Simulation-based Complex Adaptive Systems

Saurabh Mittal
The MITRE Corporation

José L. Risco-Martín
Complutense University of Madrid

Abstract

Complex Adaptive Systems (CAS) are systems that display two primary characteristics: emergent behavior and adaptive behavior. Emergent behavior manifests in a system comprising of large number of components engaged in multi-level interactions. These components are also considered as agents in a multi-agent system. Adaptive behavior manifests at the agent-environment boundary when the agent situates itself in an environment. This agent-environment interaction results in changing the agent's original behavior as it continues to adapt to the environmental conditions. Modeling a CAS has been a challenge due to limitations in bringing these two aspects together in a single formal specification in a computational environment. Lack of simulation environment for CAS model adds further problem in validation and verification of a CAS. Computationally, the emergent behavior can be understood using today's latest technology of Feature Engineering, Deep Learning and data analytics using Big Data. This would facilitate the identification of various holistic behaviors and their classification that would aid designing various algorithms for detecting the eventual emergent behavior. This aspect is largely bottom-up. Once various observers are computationally available, they can be integrated in the agent behavior repertoire so that the emergent behavior, that is now detectable and perceivable at the agent-environment boundary, can be used and acted-upon by the agent. This situates the agent in the environment and manifests as adaptable behavior. This activity is top-down as there is a conscious design process (done by a human) that is employed for such behavior refinement. This chapter will discuss the state-of-the-art in computational and simulation support needed in CAS engineering and provides foundation to manifest accurate emergent behavior rather than emergent behavior arising from the underlying computational infrastructure.

Keywords

complexity, emergent behavior, complex adaptive systems, system of systems, co-simulation, multi-paradigm modeling, cyber physical systems, cyber complex adaptive systems, network science, adaptive, multi-agent systems, evolutionary computing, emergence complexity cone, hyperheuristics

1 Introduction

System complexity continues to grow by leaps and bounds. Multi-level complexity is a fundamental nature of heterogeneous systems today. To describe this new class of super complex systems in a man-made world, labels such as Collaborating Systems, System of Systems (SoS), Cyber Physical Systems (CPS) (E. Lee, 2008), Complex Adaptive Systems (CAS), Cyber CAS (CyCAS) (Mittal, 2013b, 2014) are used interchangeably. All of them are multi-agent systems i.e. they are all contextualized in the environment, interact among themselves and with the environment, are goal-oriented with incomplete information at any given moment and manifest emergent behavior. SoS may incorporate large scale systems, as SoS components, under independent operational and managerial control. CAS are SoSs where constituent systems can be construed as agents that interact and adapt to the dynamic environment. Cyber CAS is a CAS that exist in a netcentric environment (for example, Internet) incorporating human elements where distributed communication between the systems and various elements is facilitated by agreed upon standards and protocols. CPS are SoSs wherein their physical and embedded systems are remotely controlled through their cyber components. In modern times, the Internet of Things (IoT) is beginning to incorporate all of these characteristic and is becoming a significant contributor to the increase in complexity. However, the IoT phenomenon is still in the formative stages of an apparently exponential growth. Designing these systems is equally complex and methodologies available through traditional systems engineering practices fall short of engineering these super complex systems.

Complex adaptive systems is not a new concept. Fifty years ago, in his book, Christopher Alexander (1964) drew the following context:

- "Today more and more design problems are reaching insoluble levels of complexity"
- "At the same time that problems increase in quantity, complexity and difficulty, they also change faster than before"
- "Trial-and-error design is an admirable method. But it is just real world trial and error which we are trying to replace by a symbolic method. Because trial and error is too expensive and too slow"

This scenario continues to be applicable even today, and probably will be fifty years from now as well. The sheer lack of methodologies to do CAS engineering in a heterogeneous environment makes it a fairly hard problem. Validation and verification for CAS engineering is being defined (Mittal & Zeigler, 2017). The use of real-world data and of artificial data to test a hypothetical CAS is still being characterized. Not only that, the properties of CAS do not belong to a single domain of interest but is cross-domain and multi-disciplinary. Approaches that leverage data analytics, speed iterations, built in agility and ensure a holistic view for robust decision-making that are essential to solve complex engineering programs are in the process of being incorporated in the CAS engineering process. Consequently, engineering CAS elements as well the processes to facilitate

CAS engineering are being worked out. Due to the complexity of such systems, these systems exhibit emergent behavior. Efforts are underway to harness emergent behavior as an essential component of the CAS engineering process (Mittal & Rainey, 2015).

Both modeling and simulation has supported the systems engineering process across all its phases. Paradigms like Model Based System Engineering (MBSE), Model Based Development (MBD), Model Based Engineering (MBE) is often used to describe the usage of modeling practices in systems and software engineering. Many times modeling is performed without the simulation. In that role, modeling acts as a validation mechanism to bring a common understanding to all the stakeholders that include the end-users (who use the system), the engineers (who make the system) and the business holders (who invest in the system). These validated models serve as architectural blueprint. Such models are usually static in nature and the verification or correctness of these models rests with the modeling workbench used to create these models. The metamodels implemented in the modeling editors are the foundation on which the modeling representation rests. To evaluate the dynamic behavior of these models, simulation is warranted.

With CAS, as both the model and the data it is dealing with, are multi-disciplinary, the CAS solution is constantly evolving due to the sheer richness of stakeholder interest and varied disciplines. Without a failsafe validation process, the CAS model can fall out-of-sync quickly rendering the model useless. Much of the model validation depends on the data validation, which depends on the data exchanged between the actual system and the environment compared with the data exchanged between the model and the environment. Data-driven methodologies need to scale up in a multi-disciplinary environment as well. Assuming that the modeling workbenches can support agile modeling processes, the simulation infrastructure needed to validate the dynamical behavior should be made agile as well.

Emergent behavior, a macro behavior that is irreducible at the micro level, is a characteristic property of any natural CAS. Reproducing that in an artificial environment, such as through M&S endeavor is a non-trivial problem. There are two fundamental reasons. By the definition of emergent behavior by Ashby (1956), the source of emergent behavior of a system is an incomplete understanding of the system. So, in order to model a CAS, do we have complete understanding of CAS, to begin with? The answer is clearly no because, emergent behavior is a by-product of all the structural and behavioral richness in a natural CAS. This emergent behavior results in a dynamic environment and dynamic agent behavior that adapts to it and eventually learns to survive in that dynamic environment. In a modeling endeavor, much of the low-level detail is deliberately ignored to manage the model's complexity. Abstraction results in loss of information at a finer level. Regardless, a model can be made to produce the same emergent behavior with the support of a hypothesis or theoretical constructs. To check the hypothesis, the model must be simulated to ensure correct dynamical behavior. This leads us to the second reason. Developing a multi-agent co-simulation platform in a distributed environment is also a non-trivial

challenge and out-of-order events and computational complexity of the simulator execution introduces unintended behavior that eventually contributes to inaccurate emergent behavior. Validation and verification of agent-based modeling environments is an active area of research (Yilmaz, 2006; Arifin & Madey, 2015).

This chapter discusses the complexity inherent in CAS modeling, the multi-faceted data-driven methodology and the supporting simulation infrastructure using a co-simulation methodology. Having a robust co-simulation infrastructure is a must have to eliminate unintended emergent behaviors arising out of computational simulation environment (Mittal & Zeigler, 2017). Only ensuring that will keep the focus of CAS engineers towards modeling the desired and accurate emergent behaviors.

The chapter is structured as follows. Section 2 discusses the complexity in multi-disciplinary CAS engineering. Section 3 discusses the adaptive aspect in CAS engineering. The nature of emergent behavior, its taxonomy and how it is reproduced in a modeling and simulation environment is discussed in Section 4. The data aspect supporting model evolution is discussed in Section 5. The co-simulation architecture for simulation-based CAS engineering is described in Section 6, followed by conclusions and future work in Section 7.

2 Complexity in CAS Modeling

A basic definition of CAS was provided by Holland (1992):

> CAS are systems that have a large number of components, often called agents that interact and adapt or learn

More specific definitions appreciating the associated complexity started appearing after a decade later. The earliest definition we could find is provided in (Plsek & Greenhalgh, 2001):

> A complex adaptive system is a collection of individual agents with freedom to act in ways that are not always totally predictable, and whose actions are interconnected so that one agent's actions change the context for other agents.

Miller and Page (2009) discussed the subject of CAS and the difference between a complicated and complex system, and how computational models can help describe a complicated system but hit a boundary when it comes to complex systems with adaptive behavior. While they describe the CAS components in an illustrative and computational manner, they did not actually provide a definition of CAS. A precise definition of an emerging concept is an important undertaking, as it will define what it "is" and "is not". This also hits at the heart of how M&S can support the concept definition activity and thereby, advance science. Indeed, if a concept is expressed precisely, it will be reflected accordingly in the created model, and can be validated. Failure to do so will introduce ambiguity and uncertainty.

System modeling begins with understanding the structure and behavior of a system. Once sufficient understanding is available, model-based design and analyses can proceed. To understand the structure and behavior of CAS, thereby to perform MBSE for CAS, Mittal (2013b) stated:

> A CAS is a complex, scale-free collectivity of interacting adaptive agents, characterized by high degree of adaptive capacity, giving them resilience in the face of perturbation. Indeed, designing an artificial CAS requires formal attention to these specific features. Complexity is a phenomenon that is multivariable and multi-dimensional in a space-time continuum.

From the structural perspective, a multi-agent system is analogous to a system of system (SoS). The connectedness of each of these systems is well researched in the works of Barabasi (2003) and Newman (2001). Mittal (2013b) summarized the structural scale-free characteristics of CAS, as enumerated in Table 1.

If the problem was to just address the scale-free nature with homogeneous agents/systems, there are ample tools that can model a scale-free structure. However, any real world CAS is a heterogeneous system with multiple characteristics. Modeling such a heterogeneous system requires agent characteristics dependent on the particular source of behavior for that agent, which may be domain dependent. The complexity increases when the model has multi-level definitions, i.e. a containment hierarchy exists. The problem further grows exponentially if the agent/system, even though available as a modular component, is modeled in a dissimilar paradigm, such as discrete, continuous or hybrid. Undoubtedly, in such a situation, the presence of emergent behavior is but natural.

The primary question is:

> Do we know enough about the system to guide the emergent behavior and ensure that it is the right one, to begin with?

Mittal and Zeigler (2017) stated that multi-paradigm modeling is the preferred means to bring in domain knowledge from multiple disciplines to develop a CAS model. Some of the disciplines they enumerated: Cognitive Psychology, Network Science, Human Factors, Communication Systems, Ontology, Complexity Science, Supply-Chain, Power Systems, etc.

Each of the disciplines contribute towards defining the behavior of an agent/system and provides constraints to control the structure of the resulting CAS. The agent/system behavior needs some further consideration. An agent may have variable degrees of autonomy and a system may be a passive system with limited autonomy. From the behavioral perspective, CAS are systems that display two primary behaviors: adaptive behavior and emergent behavior. Let us look at the modeling of these two behavioral aspects.

Table 1: Scale-free characteristics of complex adaptive systems

ID	Feature	Description
1	Incremental Growth	Incremental linking to other agents/systems in a persistent environment (an environment that has a spaciotemporal nature)
2	Self-organization	Agents/systems may organize themselves into clusters, groups for a common objective
3	Critical state transition	At appropriate time during network growth, the system displays fundamentally different behavior that may be termed emergent behavior
4	Emergence of hubs and clusters	Networks starts displaying small-world effects and a network hierarchy emerges
5	Power-law behavior	Some agent/systems become network enables and facilitate "long" weak ties that sustain large scale topologies
6	Non-linear interactions	System self-organized through these hubs result in emergent interactions that brings new affordances and/or constraints
7	Preferential attachment	Agents/systems exhibit affinity for other agent/systems that changes the inherent structure itself
8	Vulnerability to attacks if targeted to hubs	Any attack on the hubs results in cascaded effects and aligns itself towards self-organized criticality when it reaches a critical state
9	Threshold levels	Each agent/system has a threshold model that controls its preferential attachment
10	Concurrency and multi-tasking	Each agent/systems is modular and has defined interfaces and capabilities that are operational in a concurrent manner.

3 Adaptive aspect in CAS Modeling

Adaptive behavior manifests at the agent-environment boundary when the agent situates itself in an environment. This agent-environment interaction results in changing the agent's original behavior as it continues to adapt to the environmental conditions.

An *adaptive system* is a system that changes in the face of perturbations so as to maintain some kind of invariant state by altering its properties or modifying its environment. The ability to adapt depends on the observer who chooses the scale and granularity of description. An adaptive system is necessarily complex, but the opposite is not necessarily true.

Evolution is a result of an adaptive system. In fact, John Henry Holland (1992) was one of the biggest contributors that led to the inception of the

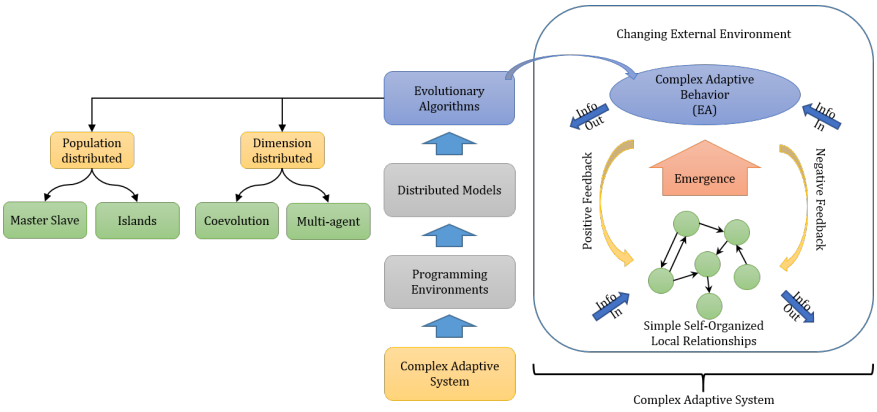

Figure 1: Evolutionary algorithms in CAS modeling

field from the perspective of adaptation. He introduced the concept of genetic evolution in describing adaptive systems. He was interested in the question of how computers could be programmed so that problem solving capabilities are built by specifying *what to be done*, instead of *how to do it* (Brownlee, 2007). Holland conceptualized an adaptive plan, which was the continuous modification of structures by means of genetic operators. The specialization of this adaptive plan, called the genetic plan, represented at the end, laid the foundation of the field of genetic algorithms and, more generally, evolutionary computation.

The design of an adaptive system that presents emergent behavior becomes a complex and challenging task (Branke & Schmeck, 2008). It is impossible to know how design choices made at a component level affect the overall system behavior. Thus, these choices must be taken by means of computationally extensive models and the corresponding simulations. As stated before, adaptability is based on evolution. Thus, the adaptive behavior of these models can be tackled using evolutionary computation. Figure 1 depicts an illustrative example of how the CAS behavior can be modeled through evolutionary computation. Some characteristics that we may find in Evolutionary Algorithms (EAs) make them excellent candidates for dealing with M&S of CASs (Alba & Cotta, 2002; Branke, 2001; Deb, 2001; Arnaldo et al., 2013).

First, EAs act as a black box in a optimization heuristic (the partial search algorithm). There are no restrictions on the fitness function. Thus, these algorithms can be used in complex simulation environments.

Second, the fitness evaluation can often be simplified by means of the reduction of the computational complexity. This fact allows us a simplification of the initial complex environment.

Third, EAs behave well with uncertainty in evaluation, are able to manage adaptation to changing or adaptive environments, as well as in real time scenarios. In this regard, EAs are considered adaptive and *interactive*. This last feature makes them useful in the study or design of desired emergent behavior.

Fourth, there exist a group of EAs called Multi-Objective Evolutionary Algorithms (MOEAs). These range of methods are designed to deal with multiple objectives or fitness functions. This is a realistic method to quantify emergent behavior that looks for different alternatives, depending on the chosen objective in a given situation. Using relations like Pareto dominance or Nash ascendancy these algorithms can offer a good spectrum of different solutions to the problem under consideration (Deb, Mohan, & Mishra, 2005).

Finally, from a performance perspective, EAs are easily parallelizable, even on heterogeneous hardware platforms or in cloud environments. Being based on the evolution of a population, the repetitive evaluation of each individual in the population can be distributed using a master-worker or an islands-based distribution. Coevolution and multi-agent systems can also be used in the distribution of EAs, as Figure 1 shows.

There are many examples in the literature of modeling complex adaptive systems and inducing desired emergent behavior. To name but a few, in (Chellapilla & Fogel, 2001), Chellapilla and Fogel used a genetic algorithm to evolve neural networks able to play the game of checkers. The major achievement of this work is based on the competitive strategy, which is evolved given only the spatial positions of pieces on the board and the piece differential, a decision approach that would normally require of human input and expertise. (Andre & Teller, 1999) successfully applied evolutionary methods to develop a program to control a team of robot soccer players. The evolutionary algorithm operated with a set of basic control functions such as turning, running and kicking. The fitness function was global, evaluating a general good play. There was no score of specific tasks like tracing the ball, kicking in the correct direction, goal scoring, etc. The robot team, named Darwin United, entered in the international RoboCup annual tournament and competed with other teams of autonomous robots (*RoboCup*, 2017). Darwin United outranked half of the human written, highly specialized programmed teams, performing quite well.

4 Emergent Behavior aspect in CAS modeling

Emergence has been a native of the land of "complex systems" and there are four schools of thought that study emergence, as summarized by Wolf and Holvoet (Wolf & Holvoet, 2005):

- Complex adaptive systems theory: Concept of macrolevel patterns arising from interacting agents.
- Non-linear Dynamical Systems theory and Chaos theory: Concept of attractors that guide the system behavior.
- The synergistic school: Concept of order parameter that influences which macro-level phenomena a system exhibits.
- Far-from-equilibrium thermodynamics: Concept of dissipative structures and dynamical systems arising from far-from-equilibrium conditions.

A new field of application of emergence is System of Systems (SoS) (Maier, 1998; Mittal & Rainey, 2015) and complex sociotechnical systems (Mittal, 2014). Modeling and simulation support to SoS engineering (Rainey & Tolk, 2015) discusses four types of emergence, extended from the works of Maier (1998):

- Simple emergence: The emergent property or behavior is predictable by simplified models of system's components.
- Weak emergence: The emergent property is readily and consistently reproduced in simulation of the system but not in reduced complexity non-simulation models of the system i.e. simulation is necessary to reproduce it.
- Strong emergence: The emergent property is consistent with the known properties but even in simulation, is inconsistently reproduced without any justification of it's manifestation.
- Spooky emergence: The emergent property is inconsistent with the known properties of the system and is impossible to reproduce in a simulation of a model of equal complexity as the real system.

Works by Mittal (2013a) and Szabo & Teo (2015) give enough argument that weak emergent properties can be formally specified using mathematical principles and reproduced in a simulation environment. Strong emergent behavior is bounded by the knowledge of the existing properties of the system. Consequently, new knowledge is always generated when strong emergent properties are known for the first time (Mittal, 2013a). When they are known during a simulation study by stochastic methods (Mittal & Rainey, 2015), and are validated as a new property that is consistent with the known properties by the system's Subject Matter Experts (SMEs), they are then incorporated as "new behavior" that is not "emergent" anymore and can be now formalized as weak emergent behavior. This can now be consistently reproduced in a simulation.

Stochastic studies and uncertainty quantification lie in the simulation domain and are supported by Estimation Theory. These computational methods used in conjunction with the existing theories hep validate the existing theories or suggest modifications to them. These also help defining the boundaries of the theory, in turn, giving us specific knowledge about the constraints and limitations of that theory which must be implemented for taming the emergent behavior. Figure 2 shows Emergence Complexity Cone linking emergent behavior taxonomy in increasing complexity on the y-axis, division of Cone into stochastic and deterministic search-spaces on the x-axis, with Cone volume depicting the variety (Ashby, 1956), Cone perimeter as constraints, and the knowledge boundary as a cylinder that addresses the variety and constraints. Knowledge cylinder around simple and weak emergence in the deterministic domain signifies ample knowledge available to develop abstractions. A diverging cone reflects the increasing complexity as constraints are loosened in the stochastic domain leading to an increase in variety and lack of theoretical constructs to understand the overall complex behavior.

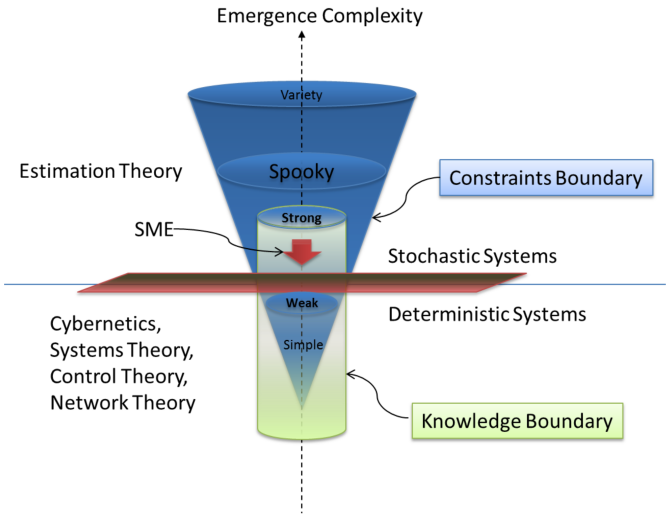

Figure 2: Emergence Complexity Cone (Mittal & Rainey, 2015)

Further, computationally, the emergent behavior can be understood using today's latest technology of Feature Engineering, Deep Learning and data analytics using Big Data and Machine Learning (Tolk, 2015). This would facilitate the identification of various holistic behaviors and their classification that would aid designing various algorithms for detecting the eventual emergent behavior that may be useful in SoS context. This aspect is largely bottom-up. Once various observers are computationally made available, they can be integrated in the agent behavior repertoire so that the emergent behavior, that is now detectable and perceivable at the agent-environment boundary through various newly developed sensors (both software and hardware), can be used and acted-upon by the agent. This process refines the agent behavior with respect to the environment it is in, thereby situating it in the environment, and hence manifesting adaptable behavior. This activity now becomes top-down as there is a conscious design process (done by a human) that is employed for such behavior refinement and incorporating of new emergent behavior as a causal behavior impacting the existing behavior of the agent. The process thus becomes increasingly cyclical, wherein simulation studies are constantly discovering new consistent emergent behaviors that are then continuously added to the agent/system behavior repertoire.

We shall now see how data is an integral part of simulation-based CAS engineering process.

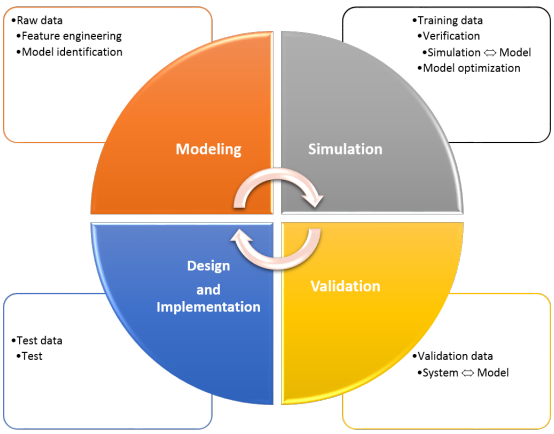

Figure 3: Data-driven M&S in CAS

5 Data-driven analytics for CAS Model evolution

Current complex systems are very dynamic in nature and usually present failures in real time that the controller must be able to repair. Given an IoT system for example, we may observe how sensors are continuously failing, but the complex system as a whole must be able to adapt itself to the new situation. In these new kind of problems, a global and adaptive modeling and optimization strategy can be especially useful. These systems are highly complex and cannot be easily tackled using classic modeling, simulation, and optimization techniques. They demand a heterogeneous and multi-level approach. For this purpose, the M&S work starts with a precise knowledge of the problem, developing techniques and methodologies that begin with data acquisition in real environments, and observing the practical constraints of the given scenario.

Figure 3 shows an overview of the data-driven methodology. With an initial raw dataset, a Model Based System Engineering (MBSE) process starts. To this end, the minimal set of features used to build the model is firstly selected. This model is then simulated for verification and optimization. Finally, the model is validated and tested and the actual system is implemented (Zeigler et al., 2000).

One of the major problems in the methodology detailed above is the constant necessity of human experts to take certain decisions in the modeling part, mainly because of the unpredictability of the emergent behavior in CAS. To tackle this issue, and as stated in Section 3, the model is usually complemented with an evolutionary computation module to emulate this emergent behavior. Some algorithms frequently used are Particle Swarm Optimization (PSO) (Zhang et

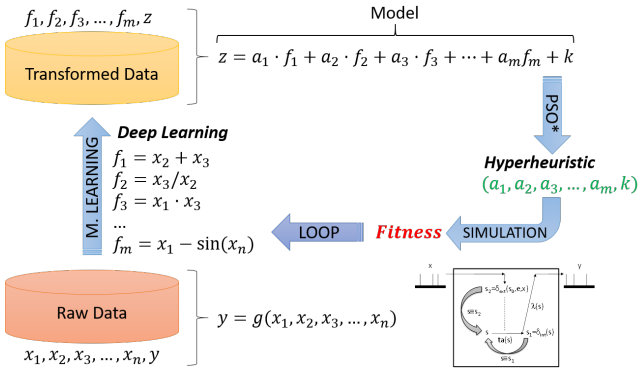

Figure 4: Specific view of the CAS modeling part

al., 2017) or Ant Colony Optimization (ACO) (Wang et al., 2016) that emulate emergent features to solve complex optimization problems. Evolutionary computation is able to reproduce complex adaptive models. However, this is not enough. Although the aforementioned models faithfully reproduce complex, dynamic and "alive" systems, the limitation is always found in the heuristic used by the corresponding meta-heuristic. So, the question is:

> Is there always the need for human experts in the process of modeling complex optimization heuristics?

The answer is no, we may find several approaches to turn these heuristics into hyperheuristics (Zheng et al., 2015). These methods try to transform the heuristic into a self- algorithm. This task is usually performed using machine-learning techniques. This technique is very useful because it is implemented under the assumption that once the model is integrated into the final system, it may also evolve. This evolution highly depends on the input data, as well as on the number of features. The set of features can also change during the whole model life cycle. Regardless of the machine-learning approach used (e.g., Feature engineering techniques, Deep Learning, pattern-matching, etc.), these techniques usually work in one of the following directions: generating new heuristics from basic sub-structures, automatically managing the search operators in the main evolutionary algorithm, or tuning the control parameters of the algorithm itself.

Figure 4 shows a specific view of the aforementioned CAS modeling part.

As the first step, it is required to have a representative data set of the system for which the model will be defined. Next, using machine learning algorithms (like Deep Learning) and in combination with the hyperheuristic approach, the set of features or input variables that are representative for the model are obtained, as well as the model itself, which is a function of these features. Feature engineering is a particularly useful technique to select an optimal set of features that best describe a complex model. Those features consist of measurable properties or explanatory variables of a phenomenon (Arroba et al., 2014).

Secondly, the first version of the model is simulated. The system's behavior is studied, simultaneously verifying and validating the reliability of the system both in virtual and in real time environments. Often, the provided model or even the modeled system require an optimization cycle. Minimization of risks, cost, as well as maximization of performance, are some examples. A varied set of optimization techniques can be applied for this purpose: MILP (Mixed Integer Linear Programming), Simulated Annealing, Genetic Algorithms, Particle Swarm Optimization, Genetic Programming, Multi-Objective Optimization, etc. The model obtained is more robust according to the performance of the machine learning and hyperheuristic phases.

Finally, as Figure 3 depicts, the system is implemented. This implementation is performed systematically. In hardware systems for example, hardware prototypes are firstly embedded into the software model. In a Hardware/Software co-simulation environment (Mittal et al., 2015; Risco-Martín et al., 2016), the target system is verified and validated using all the previous procedure and the general structure shown in Figure 3.

6 Co-Simulation environment for CAS

According to (Clymer, 2009), designing and evaluating a CAS is a three step process. First, a mathematical model is required to define and represent a CAS environment with precision. It describes the set of agent interactions that happen between the CAS agents during the system's operation. Second, a graphical modeling language that implements a set of equations for modeling dynamical behavior that enhances the visualization and understanding of CAS. Finally, an evaluation tool that ensures that the application of Systems Theory, expressed using the aforementioned graphical modeling language still holds. As we have discussed earlier, a CAS model is a multi-domain model. Consequently, constructs from multi-paradigm modeling should be adhered to.

A multi-paradigm modeling (Vangheluwe & Lara, 2003) effort aligns different paradigms with their corresponding modeling formalisms and implementation types towards a composite model capable of exchanging information across various abstraction levels.

Table 2 (Mittal & Zeigler, 2017) enumerates some of the tools that are currently used in a particular domain. Some tools are language dependent (e.g. C++, Java, LISP, DSL) and/or some are platform dependent (e.g. Windows, Linux, Mac). Some are discrete event, some are continuous and closed-form. These tools have their own software architecture, subscribe to a scientific theory and sometimes the software is proprietary. Figure 5 shows A, B, C, D, E as sample architectures. It also shows a layered M&S architecture addressing the pragmatic, semantic and syntactic levels of interoperability. For more details, see (Mittal et al., 2008; Mittal, 2014).

Developing a modeling workbench for multi-paradigm modeling environment is a non-trivial exercise and two solutions exist. Given a set of modeling formalisms that a CAS model needs, the first option requires the use of a Formal-

Table 2: Domains and their tools (not a complete list) (Mittal & Zeigler, 2017)

ID	Domain	Modeling Tool/Architecture
1.	Cognitive Psychology	ACT-R (2017), jACT-R (2017), SOAR (2017)
2.	Network Science	Gephi (2017), NetworkX (2017), Igraph (2017), Statnet (2017), Pajek (2017)
3.	Human Factors	JACK (2017), Kinemation (2017)
4.	Communication Systems	OpNet (2017), OmNet (2017), NS-3+ (2017), MATLAB/Simulink (2017)
5.	Ontology	Protégé (2017), TwoUse (2017), NeOn (2017), FlexViz (2017)
6.	Complexity Science	NetLogo (2017), RePast/Symphony (2017), DEVS (2000), R (2017)
7.	Supply-chain	MS Excel, Arena, SAS
8.	Power systems	GridLab-D (2017)

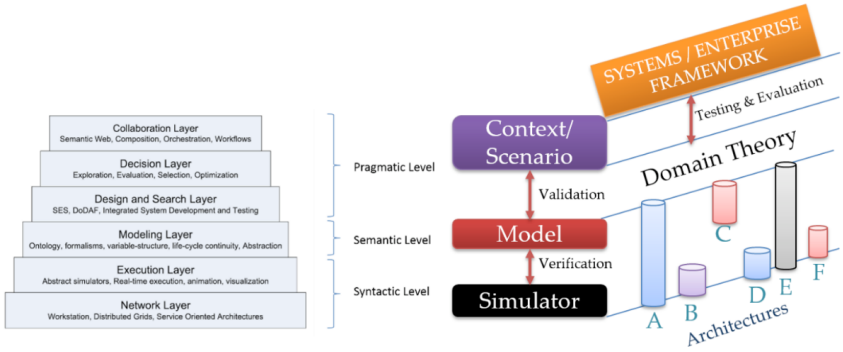

Figure 5: M&S with Verification and Validation and T&E (Mittal, 2014)

ization Transformation Graph (Vangheluwe, 2000) that transforms a relatively simpler formalism to be transformed into a more rigorous formalism. Such was an approach used by AtoM3 (Vangheluwe & Lara, 2003). However, one then has to understand each of the other formalisms and their semantics to ensure that there are no leaky abstractions and the mapping is correct. The other approach is to transform these modeling formalisms to hybrid super-modeling formalism that can model both discrete and continuous systems at the fundamental level, for example, DEV&DESS (Cellier, 1977; Praehofer, 1991; Zeigler et al., 2000; Mittal & Martin, 2013). This would ensure that the appropriate abstractions from each of the formalisms are integrated at the state-event and time-event levels and are simulated in a mathematically verifiable way, as implemented in the DEVS formalism (Zeigler et al., 2000).

Bringing various simulators together is more than a typical software engi-

neering integration exercise. The relationship between the modeling formalism and the underlying simulator is sacrosanct and unless is rigorously implemented, will, in all probability, yield emergent behaviors. The source of such emergent behavior then is not the lack of knowledge to model, but failure to implement the simulator, which now has become a multi-simulator. Without the DEVS super-formalism as a foundation (Mittal & Zeigler, 2017) that specifies an abstract simulation protocol between the model and the simulator (Zeigler et al., 2000), it would resemble engineering a multi-threaded software program with no verifiable inter-thread communication protocols to guarantee timeliness and accurate concurrent execution in an SoS setting. The task of integrating various simulators to perform together as a composite simulation is termed as co-simulation. This involves weaving the time series behavior and data exchanges accurately, failure of which, will yield inaccurate simulation results. Every such hybrid system would require a dedicated effort to build a co-simulation environment. Earlier work on DEVS-Bus (Kim & Kim, 1998), the netcentric SOA simulation infrastructure (Mittal & Martin, 2013) and recent work in building cyber-physical simulation environments (K. Lee et al., 2015), agent-based power-hardware-in-the-loop with simulation infrastructure at National Renewable Energy Lab, Dept. of Energy, USA (Mittal et al., 2015; Pratt et al., 2017) along with a multi-agent toolkit MECYSCO (Camus et al., 2017) provide a solid foundation to execute a hybrid discrete event complex continuous model in a parallel distributed co-simulation environment using a super-formalism such as DEVS formalism, producing accurate results. Earier efforts at Oak Ridge National Lab (Nutaro, Kuruganti, Shankar, Miller, & Mullen, 2008) and usage of agent-based co-simulation (Kilkki et al., 2014), both for Smart Grid M&S provide further evidence. The integrated model can be used for combined simulation of electrical, communication and control dynamics.

When the simulator code is open-source or available, the architecture of the tools could be understood and interventions can be made to weave the external tool's input. However, many times these simulation tools are proprietary architectures with no access to the code-base. In these cases, there is truly no means to verify the model's execution at the simulator level. At this point, the emergent behavior, which now is a function of computational engineering, cannot be overcome. In other words, even though model representation is valid, it cannot be verified computationally. Without a robust multi-simulation or co-simulation environment, advances in CAS will hit a ceiling and any progress on modeling will not achieve desired and reproducible results.

7 Verification and Validation of CAS Models

Complex adaptive systems: An introduction to computational models of social life*Validation* is the process of testing a model for validity. To validate, input and output trajectories between the source system (whether real or conceptual) and the model under test must be generated. Validity, whether replicative, predictive, or structural requires that these trajectories be equal (Zeigler et al.,

2000). *Verification* is the attempt to establish that the simulation relation holds between a simulator and a model i.e. the simulator faithfully implements the model's dynamic behavior. There are two general approaches to verification: formal proof of correctness and extensive testing (Zeigler et al., 2000). This is in congruence with the ideas of Robert Sargent (2011) that links data validity as a central concept linking the conceptual model, the computerized model and the problem entity (a.k.a real world "system"). Per Sargent, the relationship between the real world system and its conceptual model is called conceptual model validation, while the relationship between the system and the computerized model is called operational validation (that is support by a computational execution of the model). This is the modeling relation per Zeigler's definition. The relationship between the conceptual model and the computerized model is identified as computerized model verification. Rightly so, per Zeigler, this is the simulation relation that ensure a model is implemented correctly in a simulation environment and the entity (simulator) that ensures that a strict relation (i.e. simulation relation) exists between the conceptual model and the computerized model. Bair and Tolk (2013) summarized various definition towards a unified theory of validation. While model validation is user-faced, verification is implementation-specific (Mittal & Zeigler, 2017). The path to verification of simulation models is not straightforward and a huge gap exists. Formal methods for model verification is an active area of research (Gore & Diallo, 2013; Tolk et al., 2013).

We have shown how the proposed CAS-based M&S methodologies are emerging in current real-world applications. However, these M&S techniques remain difficult to Verify and Validate (V&V). Performing V&V is an exhaustive exercise for any simulation model. Due to inherent complexity in CAS simulation that comprise of a heterogeneous agent-based model where the behavior of each agent changes dynamically, V&V is a challenge of its own. The potential complexity to this issue does not stop here. CAS models usually contain a large number of variables, so there exist a prominent risk of over-fitting in the process of feature selection. Additionally, because CAS simulation is stochastic, a single run is insufficient to verify or to validate the quality of model parameters. Numerous runs are required to build confidence in the simulation results. Last but not the least, are the consequences of emergent behaviors: emergent properties of CAS V&V cannot be easily expressed. Thus, current standardized and formalized V&V methodologies will need to be modified and adapted to incorporate an evolutionary V&V framework for integrated CAS testing and evaluation (T&E) in unknown scenarios.

For CAS models, the validation aspect answers the question about how useful a model is in a particular scenario. The verification aspect falls-back to the co-simulation of various tools and paradigms that need to be brought in for an accurate simulation of a model. The entire simulation experiment, the model and the simulation-infrastructure must be automated through a model-based repository and transparent simulation framework.

According to Mittal (2014), validation is the only aspect that can be ensured in CAS M&S. Verification has to be ascertained using statistical and stochastic

methods. Rouff et. al. (2012) propose the following verification methodology:

1. A stability analysis capability that identifies instabilities given a system model and partitions the system model into stable and unstable component models.

2. A state space reduction capability that prunes the state space of an unstable component model without loss of critical fidelity.

3. High performance computing simulations to explore component behavior over a wide range of an unstable component's reduced state space and produce a statistical verification for the component.

4. A compositional verification capability that aggregates individual component verification.

5. Operational monitors to detect and take action to correct undesired unstable behavior of the system during operation.

However, we may also formulate the hypothesis that, inside a complex stochastic system, the V&V techniques should be stochastic as well. This would allow us the exploration of new combinations of model parameters as good candidates to reduce the divergence between the simulation and the real-world data. This process can be formulated through a deep learning approach at the meta-model layer over the CAS simulation. Indeed, an opportunity that we must avail in our next article.

8 Conclusions and Future work

Complex adaptive systems have emergent and adaptive structure and behavior. While there are many options available for CAS modeling, without a simulation-based approach, models cannot be verified and experimented in a n exhaustive and stochastic manner. Paradigms like MBSE , MDE, MBE while support traditional systems engineering need to be augmented with simulation-based methodologies to ensure they support complex systems engineering that integrate discrete and continuous systems yielding complex hybrid systems. This then needs to be augmented with evolutionary computation techniques to incorporate adaptive and emergent behaviors in a computational environment for large scale experimentation, testing and evaluation.

CAS model complexity at the structural level must be studied using Network Science to ascertain the impact a connected environment has on a particular agents/system situated in that environment. This is essential as the structure of the environment yields the overall behavior of CAS. Any change in the structure of the overall system resulting from behavior of an individual agent/system is a critical event that results in changing the behavior of agents/systems in that agent's neighborhood or far across the network through weak-ties. Indeed, a

new world that replete with cyber-physical systems and Internet of Things is a complex world.

The adaptive nature of CAS needs to be modeled using evolutionary computation techniques such as genetic algorithms, where in , global fitness functions reuse basic behavior building blocks to yield a behavior that is more in tune with the environment, thereby, situating the agent /system in the environment. This can only be achieved in a computational environment using simulation-based methodologies.

Emergent behavior is an essential element in any CAS study. While desired emergent behavior can be modeled in a top-down manner, computational environment becomes a necessity when bottom-up behavior needs to be studied and evaluated. Various agent-based modeling tools are available that can be used to model a fairly abstract model of CAS. The formalization of emergent behavior must be supported with fundamental scientific theories such as Cybernetics, Systems Theory, Network Science, Control Theory and Estimation Theory such that the observed emergent behavior in a computational environment is consistent with the known theories as well as with the system's domain of application. Any inconsistent emergent behavior should then be rejected on the basis of such evidence providing a learning opportunity to perform model correction or simulator correction. Both model correction and simulator correction are non-trivial endeavors when model is a multi-paradigm model and the simulator is a co-simulator that incorporates multi-simulation. A multi-paradigm model has to conform to a super-formalism such as DEVS formalism that can model various discrete event and complex continuous hybrid formalisms and adhere to Systems Theory thereby, yielding guaranteed emergent behavior. Having ensured that, the focus next is on the model-simulator relationship. The co-simulation environment must preserve this relationship to eliminate emergent behaviors arising out of simulator engineering.

Data-driven methodology is another essential element in any CAS study. The support given by a correct M&S must be augmented with a data strategy as it helps validation and verification of the model as well as of the simulator itself. The empirical data guides the model formulation . Various heuristics, meta heuristics and hyper heuristics are then designed as algorithms, which are then computationally implemented in an agent-based simulation environment. The simulation experiments yields simulation data that is then compared with raw data for evaluating model's validity. Any deviation between these two datasets is then treated as an opportunity to improve the model , as well as of heuristics. By the definition of emergent behavior, these macro-behaviors are sometimes irreducible at the micro-level. Consequently, the integration of data-driven methodology is essential to fine-tune both the model and heuristics as it is unknown where to make the correction at the agent level. These heuristics represent macro-level algorithms that act on a group of agents/systems. Once this cyclical process is implemented and is supported by a robust co-simulation environment, meaningful CAS engineering can be attempted.

CAS engineering is a complex endeavor requiring professionals from multiple disciplines. Their subject matter expertise of the agent, the systems and the

operational environment has to be translated into valid models in a verifiable simulation environment. This requires a partnership between domain experts, modeling experts and simulation infrastructure experts. CAS engineering will not become possible unless the undesired emergent behaviors are completely ruled from a computational environment or are known a priori so that they can be knowledgeably eliminated. A computational simulation-based environment provides experimentation opportunities to validate a CAS model, such that it become s predictable. Only when a model becomes valid and predictable, real world systems engineering can begin.

Review Questions

1. What is the difference between complicated and complex systems?
2. What theoretical background do you need to understand both complex and adaptive systems?
3. What is the difference between a complex adaptive system and System-of-system?
4. When does a system of system becomes a complex adaptive system?
5. How can you incorporate emergent behavior in engineering complex adaptive systems?
6. Why is a simulation environment engineering a critical component in engineering complex adaptive systems?
7. Which is the role of evolutionary computation in the design of complex adaptive systems?
8. Is there always the need for human experts in the process of modeling complex optimization heuristics?
9. What is the difference between model verification and model validation? Can we ensure verification and validation in CAS models?

References

Adaptive character of though-rational (act-r). (2017). `http://aiweb.techfak.uni-bielefeld.de/content/bworld-robot-control-software/`. ([Online; accessed 20-Feb-2017])

Alba, E., & Cotta, C. (2002). Parallelism and evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, *6*, 443-462.

Alexander, C. (1964). *Notes on the synthesis of form.* Harvard University Press.

Andre, D., & Teller, A. (1999). Evolving team darwin united. In M. Asada & H. Kitano (Eds.), *Robocup-98: Robot soccer world cup ii* (pp. 346–351). Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/3-540-48422-1_28 doi: 10.1007/3-540-48422-1_28

Arifin, S. M. N., & Madey, G. R. (2015). Verification, validation, and replication methods for agent-based modeling and simulation: Lessons learned the hard way! In L. Yilmaz (Ed.), *Concepts and methodologies for modeling and simulation: A tribute to tuncer ören* (pp. 217–242). Cham: Springer International Publishing. Retrieved from http://dx.doi.org/10.1007/978-3-319-15096-3_10 doi: 10.1007/978-3-319-15096-3_10

Arnaldo, I., Cuesta-Infante, A., Colmenar, J. M., Risco-Mart?n, J. L., & Ayala, J. L. (2013). Boosting the 3D thermal-aware floorplanning problem through a master-worker parallel MOEA. *Concurrency and Computation: Practice and Experience*, *25*(8), 1089?1103. doi: 10.1002/cpe.2902

Arroba, P., Risco-Martín, J. L., Zapater, M., Moya, J. M., & Ayala, J. L. (2014). Enhancing Regression Models for Complex Systems Using Evolutionary Techniques for Feature Engineering. *Journal of Grid Computing*, *13*(3), 409-423. Retrieved from https://drive.google.com/file/d/0B2sHHn2H0oxtRnpDMDZDb0NsOW8/view?usp=sharing doi: 10.1007/s10723-014-9313-8

Ashby, W. (1956). *An introduction of cybernetics*. Chapman and Hall.

Bair, L. J., & Tolk, A. (2013). Towards a unified theory of validation. In *Proceedings of the 2013 winter simulation conference: Simulation: Making decisions in a complex world* (pp. 1245–1256). Piscataway, NJ, USA: IEEE Press. Retrieved from http://dl.acm.org/citation.cfm?id=2675983.2676141

Barabasi, A. (2003). *Linked: How everything is connected to everything else and what it means for business, science and everyday life*. Penguin Books.

Branke, J. (2001). *Evolutionary optimization in dynamic environments*. Kluwer Academic Publishers.

Branke, J., & Schmeck, H. (2008). Organic computing. In (p. 123-140). Springer Berlin Heidelberg.

Brownlee, J. (2007). *Complex adaptive systems* (Tech. Rep.). Complex Intelligent Systems Laboratory, Centre for Information Technology Research, Faculty of Information Communication Technology, Swinburne University of Technology, Melbourne, Australia.

Camus, B., Paris, T., Vaubourg, J., Presse, Y., & Bourjot, C. (2017). *Mecysco: a multi-agent devs wrapping platform for the co-simulation of complex systems [research report]* (Tech. Rep.). ORIA, UMR 7503: CNRS.

Cellier, F. E. (1977, September). Combined continuous/discrete system simulation languages: Usefulness, experiences and future development. *SIGSIM Simul. Dig.*, *9*(1), 18–21. Retrieved from http://doi.acm.org/10.1145/1102505.1102514 doi: 10.1145/1102505.1102514

Chellapilla, K., & Fogel, D. B. (2001, Aug). Evolving an expert checkers playing program without using human expertise. *IEEE Transactions on Evolutionary Computation*, *5*(4), 422-428. doi: 10.1109/4235.942536

Clymer, J. R. (2009). *Simulation-based engineering of complex systems.* Wiley.

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms.* Wiley.

Deb, K., Mohan, M., & Mishra, S. (2005). Evaluating the ϵ-domination based multi-objective evolutionary algorithm for a quick computation of pareto-optimal solutions. *Evolutionary Computation*, *13*(4), 501-525. doi: 10 .1162/106365605774666895

Flexviz. (2017). `https://sourceforge.net/projects/flexviz/`. ([Online; accessed 20-Feb-2017])

Gephi. (2017). `http://gephi.org`. ([Online; accessed 20-Feb-2017])

Gore, R., & Diallo, S. (2013). The need for usable formal methods in verification and validation. In *Proceedings of the 2013 winter simulation conference: Simulation: Making decisions in a complex world* (pp. 1257–1268). Piscataway, NJ, USA: IEEE Press. Retrieved from `http://dl.acm.org/citation.cfm?id=2675983.2676142`

Gridlab-d. (2017). `http://www.gridlabd.org`. ([Online; accessed 20-Feb-2017])

Holland, J. (1992). Complex adaptive systems. *Daedalus*, *121*, 17-30.

Igraph. (2017). `http://igraph.org/redirect.html`. ([Online; accessed 20-Feb-2017])

Jack. (2017). `https://www.plm.automation.siemens.com/en_us/products/tecnomatix/manufacturing-simulation/human-ergonomics/jack.shtml`. ([Online; accessed 20-Feb-2017])

Java act-r. (2017). `http://jact-r.org/`. ([Online; accessed 20-Feb-2017])

Kilkki, O., Kangasrääsiö, A., Nikkilä, R., Alahäivälä, A., & Seilonen, I. (2014). Agent-based modeling and simulation of a smart grid: A case study of communication effects on frequency control. *Engineering Applications of Artificial Intelligence*, *33*, 91 - 98. Retrieved from `http://www.sciencedirect.com/science/article/pii/S0952197614000839` doi: http://dx.doi.org/10.1016/j.engappai.2014.04.007

Kim, Y., & Kim, T. (1998). A heterogeneous simulation framework based on the devs bus and the high level architecture. In (Vol. 1).

Kinemation. (2017). `http://www.cs.cmu.edu/~german/research/HumanApp/humanapp.html`. ([Online; accessed 20-Feb-2017])

Lee, E. (2008). Cyber physical systems: Design challenges. In *Ieee international symposium on object oriented real-time distributed computing.*

Lee, K., Hong, J., & Kim, T. (2015). System theoretic formalisms for combined discrete-continuous system simulation. *ETRI Journal*, *37*.

Maier, M. (1998). Architecting principles for system-of-systems. *Systems Engineering*, *1*, 267-284.

Matlab/simulink. (2017). `ttps://www.mathworks.com/products/simulink.html`. ([Online; accessed 20-Feb-2017])

Miller, J. H., & Page, S. E. (2009). *Complex adaptive systems: An introduction of computational models of social life.* Princeton Press.

Mittal, S. (2013a). Emergence in stigmergic and complex adaptive systems: A

formal discrete event systems perspective. *Cognitive Systems Research*, *21*, 22-39.

Mittal, S. (2013b). Netcentric complex adaptive systems. In *Netcentric systems of systems engineering with devs unified process.* CRC Press.

Mittal, S. (2014). Model engineering for cyber complex adaptive systems. In *European modeling and simulation symposium.*

Mittal, S., & Martin, J. L. R. (2013). *Netcentric system of systems engineering with devs unified process.* Boca Raton, FL USA: CRC Press.

Mittal, S., & Rainey, L. (2015). Harnessing emergence: The control and design of emergent behavior in system of systems engineering. In *Proceedings of summer computer simulation conference.* SCS.

Mittal, S., Ruth, M., Pratt, A., Lunacek, M., Krishnamurthy, D., & Jones, W. (2015). A system-of-systems approach for integrated energy systems modeling and simulation. In *Summer computer simulation conference.*

Mittal, S., & Zeigler, B. P. (2017). Theory and practice of m&s in cyber environments. In A. Tolk & T. Oren (Eds.), *Modeling and simulation as a profession.* Wiley & Sons.

Mittal, S., Zeigler, B. P., Martín, J. L. R., Sahin, F., & Jamshidi, M. (2008). Modeling and simulation for systems of systems engineering. In *System of systems engineering* (pp. 101–149). John Wiley & Sons, Inc. Retrieved from `http://dx.doi.org/10.1002/9780470403501.ch5` doi: 10.1002/9780470403501.ch5

Neon toolkit. (2017). `http://semanticweb.org/wiki/NeOn_Toolkit.html`. ([Online; accessed 20-Feb-2017])

Netlogo. (2017). `https://ccl.northwestern.edu/netlogo/`. ([Online; accessed 20-Feb-2017])

Networkx. (2017). `https://networkx.github.io/index.html`. ([Online; accessed 20-Feb-2017])

Newman, M. (2001). Clustering and preferential attachment in growing networks. *Physical Review*, *64*.

Ns3. (2017). `https://www.nsnam.org`. ([Online; accessed 20-Feb-2017])

Nutaro, J., Kuruganti, P. T., Shankar, M., Miller, L., & Mullen, S. (2008). Integrated modeling of the electric grid, communications, and control. *International Journal of Energy Sector Management*, *2*, 420-438.

Omnet. (2017). `https://omnetpp.org`. ([Online; accessed 20-Feb-2017])

Opnet modeler. (2017). `https://www.riverbed.com/products/steelcentral/opnet.html`. ([Online; accessed 20-Feb-2017])

Pajek. (2017). `http://mrvar.fdv.uni-lj.si/pajek/`. ([Online; accessed 20-Feb-2017])

Plsek, P. E., & Greenhalgh, T. (2001). The challenge of complexity in health care. *BMJ*, *323*(7313), 625–628. Retrieved from `http://www.bmj.com/content/323/7313/625` doi: 10.1136/bmj.323.7313.625

Praehofer, H. (1991). System theoretic formalisms for combined discrete-continuous system simulation. *International Journal of General Systems*, *19*, 226-240.

Pratt, A., Ruth, M., Krishnamurthy, D., Sparn, B., Lunacek, M., Jones, W.,

... Marks, J. (2017). Hardware-in-the-loop simulation of a distribution system with air conditioners under model predictive control. In *Ieee power engineering society general meeting.*

Protégé. (2017). http://protege.stanford.edu. ([Online; accessed 20-Feb-2017])

Rainey, L., & Tolk, A. (Eds.). (2015). *Modeling and simulation support for system of systems engineering applications.* Hoboken, NJ USA: John Wiley and Sons.

Repast/symphony. (2017). https://repast.github.io/repast_simphony.html. ([Online; accessed 20-Feb-2017])

Risco-Martín, J. L., Mittal, S., Fabero, J. C., Malagón, P., & Ayala, J. L. (2016). Real-time Hardware/Software Co-Design Using DEVS-based Transparent M&S Framework. In *Proceedings of the 2016 summer simulation multiconference (summersim'16).* Retrieved from https://drive.google.com/open?id=0B2sHHn2H0oxtU1Z3M0d3R3JIXzg

Robocup. (2017). http://www.robocup.org. ([Online; accessed 23-Feb-2017])

Rouff, C., Buskens, R., Pullum, L., Cui, X., & Hinchey, M. (2012). The adaptiv approach to verification of adaptive systems. In *Proceedings of the fifth international c* conference on computer science and software engineering* (pp. 118–122). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/2347583.2347600 doi: 10.1145/2347583.2347600

The r-project. (2017). https://www.r-project.org. ([Online; accessed 20-Feb-2017])

Sargent, R. G. (2011). Verification and validation of simulation models. In *Proceedings of the winter simulation conference* (pp. 183–198). Winter Simulation Conference. Retrieved from http://dl.acm.org/citation.cfm?id=2431518.2431538

Soar. (2017). http://soar.eecs.umich.edu. ([Online; accessed 20-Feb-2017])

Statnet. (2017). https://statnet.csde.washington.edu/. ([Online; accessed 20-Feb-2017])

Szabo, C., & Teo, Y. M. (2015, September). Formalization of weak emergence in multiagent systems. *ACM Trans. Model. Comput. Simul.*, *26*(1), 6:1–6:25. Retrieved from http://doi.acm.org/10.1145/2815502 doi: 10.1145/2815502

Tolk, A. (2015). The next generation of modeling & simulation: Integrating big data and deep learning. In *Proceedings of the conference on summer computer simulation* (pp. 1–8). San Diego, CA, USA: Society for Computer Simulation International. Retrieved from http://dl.acm.org/citation.cfm?id=2874916.2874964

Tolk, A., Diallo, S. Y., Padilla, J. J., & Herencia-Zapana, H. (2013). Reference modelling in support of m&s—foundations and applications. *Journal of Simulation*, *7*(2), 69–82. Retrieved from http://dx.doi.org/10.1057/jos.2013.3 doi: 10.1057/jos.2013.3

Twouse. (2017). http://semanticweb.org/wiki/TwoUse_Toolkit.html. ([Online; accessed 20-Feb-2017])

Vangheluwe, H. (2000). Devs as a common denominator for multi-formalism

hybrid systems modelling. In *Cacsd. conference proceedings. ieee international symposium on computer-aided control system design (cat. no.00th8537)* (p. 129-134). doi: 10.1109/CACSD.2000.900199

Vangheluwe, H., & Lara, J. (2003). Foundations of multi-paradigm modeling and simulation: Computer automated multi-paradigm modelling: Metamodelling and graph transformation. In *Proceedings of the 35th conference on winter simulation: Driving innovation* (pp. 595–603). Winter Simulation Conference. Retrieved from `http://dl.acm.org/citation.cfm?id=1030818.1030900`

Wang, X., Choi, T. M., Liu, H., & Yue, X. (2016, Nov). Novel ant colony optimization methods for simplifying solution construction in vehicle routing problems. *IEEE Transactions on Intelligent Transportation Systems*, *17*(11), 3132-3141. doi: 10.1109/TITS.2016.2542264

Wolf, T., & Holvoet, T. (2005). Emergence versus self-organization: Different concepts but promising when compbined. *Lecture Notes in Artificial Intelligence*, *3464*, 1-15.

Yilmaz, L. (2006). Validation and verification of social processes within agent-based computational organization models. *Computational & Mathematical Organization Theory*, *12*(4), 283–312. Retrieved from `http://dx.doi.org/10.1007/s10588-006-8873-y` doi: 10.1007/s10588-006-8873-y

Zeigler, B. P., Praehofer, H., & Kim, T. G. (2000). *Theory of Modeling and Simulation. Integrating Discrete Event and Continuous Complex Dynamic Systems* (2nd ed.). Academic Press.

Zhang, Y., w. Gong, D., & Cheng, J. (2017, Jan). Multi-objective particle swarm optimization approach for cost-based feature selection in classification. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *14*(1), 64-75. doi: 10.1109/TCBB.2015.2476796

Zheng, Y. J., Zhang, M. X., Ling, H. F., & Chen, S. Y. (2015, Feb). Emergency railway transportation planning using a hyper-heuristic approach. *IEEE Transactions on Intelligent Transportation Systems*, *16*(1), 321-329. doi: 10.1109/TITS.2014.2331239

Biography

Dr. **Saurabh Mittal** is Lead Systems Engineer/Scientist at MITRE, founder and President of Dunip Technologies, LLC and Vice President-Memberships for Society of Computer Simulation (SCS) International. He is also currently serving as General Chair for Spring Simulation Multi-conference 2017, Associate Editor-in-Chief for Transactions of SCS and Editor-in-Chief of Enterprise Architecture Body of Knowledge (EABOK) Consortium. Previously, he was a fulltime scientist and architect at National Renewable Energy Laboratory, Department of Energy at Golden, Colorado, where he contributed to complex energy systems co-simulation environments. He also worked at L3 Link Simulation & Training as a contractor to US Air Force Research Lab at Wright-Patterson Air Force Base, Ohio where he integrated artificial agents and various

cognitive architectures in Live, Virtual and Constructive (LVC) environments using formal Systems theoretical model-based engineering approaches. He was a Research Assistant Professor at the Department of Electrical and Computer Engineering at the University of Arizona. Dr. Mittal served as General Chair of Springsim'17, SummerSim'15, Vice General Chair for SpringSim'16 and SummerSim'14, and Program Chair for SpringSim'15. He has co-authored nearly 60 articles in various international conferences and journals, including a book titled "Netcentric System of Systems Engineering with DEVS Unified Process", CRC Press in 2013. The book serves the areas of executable architectures, service-oriented simulation, System of Systems engineering using Department of Defense Architecture Framework (DoDAF), multiplatform modeling, intelligence-based, complex, adaptive and autonomous systems, and large scale M&S integration and interoperability.

Dr. **José L. Risco-Martín** is Associate Professor at Complutense University of Madrid. He is head of the Department of Computer Architecture and Automation. He is currently serving as General Chair of Summer Simulation Multi-Conference 2017. Previously, he was Assistant Professor at Colegio Universitario de Segovia and Assistant Professor at C.E.S. Felipe II de Aranjuez. Dr. Risco-Martín served as Program Chair for SummerSim'15, General Chair for Summer Computer Simulation Conference 2016 and Vice General Chair for for SummerSim'16. He has co-authored more than 100 articles in various international conferences and journals. His research interests focus on design methodologies for integrated systems and high-performance embedded systems, including new modeling frameworks to explore thermal management techniques for Multi-Processor System-on-Chip, dynamic memory management and memory hierarchy optimizations for embedded systems, Networks-on-Chip interconnection design, low-power design of embedded systems and more generally Computer-Aided Design in M&S of Complex System, with emphasis on DEVS-based methodologies and tools.