Agent-Based Modeling the Emergent Behavior of A System-of-Systems

John C. Hsu
Royal Academy of Engineering
Mark Price
Mechanical & Aerospace Engineering
Queens University
Belfast, Northern Ireland
jc0hsu@gmail.com

John R. Clymer
Jose Garcia-Jr.
Electrical Engineering
California State University
Fullerton, CA

Efrain Gonzalez
Southern California Edison
Irwindale, CA 91706

Abstract

Net-Centric Operations (NCO) is operated on a System-of-Systems (SoS) communication environment. Emergent behavior is one of the five SoS characteristics. Layered SoS architectures start from the top SoS-level containing emergent behavior. Conceptual agent-based modeling was presented to simulate the four principles of emergence. The simulations covered the condition of emergence, non-linear behavior and coupling relationship between agents. The four principles are interrelated. The emergent behavior has impacts on traditional systems engineering process. Neural network artificial intelligence may be needed to assist the understanding of emergent behaviors for architectural model development. Agent-based modeling needs further development and should be integrated with neural network and SysML.

Introduction

Net-Centric Operations (NCO) is an environment where collaboration between networks, systems and the elements within systems is possible. Network communications is the foundation to make systems linked or networked to share information across geographic borders. The basics of network communications are to transmit data throughout the network, between systems, devices or computers. This is a Systems-of-Systems (SoS) environment and the five characteristics of SoS presented by (Maier 1996) are listed in the following:

- Operational independence of the System Elements,
- Managerial independence of the System Elements,
- Evolutionary development,
- Emergent behavior, and
- Geographic distribution.

The Internet exhibits a rich set of emergent behaviors represented by the complex distributed applications that run on top of the communication substrate. The most complex of these is the World Wide Web, itself a virtual or collaborative system-of-systems. Many examples of emergent behavior of natural systems-of-systems were introduced and discussed in (Hsu 2007). Several definitions of emergence were introduced and the four Principles of Emergence were derived in (Hsu 2007). They will not be reiterated here.

The SoS displays a global complexity that cannot be adequately managed by hierarchical structures and central control. Understanding and responding to emergent SoS behavior – both positive and negative – is another important aspect of System-of-Systems Engineering (SoSE). This will place a premium on the discovery and clever use of design principles that produce emergent behavior through voluntary collaboration. To address such challenges, traditional Systems Engineering is necessary but not sufficient for the engineering of a system-of-systems.

Architecting a System-of-Systems

There is an emergent class of systems that are built from component systems in large scale. Prominent examples are NCO systems. Does the process of architecting and/or developing these systems differ in important ways from other types of systems? SoS should be distinguished from large but monolithic system by the independence of their component systems, their evolutionary nature, but most importantly, their emergent behaviors. The behavior and/or performance of the SoS cannot be represented in any form that is simpler than the SoS itself. There is no simple way (i.e. simpler than the SoS itself) to relate the functions of the parts to the functions of the whole. The traditional hierarchical functional decomposition is no longer valid due to the non-linear characteristics of emergent behavior; however, since the emergent behavior is non-existent in each component system, the hierarchical functional decomposition is still applicable to component system level.

The first challenge of architecting a SoS is at the top SoS level incorporating the emergent behavior. The next challenge is how to flow down the SoS level architecture to the component system level if they are hierarchical structures especially for the legacy systems. The model-based architecture-centric approach may be one of the answers. The customer requirements in the form of CONOPS model(s) are captured in the SoS architecture model(s). The component system architecture models can continue to capture CONOPS of component system level and the data flow from the upper SoS-level architecture. The subsystem architecture models can continue to capture CONOPS of subsystem levels (if there are any) and the data flow from the component system-level architecture. In this architecture top down development sequence the layered architecture models are developed and shown in Figure 1. In a layered model, the overall SoS is broken down into different collections of services, with each collection expressing the services that are available to layers above it in the "protocol stack" (USAFSAB 2005). Layered architectures allow different developers to work in parallel and insure that changes in one layer of the protocol do not interfere with operations above and below that layer. Thus, layered architectures implement loose coupling between the services that makes up the overall SoS. System design including hardware and software will be based on architecture models in different levels.

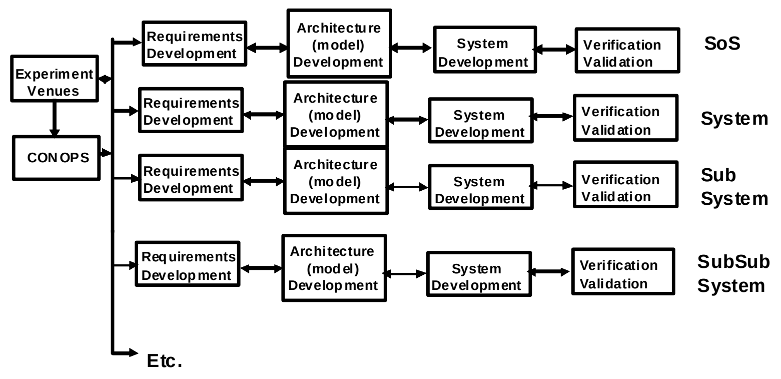


Figure 1. Layered Architectures of System-of-Systems

The SysML (Hsu 2006) is the latest modeling language for systems engineers to be applicable for both hardware and software. It is still in developing phase and not designed to capture emergent behaviors. Agent-based modeling technique has been applied in biological studies (Johnson 2002) and (Lodding 2004), human systems (Gore 2002) and (Seel 2003) and software development (Fisher 2006) and (Mogul 2005). It is especially in social simulation of human behavior by the North American Association for Computational Social and Organizational Sciences (NAACSOS), the European Social Simulation Association (ESSA) and the Pacific Asian Association for Agent-Based Approach in Social Systems Science (PAAA), etc. It should be attempted to use this modeling technique for the development of a SoS architecture.

Agent-Based Modeling

What is an agent? In a simplified way of explanation (Holland 1995), it is useful to think of an agent's behavior as determined by a collection of rules. Stimulus-Response rules are typical and simple. IF stimulus s occurs, THEN give a response r, for example, IF the car has a flat tire THEN get the jack. Stimulus is what the agent can receive and Response is what the agent can give. Each component system will be represented by an agent. Agent-based models consist of dynamically interacting simple rules based agents. The systems within which they interact can create real world-like complexity resulting in far more complex and interesting behaviors. Proactive decisions based on reinforcement learning are also possible agent behavior.

OpEMCSS (Operational Evaluation Modeling for Context Sensitive System) (Clymer 2007) simulates multi-agent systems that learn (Classifier Systems) and can perform fuzzy control. An agent is referred to sometimes as an "intelligent" agent or autonomous entity. It can be a physical entity that moves and acts in a real world environment or an abstract one that exists entirely in a computer. An agent context diagram is shown in Figure 2. For a physical entity it can be a subsystem or component of a larger system that interacts with other systems or subsystems, which results in emergent behavior. Agents can be hardware-based, software-based, process-based, people-based, or any multitude of entities that exhibit complex adaptive behavior. OpEMCSS makes possible the modeling of complex adaptive systems that have context sensitive system interactions.

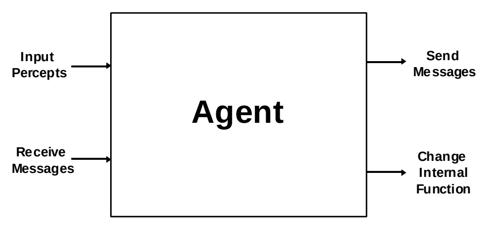


Figure 2. Agent Context Diagram

Modeling of multi-agent systems, is done through the use of special blocks in OpEMCSS, such as the Classifier Event Action block. This block operates in a rule learning mode, enabling a simulation that generates decision contexts which work to eliminate rules through learning, or rule induction. This enables expert learning in the transformation of a non-linear function.

Modeling the Emergent Behavior

There are four Principles of Emergence introduced by (Hsu 2007). OpEMCSS will be used to simulate and study these four principles. A perfect OpEMCSS example is the World Model. It is comprised of many agents with random initial placement and vectors. The World simulation uses agents that start to move randomly (in a fixed direction and velocity) in a 400x400 two dimensional (playing) field until they come in proximity with each other. The agents obey one basic rule. When the agents come together (within a certain range of each other), one of them will adopt the direction and velocity of motion of the other agent. The net effect is that the two agents will move together when they come into contact. Because the agents are moving randomly little groups of agents begin to form. The emergent quality of the system is that eventually the little groups combine to move as a large group in the same direction. As the agents collaborate and interact, they work towards a common vector or goal.

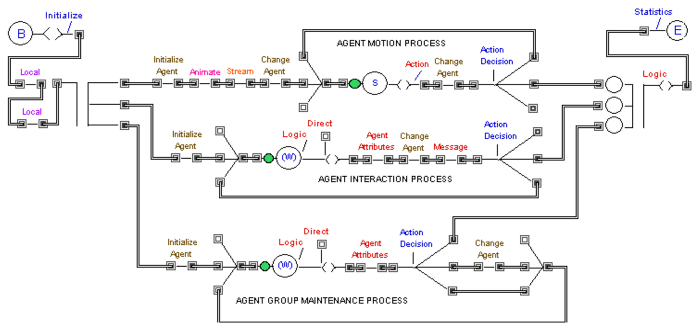


Figure 3. OpEMCSS Agent-Based World Model

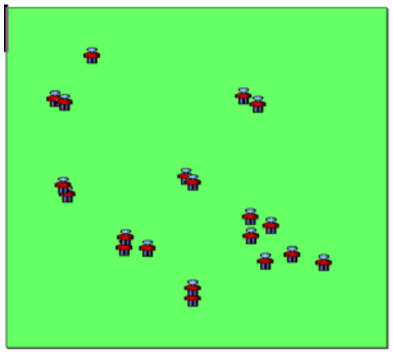


Figure 4. The Beginning of Agents Movement

The Agent-Based World Model of the interaction of autonomous, adaptive agents, interacting and collaborating characteristic to natural and human systems, is shown in Figure 3. For example, the twenty (20) agents would start move randomly in their own directions at the beginning of simulation as shown in Figure 4. When the twenty (20) agents reached emergent behavior, they moved together in any directions as shown in Figure 5.

IntRing is the parameter in the model to set the interactive range between agents. This interactive range can be interpreted as a measurement of bondage between agents. It was set at 15 for the simulation as shown in Figure 6. The simulation started with two agents since this was the minimum number of agents (systems) required to exhibit the emergent behavior under a SoS environment. Setting the number of agents was shown in Figure 7. The simulation runs started from two (2) agents and increased to twenty (20) agents. For each simulation run, the time for agents to reach emergent behavior was recorded. The time to emergence versus number of agents was shown in Figure 8. Please note that the time shown in the figure was the simulation run time and not the real time. But it can still be used as a relative measurement of time to emergence. It can be seen that the time to reach emergence is irregular with the number of agents in the play. The same simulation was repeated and the results were shown in Figure 9. Although the time to reach emergence was still irregular, the time taken to reach emergence was different for the same number of agents from that shown in Figure 8. The irregularity and different results for repeated simulation runs will be explained in the Discussion Section.

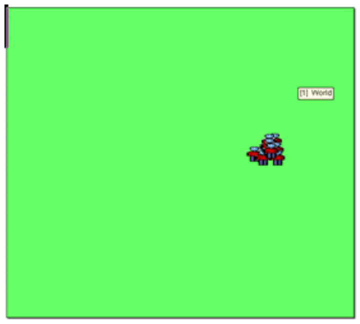
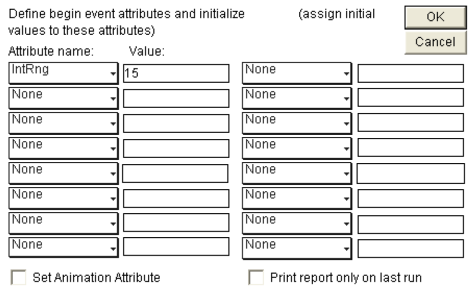


Figure 5. The Agents Moving together



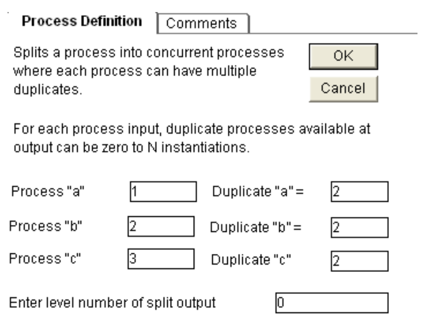
Define begin event attributes and initialize (assign initial values to these attributes)

Attribute name:	Value:		
IntRng	15	None	
None		None	
None		None	
None		None	
None		None	
None		None	
None		None	
None		None	

☐ Set Animation Attribute ☐ Print report only on last run

Figure 6. Define Begin Event Attributes

As mentioned in the above, IntRing is simulating the range within which the agents will be interactive. The next simulation would vary the IntRing for a fixed number of agents. The number of agent was set at ten (10) using the diagram shown in Figure 7. Changing the IntRing would use the diagram shown in Figure 6. The purpose of this simulation was to find the time to reach emergence for ten (10) agents as a function of bondage between agents. The results were shown in Figure 10. The bondage values shown in the figure was a measurement of relative binding force and not on absolute force. The resulting values may be different if the run was repeated due to the random autonomous characteristics of agents but the emergent behavior as a function of bondage should be the same, i.e., the time to reach emergence would be increased when the binding force between agents was increased.



Splits a process into concurrent processes where each process can have multiple duplicates.

For each process input, duplicate processes available at output can be zero to N instantiations.

Process "a"	1	Duplicate "a" =	2
Process "b"	2	Duplicate "b" =	2
Process "c"	3	Duplicate "c"	2

Enter level number of split output: 0

Figure 7. Set Agent Numbers

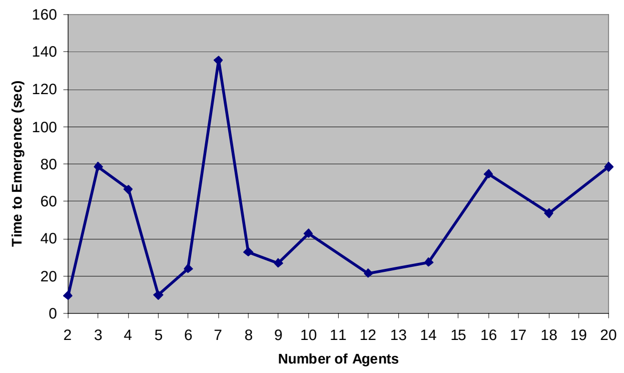


Figure 8. Time to Emergence vs Number of Agents

Discussion

The results from the above simulations will be discussed to demonstrate the compliance with the four Principles of Emergence.

1. **Condition of Emergence -** Emergence will not occur for a single independent system. An avalanche condition (Goldstein 1999), or a critical state, has to exist prior to the occurrence of emergence for a SoS environment.

   The number of agents has to be more than one (1) to satisfy the condition of emergence. Even though there is more than one agent, it takes a time delay to reach emergent behavior as shown in Figures 8 and 9. The time which takes to reach the emergent behavior is independent of the number of agents. It is governed by the agent's characteristics of autonomous and random movement.

2. **Emergent Behavior is Inversely Proportional to the Degree of Bondage between Systems** – The more tightly the component systems are coupled the less likely that the global emergent behavior will prevail.

   Figure 10 demonstrated that the time to reach the emergent behavior was longer when the binding force between agents was increased. The bondage between agents was a relative measurement determined by the interaction range between agents. The shorter the range would be the tighter the bondage between agents.

The tighter coupling would not allow agents to move freely simulated by an environment of shorter range of interaction that would be more difficult for agents to interact, vice versa, a loose coupling simulated by a longer interaction range that would be easier for agents to interact. It should be noted that a longer time-to-emergence implies a slower prevailing emergent behavior for tighter coupling and a less time-to-emergence implies a quicker occurrence of emergent behavior for loose coupling. It is interesting to note that the emergent behavior will not be further prevailed below a binding measurement of fifteen (15) units as shown in Figure 10. It may infer that beyond a certain level of loose coupling, the emergent behavior has prevailed to its maximum level.

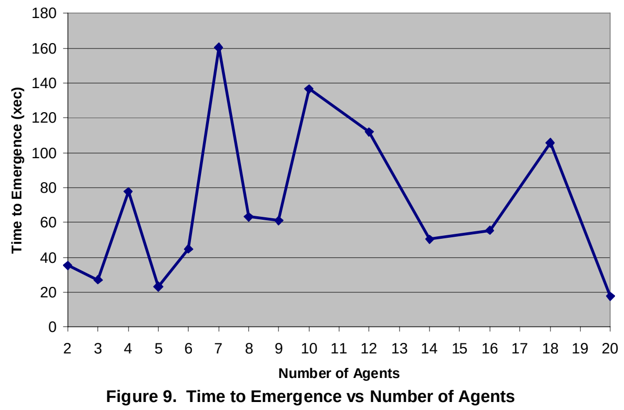


Figure 9. Time to Emergence vs Number of Agents

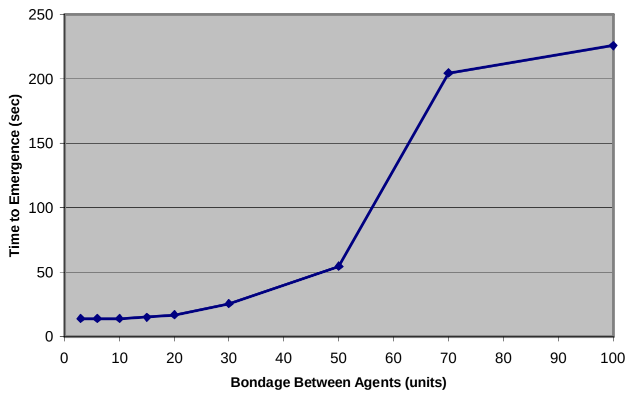


Figure 10. Time to Emergence vs Bondage Between Agents

3. **Emergent Behavior is Non-linear** - The influence by emergent behavior is not scalable (Sunami 2004). The output is not proportional to the inputs. It is related to the Metcalf Law due to the iteration and feedback characteristics, which can amplify effects at increasing or decreasing levels of scale.

   It was demonstrated in Figures 8 and 9 that time-to-emergent behavior was not in a linear relationship with the number of agents increased or decreased; instead, it exhibited non-linear random behavior. As it can be seen in Figures 8 and 9, the two simulation runs did not have the same results for the same number of agents. This behavior is not predictable and will contribute to the complicacy in estimating life cycle cost, analyzing risk, assessing safety and reliability, and verifying and validating the SoS.

4. **Emergent behavior is Self-organized** - Emergence begins at the ground level. They are each "self-organizing" systems (Johnson 2002), which typically display emergent properties. Self-organization (Dabrowski 2006) is a process in which the internal organization of a system, normally an open system, increases in complexity without being guided or managed by an outside source.

The inconsistent and random emergent behaviors observed in Figures 8 and 9 was the evidence of self-organized nature of agents and a bottom-up process. The agents interact randomly and autonomously per simple rules without any external force. The process involves constant communication and feedback among the lowest level of organization, pattern recognition, local action affecting global behavior, and takes into consideration the element of unpredictability in a chaotic system.

These four principles are interrelated. When the **Condition of Emergence** (# 1) will emerge governed by the **Self-Organized Emergent Behavior** (# 4)? Hence, the **Condition of Emergence** is random and unpredictable? The **Non-Linear Emergent Behavior** (# 3) is the result of **Self-Organized Emergent Behavior** (# 4) depending on how the simple rules are autonomously and randomly played among agents. The trend of **Emergent Behavior Inversely Proportional to the Degree of Bondage between Agents** (#2) is true but the exhibition of emergent behavior will be different each time due to the **Self-Organized Emergent Behavior** (# 4) and **Non-Linear Emergent Behavior** (# 3).

It is noteworthy to discuss risk assessment and mitigation on the SoS level. Due to the nature of emergent behavior, the summation of component system-level risk may be more or less than the sum of the individual system-level risk. If the component system risk level is low, the summation of the individual system risk onto SoS level may be high. On the other hand, if the individual system risk is high but the combination of these systems' risks could be low. The simulated conditions in Figures 8 and 9 showed that by adding or reducing the number of agents (systems) the emergent behavior could occur earlier or later (amplified at increasing or decreasing level). The simulated run in Figure 10 implied that the degree of bondage between agents after formation of a SoS level from the individual systems (agents) could also change the combined risk level. Summation of component system reliability onto a SoS level reliability has the same analogy.

The modeling efforts for emergent behavior presented in the above is conceptual and only at the beginning stage for this new endeavor to integrate emergent behavior into architecture models. It can be seen that there are many possible emergent conditions. Flexible and adaptable open system architecture is very important in dealing with these emergent behaviors. Most of the times, the emergent behavior would show up during the operations of SoS. The architecture models should be robust and easily modified to incorporate the newly discovered emergent behaviors.

Sometimes, a system with a rather large level of complexity, such as a very large number of interconnections and rules, that also has some level of self-organization, self-modification, or such, will suddenly start to exhibit behaviors that were totally unpredicted by those people who designed and built the system. The result of emergent behavior may involve influence, cooperation, cascade effects, distributed control, and orchestration. Neural networks may be helpful in modeling the emergent behavior since the emergent behavior is complex and sometimes self-initiated. Neural network can provide an on-going learning capability to capture these emergent behaviors. It is suggested the agent-based modeling integrated with neural network methodology and SysML. It may be necessary to have a separate group of specialists in agent-based modeling and neural network working with the modelers specialized in SysML to develop architecture models jointly.

Conclusion

From the demonstrated agent-based simulations, the emergence principles have SoS-level influence on:

- Top down development via functional decomposition
- System-of-System specification structure & contents
- Functional and performance requirements flow down
- Requirements traceability
- Requirements validation and verification

Emergence can be beneficial or harmful. They exist in complex systems or SoS. Emergence is the primary mechanism for both success and failure in SoS. A need exists for developing accurate architecture models including emergent behavior for a SoS. The challenge is how to develop emergent behaviors that can be incorporated in architectural models. Emergence is not the enemy. We can learn how to use it to control the emergent behavior.

Simple and conceptual agent-based models were presented in this paper to simulate the four principles of emergent behaviors. The unpredicted emergent behavior may prevail during SoS operation. At that point the architecture has already been developed. The SoS is in a continuous learning mode. The neural network artificial intelligence may be the answer for this situation if it is incorporated into the architecture model. It is recommended the agent-based simulation integrated with SysML.

References

Clymer, John R., *OpEMCSS: Exploring the Intricacies of Simulation for MBSE,* John R. Clymer & Associates, 2007.

Dabrowski, Chris and Mills, Kevin, *A Program of Work for Understanding Emergent Behavior in Global Grid Systems,* National Institute of Standards and Technology, February 13, 2006.

Fisher, David, *An Emergent Perspective on Interoperation in Systems of Systems,* (CMU/SEI-2006-TR-003), Pittsburgh, PA, Software Engineering Institute, Carnegie Mellon University, March 2006.

Goldstein, Jeffrey, *Emergence as a Construct*: *History and Issues*, Emergence 1, 1999.

Gore, B. F., *An Emergent Behavior Model of Complex Human–System Performance: An Aviation Surface-Related Application***,** VDI-BERICHT NR. 1675, Dusseldorf, Germany, 2002

Holland, John H., *Hidden Order*, Helix, Addison-Wesley Company, Reading, Mass, 1995

Hsu, John C. and Butterfield, Marion, *Modeling Emergent Behavior for Systems-of-Systems*, 17th Annual INCOSE International Symposium, 2007.

Hsu, John C. and McDonough, J. Mike, *Applying the Object Oriented Systems Engineering Method to A Simple Hardware System, INCOSE Symposium*, Toulouse, France, 2004.

Hsu, John C., *Applying Systems Modeling Language to A Simple Hardware System,* INCOSE Symposium, Orlando, Florida, 2006.

Johnson, Steve, *Emergence: The Connected Lives of Ants, Brains, Cities, and Software*, Simon and Schuster, New York, 2002.

Lodding, Kenneth, *Hitchhiker's Guide to Biomorphic Software*, ACM Queue vol. 2, No. 4, June 2004.

Maier, M., *Architecting Principles for Systems-of-Systems*, Proceeding of the 6th Annual INCOSE Symposium, p. 567-574, 1996.

Mogul, Jeffrey C., *Emergent (Mis)behavior vs. Complex Software Systems*, Hewlett Packard Laboratories, HPL-2006-2, December 2005.

Seel, Richard, *Emergence in Organisations*, 2003.

Sunami, Christopher, *emergence: Less is More*, The Kitopedia, 2004.

United States Air Force Scientific Advisory Board, *System-of-Systems Engineering for Air Force Capability Development*, SAB-TR-05-04, July 2005.