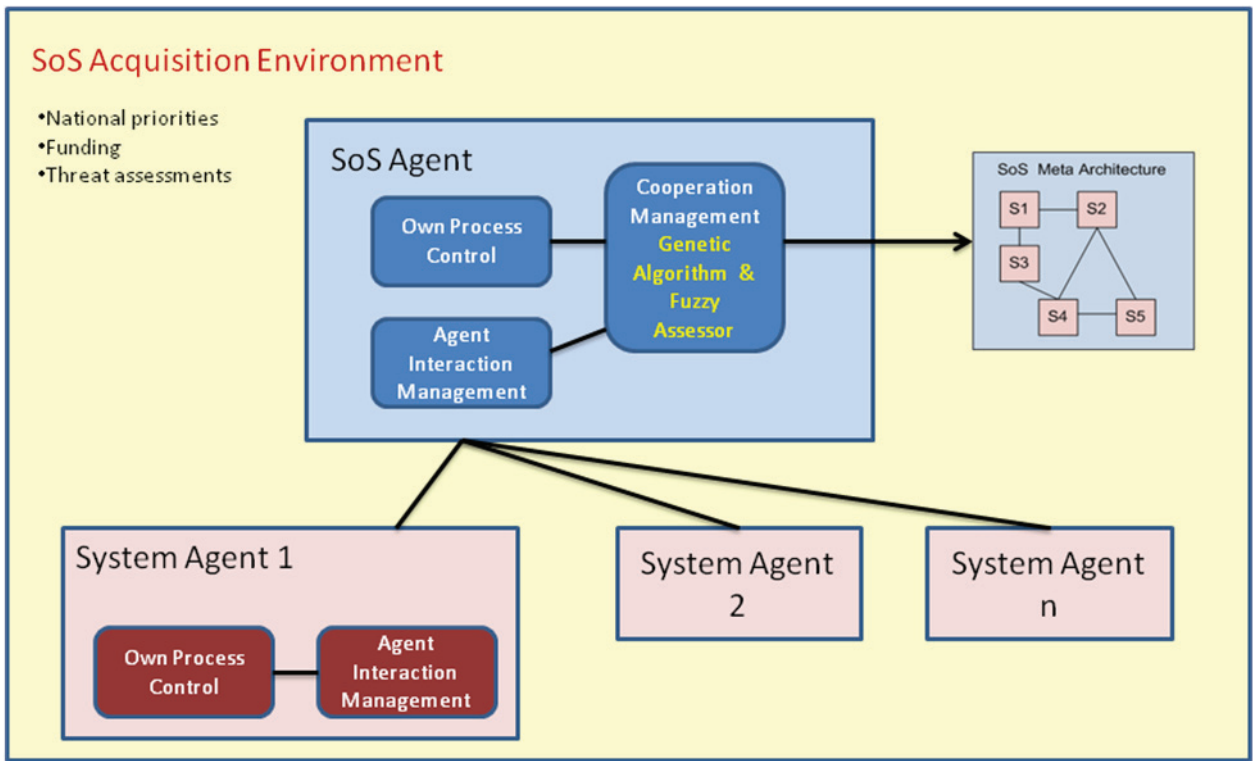

Figure 2. Overall Agent based Model of SoS Acquisition

2.1. SoS Acquisition Environment

The SoS agent and the individual System agents may be influenced by 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 appropriate 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: $E_T = E_0\sigma_T$

2.2. 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 an initial baseline SoS architecture is available. The SoS agent follows the six core SoS engineering activities outlined in the Wave Process Model [2] to develop the SoS. The SoS architecture evolves based on the behavior of individual systems as well as changes in the external environment.

2.3. 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 that comprise desired SoS capabilities and SoS performance parameters 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:

$$SoS.w_i = (w_1, w_2, ..., w_n)$$

$$SoS.P_i = (P_1, P_2, ..., P_n)$$

Determine SoS desired performance parameters:
Identify initial SoS Target Measures:

$$SoS.M_0 = [a_{ij}]_{n\times 3} \text{ where}$$

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

2.4. Conduct SoS Feasibility Analysis

The SoS agent tentatively allocates SoS capabilities to individual systems or group of systems. This allocation defines a baseline SoS architecture identifying individual systems and interfaces necessary to achieve the SoS target measures. Genetic Algorithms can generate alternative SoS architectures as chromosomes. 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. Program management measures such as schedule and 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 collaborate on 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 these abstracted model elements in mathematical notation.

3. Genetic Algorithm

An initial SoS architecture is first proposed at random so developers and acquisition officers can improve on it using the ABM, given an initial set of conditions and based on agent capabilities. An SoS architecture includes systems and interfaces that reflect these capabilities.

Then, genetic algorithms (GA) can be used to populate the meta-architecture with recommendations of better SoS architectures forming a trade space. In due course, the proposed architectures are individually evaluated by the fuzzy assessor. Eventually, the best architecture is selected. Genetic algorithms have been used in the past to generate optimum architectures in conjunction with Fuzzy Logic [15].

For genetic algorithms, all systems and interfaces can be represented side by side in a chromosome. In the chromosome structure, each degree of cooperation may be represented as a binary number representing the range of values possible. In our simplified model, each system or interface found in a possible architecture will be represented by a simple binary digit, with cooperation taking the value "1" while inability to cooperate will take a "0".

Incorporating the interfaces into the chromosome is based on the following idea. Let s_i be the System i where $i \in \{1,2, \dots, n\}$ and n is the total number of possible Systems. It is possible to have multiple systems in the set A of systems that are capable of providing the same capability. In addition, let s_{ij} be the interface between the systems i and j where also $j \in \{1,2, \dots, n\}$. Consider the set of all interfaces a graph G of size n. Then, it can be represented by its $n \times n$ adjacency matrix $S(G)$, whose elements s_{ij} are given by the following:

$$s_{ij} = s_{ji} = \begin{cases} 1, & \text{if exists an edge between vertices i and j} \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

Since an interface cannot connect a system to itself then: $s_{ii} = 0 \quad \forall i = j$

That is the diagonal of the adjacency matrix $S(G)$ will have the values zero. In addition, since an interface needs to be considered only once for the connection of two systems, only the upper triangle of the matrix needs to be considered, whereas the remaining elements of the matrix can take the value of zero. The following illustrates a

3 x 3 adjacency matrix representing interfaces in the upper triangle, which depicts existing interfaces between three systems:

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:

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

and

Evaluate the fitness of each individual SoS architecture chromosome:

$$\vec{SoS.C}_{g,n} \in SoS.C_g$$

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

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

$$Fitness.\vec{SoS.C}_{g,n} = SoS.B_T \text{ from Fuzzy Associative Memory}$$

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:

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

$$S(G) = \begin{Bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{Bmatrix}$$

Based on the above discussion, the chromosome can be simplified as follows so that only the upper triangular portion of the respective adjacency matrix is used. Figure 3 shows the chromosome format.

In order to address the performance value of the chromosome based on the key performance attributes in the prototype implementation, a matrix may be generated at random to relate the architecture attributes to the systems and interfaces identified in the chromosome. The tabulated values are then used as inputs to the fuzzy assessor discussed below."

3.1. 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 and circumstances. 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 that maps desired target measures to SoS architecture score/rating. The Fuzzy Assessor determines architecture score for the expected SoS architecture at wave time T. 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.

Figure 3. Chromosome Representation

Table 4: Develop and Evolve 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 is the number of FAM rules

$$(A_1, B_1), ..., (A_m, B_m)$$

$$System.Information_i = A_i$$

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. Fuzzy Assessor Model

The Fuzzy Assessor was designed to operate on a reduced set of four fuzzy attributes of: affordability, flexibility, robustness, and performance. A set of value membership function for each of the architecture attributes had to be developed. An even number of fuzzy values for each attribute prevents an evaluator from simply "punting" by choosing the middle value of an odd numbered set. The choice of membership functions affects the results of the Fuzzy Assessor so it was important that the membership functions accurately represent the attribute data. The determination of membership functions to use for these attributes was made based on structured interviews and discussions with stakeholders. The data from these interviews and questionnaires could be analyzed to find membership functions for other domains. The technique is extensible. In our example, the affordability values range from "totally unachievable," through "almost affordable," "looks quite affordable," to "could give resources back." The shape of the membership functions and amount of overlap in their shapes was tuned to be generically reasonable while covering existing data; other domains might use differently shaped membership functions.

The four attributes were chosen to represent a reasonable but extensible architectural evaluation basis, yet still be simple enough to comprehend the results within the model. Affordability was explained above. Flexibility has more to do with the development of the SoS and ability to change direction, and whether SoS objectives are achievable with varying degrees of participation from the component systems, overall resource support from the SoS agent, or changes in environment such as threat or competition. Robustness has more to do with the SoS success under varying degrees of participation by the component systems in the mission application. Finally, performance is evaluated against technical measures of the SoS goals (or requirements). A structured interview process with stakeholders by a subject matter expert facilitator can create domain appropriate scales for the fuzzy attribute values, such as that shown in Table 5.

4.1. 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 described in the following step. Table 6 summarizes the model elements in mathematical notation.

Table 5: Fuzzy Architecture Attribute value examples for ISR

\ Value Attribute\	**Unacceptable**	**Marginal**	**Acceptable**	**Exceeds performance**
Performance (KPPs for ISR SoS) Coverage (sq km/hr) Resolution # of channels Timeliness Adaptability	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) vs. 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 or lack of a single System	Between 10% and 30% degradation on one or more KPPs due to projected external disturbances or lack of a single System	Between 5% and 10% degradation in one or more KPPs due to projected external disturbances or absence of more than one System	Not more than 5% degradation in any KPP due to estimated external disturbances
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; most key SoS capability applications loosely coupled 50-75% of key functionality is allocated to software	Architecture is fluid and all key SoS capability applications loosely coupled > 75% of key functionality is allocated to software

4.2. Implement SoS

At the end of the wave cycle, the current SoS architecture is evaluated against initial SoS baseline architecture to identify functionality gaps. The SoS architecture score determined by the fuzzy assessor is also used in the analysis to identify performance gaps. This step is an input to planning SoS update step. Table 7 summarizes model elements in mathematical notation.

4.3. Continue SoS analysis

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

4.4. 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 to cooperate with the SoS agent's request. The decision depends on several factors including system's *willingness* to cooperate related to the degree of selfishness of the individual system or other constraints preventing cooperation, and system's *ability* to cooperate which depends on system's resources that will allow it 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 individual system decides not to cooperate, it has the option of requesting a later deadline to provide the

capability. Table 8 and Table 9 summarize the abstracted model elements in mathematical notation for individual systems.

Table 6: Plan SoS update

At wave time T:

Adjust/update SoS Target Measures:

$$SoS.\Delta C_i = (\Delta C_1, \Delta C_2, ..., \Delta C_n)$$

Capability update factor

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

Performance update factor

$$SoS.\Delta P_i = (\Delta P_1, \Delta P_2, ..., \Delta P_n)$$
$$SoS.\Delta P_i = f(E_t, SoS.Gap_T)$$

SoS Target measures update factor

$$SoS.Alpha_T = [a_{ij}]_{n\times 2} \text{ where}$$
$$a_{i1} = SoS.\Delta C_i \text{ and } a_{i2} = SoS.\Delta P_i$$
$$\textit{at T=0} \quad SoS.Alpha_T = 0$$

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

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

Table 7: Implement SoS architecture

At wave time T:

Gap analysis:

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

Table 8: 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 9: Reply back to SoS agent

If $System.coop_i = 1$

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

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

else Time to cooperate: $System.cooptime_i = t \text{ where } t > SoS.d_i$

5. Initial Implementation of the Agent-Based Model

An ABM implements the generic SoS model, the genetic algorithm, and the fuzzy assessor. The ABM consists of an SoS agent, a set of system agents, and the chromosome data structure (Figure 3) representing the SoS meta-architecture. The ABM was developed using an Object-Oriented System Architecture approach [16].

5.1. System Agent

The system agent represents the individual system that has some capability required by the SoS. The system agent has three states: Cooperation, Maybe, and Non-Cooperation. The Maybe state is the state when the system is evaluating its architecture and other factors to determine if it will cooperate with the SoS. The system agent can be influenced by different factors (such as social, political, economic, etc.) which can affect the willingness of the individual system to cooperate with the SoS request for capability. In addition, in order to provide a capability to the SoS, the system might have to modify its own architecture. Thus, the system must analyze the impact of providing the requested capability to the SoS.

5.2. SoS Agent

The SoS agent represents the overarching SoS that is being developed. The three SoS states were taken from the Wave Model described in [2]. These states are Develop/Evolve SoS Architecture, Plan SoS Update, and Implement SoS Architecture.

Initially, the SoS agent begins in the Develop/Evolve SoS Architecture state and the Architecture Algorithm presented above is run to obtain the starting SoS meta-architecture. Once the SoS meta-architecture is defined, the SoS agent requests capabilities from the individual systems. When the SoS agent receives the responses from the individual systems, the SoS agent updates the SoS meta-architecture based on the capabilities the individual systems provide.

The Fuzzy Architecture Assessor is used in the SoS agent to evaluate the resulting SoS meta-architecture. The inputs to the assessor are the degree of system agent cooperation and measures of the architecture attributes of flexibility, robustness, affordability, and performance.

5.3. Agent-Based Model Applicability

Using this ABM, SoS developers and acquisition officers can run “what if” scenarios to examine several SoS meta-architecture and the quality of the resulting architecture given a set of initial conditions and agent interaction rules. The agent-based model provides true independence between the development of the SoS and the development of the individual systems. The model was implemented in AnyLogic [17] because of its support of agent-based modeling and its basis in JAVA. The model can be provided as a JAVA applet that can be executed without an AnyLogic license.

6. Concluding Remarks

This research has provided an approach to investigating SoS development utilizing a generic SoS development, genetic algorithm, fuzzy assessor, and an agent-based model implementation. A genetic algorithm is used to populate the initial SoS meta-architecture and formulate the trade space of possible architectures with the optimum architectures. The fuzzy assessor is used to evaluate the set of SoS meta-architectures to determine the highest quality architectures. Finally, the agent-based model implements the generic SoS development, the genetic algorithm, and the fuzzy assessor into an executable application of independent agents that can represent the behavioral stakeholder and system influences on the SoS development. The agent-based model can be used by acquisition officers and government representatives to analyze the impact of different acquisition strategies and policies on the SoS development. In this way, the implementation provides data that supports the up-front systems engineering decisions made by acquisition officers. A prototype model developed is currently being tested on a sample of the DoD ISR domain.

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. David Dombkins, "Complex Project Management" Booksurge Publishing, South Carolina: 2007.
2. J. Dahmann, G. Rebovich, J. A. Lane, R. Lowry, K. Baldwin "An Implementers' View of Systems Engineering for Systems of Systems" Proceedings of IEEE International Systems Conference 2011, April 4-7, 2011, Montreal, Quebec, Canada.
3. Nil Kilicay Ergin and Cihan H Dagli, ed. Jamshidi M., "System of Systems Architecting" in System of Systems: Innovations for the 21st Century, Wiley & Sons Inc. 2008.
4. C. Dagli Editor "Complex Adaptive Systems", Procedia Computer Science Volume 6, Elsevier 2011. (http://www.sciencedirect.com/science/journal/18770509).
5. 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, 11 October 2011.
6. J. P. Dauby and S. Upholzer, "Exploring Behavioral Dynamics in Systems of Systems", in Complex Adaptive Systems, Editor C. Dagli, Procedia Computer Science Volume 6, Page 34-39, Elsevier 2011.
7. J. P. Dauby and C. H. Dagli, "The canonical decomposition fuzzy comparative methodology for assessing architectures," Systems Journal, IEEE, vol. 5, no. 2, pp.244-255, June 2011.
8. Brazier F. M. T., Jonker C. M. and Truer J., "Formalization of a cooperation model based on joint intentions" In J.P. Muller, M.J. Wooldridge, N. R. Jennings (eds), Intelligent Agents III (Proc. Of the Third International Workshop on Agent Theories, Architectures and Languages, ATAL'96), Lecture Notes in AI, Vol. 1193, Springer Verlag, ppg. 141-155, 1997.
9. Brazier F.M.T, Dunin-Keplicz B., Jennings N. R., and Treur J., "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, 1996.
10. Bonabeau E. "Agent based Modeling: Methods and Techniques for Simulating Human Systems," Proceedings of the National Academy of Sciences, Vol. 99, pp. 7280-7287, 2002.
11. Nil Kilicay-Ergin, David Enke and Cihan Dagli, "Biased trader model and analysis of financial market dynamics", International Journal of Knowledge-based Intelligent Engineering Systems, accepted June, 2011.
12. William Weiss, "Dynamic Security: An Agent-based Model for Airport Defense" Proceedings of the Winter Simulation Conference, 2008.
13. Donald Dudenhoeffer and Michael Jones "A Formation Behavior for Large Scale Micro-Robot Force Deployment" Proceedings of the 32nd Conference on Winter Simulation, 2000.
14. Office of the Deputy Under Secretary of Defense for Acquisition and Technology, Systems and Software Engineering. Systems Engineering Guide for Systems of Systems, Version 1.0. Washington, DC: ODUSD(A&T)SSE, 2008.
15. K. Haris and C. Dagli. "Architecture Trade-off Analysis and Reconfiguration", Proceedings Conference on Systems Engineering Research 2011 April 15-16, ISBN -978-0-9814980-1-0.
16. Paulette Acheson, "Methodology for Object-Oriented System Architecture Development" IEEE Systems Conference 2010.
17. AnyLogic. www.anylogic.com.