MODEL ENGINEERING FOR CYBER COMPLEX ADAPTIVE SYSTEMS

Saurabh Mittal

Dunip Technologies LLC, Colorado, USA
smittal@duniptech.com

ABSTRACT

Sociotechnical systems pervade every facet of our life today. These IT systems interact with live users that result in emergent behaviors leading to their classification as complex adaptive systems (CAS). Cyber-CAS (CyCAS) exists in contemporary society when such systems have Internet as their platform. Modeling and Simulation (M&S) for CyCAS is extremely difficult: partly due to inherent complex nature of enterprise IT architectures and partly due to lack of sufficient formal processes used in enterprise systems architecting. Fundamentally, the interaction between various elements is hard to pin down as enterprise architecture components are often treated as black-box with limited information about their internal behaviors. In this article, I provide an overview of the M&S approach for enterprise frameworks, introduce CyCAS, the tools and metrics for various CyCAS Views and the sandbox requirements for CyCAS M&S.

Keywords: Cyber Complex Adaptive Systems, CyCAS, DUNIP, second order cybernetics, DEVS

1. INTRODUCTION

Today the sociotechnical systems pervade every facet of life, such as economics, politics, medicine, agriculture, energy, and so on. The information is distributed across the entire landscape and there are a million eyes instantly reacting to a new technology or a blog post that turns viral any minute. Modeling such a system seems impossible, as the notion of system cannot be formally identified even though everybody talks of a 'system' in general terms. These Information Technology (IT) systems have different architectures for different purposes and as human interact with these systems, both the IT systems and the interacting users adapt to the dynamic landscape. To manage the design of an Enterprise IT system, each underlying architecture subscribes to an enterprise architecture framework that anchors various enterprise architectures towards a common goal. Modeling an enterprise framework for the formal modeling and simulation effort remains a vision and the problem of executable architecture has not yet been completely solved. Agent based modeling (ABM) is the strongest contender towards modeling and simulation of system that require many autonomous or semi-autonomous entities. However, ABM alone cannot describe these complex adaptive systems in the cyber-world.

In this article, I will define Cyber Complex Adaptive Systems (CyCAS) and how the foundation of formal modeling and simulation could be applied towards developing a framework to study CyCAS. I will elaborate on the various perspectives CyCAS can be described through and then present the tools and metrics required to support the development of these perspectives.

The article is organized as follows. Section 2 provides a background overview of foundational M&S theory, systems framework and interoperability considerations. Section 3 describes the M&S framework for tooling and various enterprise architecture frameworks that make the problem even more complex. A discussion of complexity and system-of-system (SoS) perspective is provided in Section 4. Section 5 introduces CyCAS with Section 6 proposing various CyCAS Views within the CyCAS enterprise framework. Section 7 elaborates on the sandbox requirements for CyCAS modeling. Finally, conclusions are presented.

2. M&S ARCHITECTURE, SYSTEMS FRAMEWORK AND INTEROPERABILITY

As we all know, model is an abstraction of reality with some valid assumptions and the simulator is an algorithm or a program that manifests the behavior of model. The discipline of designing a simulator is called Simulation engineering and considerable effort has been spent in the last few decades in developing robust simulation engines that leverage high performance computing (HPC). On the contrary, model engineering (ME) aims at setting up a systematic, standardized, and quantifiable engineering methodology to manage the data, knowledge, activities, processes and organizations/people involved in the full life cycle of the model, in order to obtain the general term of credible model theory, methods, technology, standards and tools with the minimum cost.

Model engineering has been in use since the mid-1980s through the Model-based Engineering (MBE) and Model-based Systems Engineering (MBSE) paradigms (Zeigler 1976, Zeigler 2000, Mittal and Martin 2013b). Progress in ME should profit from the progress made in model-based and model-driven methods developed in various systems and software engineering communities. At the same time, it should address the distinct challenges posed in the modeling and simulation (M&S) domain clarified by its theoretical and conceptual frameworks.

M&S engineering can be best understood by the layered M&S architecture initially proposed by Zeigler, Praehofer and Kim (2000), and later elaborated by Mittal et.al (2008) in the context of system-of-systems engineering, and Linguistic Levels of Interoperability (i.e., pragmatic, semantic and syntactic levels). Simulation is associated with the computational domain while Modeling is generally associated with the semantic domain (Figure 1a). They both come together at the pragmatic level where the purpose and limitations are defined. The development of an M&S solution is guided through the pragmatics that involve various considerations for M&S engineering, such as, scope, collaboration, decision, search-space, and computational requirements. Computer and Software Engineers largely handle the discipline of simulator engineering while the discipline of model engineering accounts for the subject matter expert (SMEs) versed in domain theory and the model developer. In some cases, either SMEs take the plunge to learn various programming languages to encode the model (and the associated model editing environment) that serve their purposes that go beyond the existing simulation tools or they prefer to stay within the confines of mathematics. Regardless of the approach, the scope of model is defined by the questions it is meant to answer or phenomenon it should describe. Certainly, the question of which model is accurate is a difficult question as it is dependent on the pragmatics (Figure 1b). When model accurately represents the pragmatics, it is called a valid model and the process is called Validation. When the model behavior is correctly implemented by the simulator, the model is verified. This process is called Verification. For any model engineering effort, the verification and validation (V&V) (Figure 1b) studies are as critical as the M&S engineering process and any M&S effort without V&V raises questions on the credibility of the model.

The simulators are computational entities built using various programming languages with a defined syntax. Communication and integration between different simulators has been pursued in the derived discipline of Distributed Simulation with an objective of achieving simulation interoperability. Various standards (DIS, HLA, TENA, CORBA, etc.) are in use that have facilitated syntactic interoperability. However, the model interoperability, as it happens in the semantic domain has remained elusive, primarily for the reason that model pragmatics are different for different models as assumptions are seldom aligned beforehand. When the pragmatics are aligned, model interoperability can be achieved as the composite model can now be validated.

While conceptually, this is easy to understand, the problem becomes compounded with complexity when the models and simulators are engineered with distinct architectures, some proprietary, some open-source, some considering model and the simulator as separate elements and some coupling them together, making them "silos" with no system-of-systems vision and extensibility. As they are used in the field, they are called "simulation systems" and become black-boxes thereby masking away the V&V process that need to be defined for the system-of-system (SoS) that they begin to participate in. As elaborated in Mittal et.al. (2008), the component-based nature of SoS needs a next level of specification provided by an architecture framework that allows specification and integration of architectures. These architectures within the enterprise framework can then be subjected to the formal Test and Evaluation (T&E) procedures thereby validating their designs (Figure 1c). Verification in SoS remains elusive and it is largely assumed that the component system accurately performs the claimed functionality.

Model engineering in an enterprise environment need to be investigated in more detail and the V&V process need to be defined as well.

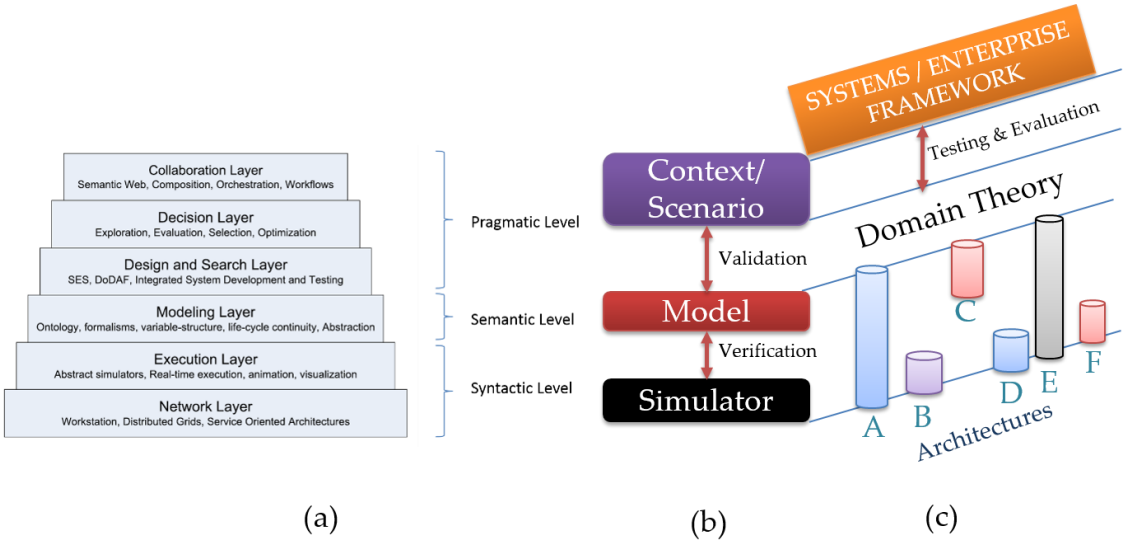

Figure 1: M&S Theory, Architectures, V&V, T&E and Interoperability

3. MSVC FRAMEWORK AND ENTERPRISE ARCHITECTURE FRAMEWORKS

Modeling editors and simulation engines are distinct pieces of an M&S solution and require different engineering approaches. However, when there are considered one and lack any formal engineering effort, they are programmed as one "simulation software" intending to serve only the purpose at hand or maybe just rapid prototyping.

When the model and simulator development is considered distinct, and when coupled with formal Software Engineering practices, they can be best be summarized by the Model-Simulator-View-Controller (MSVC) architecture framework (Mittal, Mak, Nutaro 2006) (Figure 2). As stated earlier, the pragmatics provide model's objective, the semantics provide the model definition and the syntax provides the model's computational implementation. In the Software Engineering realm, Model-View-Controller (MVC) architecture is the most widely used architecture to build software that has a decoupled Presentation layer. Many extensible Integrated Development Environment (IDE) frameworks are available that utilize MVC architecture framework (for example Netbeans Rich Client Platform (RCP), Eclipse RCP, Enterprise J2EE, Microsoft .NET framework, etc.)

The Model in MVC is where the actual semantics are constructed, the View is where the presentation and user-input of model is created and the Control is where the communication between the Model and the View is engineered. In MSVC, there is an additional component of Simulator that represents the underlying platform over which the model is executed. Moreover, the Controller aspect is made rich and takes active role as it pertains to the pragmatics aspect of the M&S development exercise. The MSVC framework is discussed at length by Mittal, Mak and Nutaro (2006) where they associate it with DEVS M&S theory.

While usage of RCP platforms provide a solution for a desktop environment and the resulting solution can effectively be a considered a black-box, enterprise level solutions bring more complexity in the mix where cohesion and loose-coupling provide a different design paradigm.

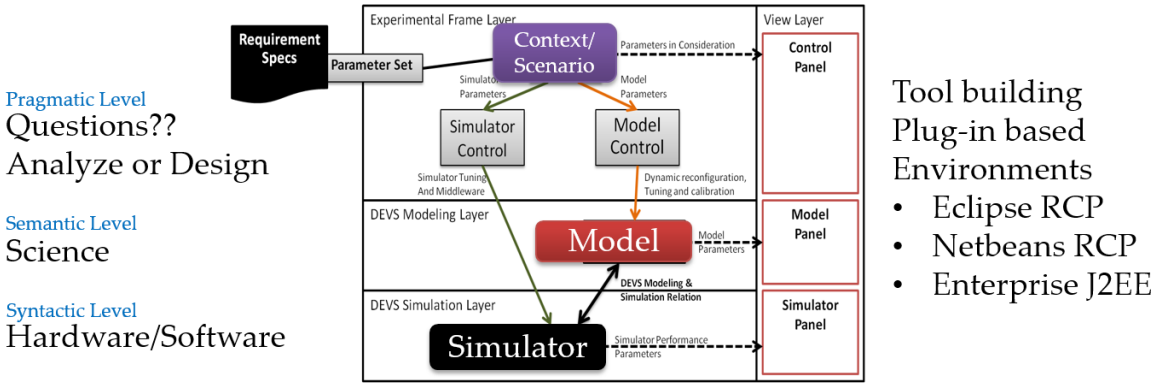

Figure 2: Model Engineering using MSVC Framework and Open Source tools

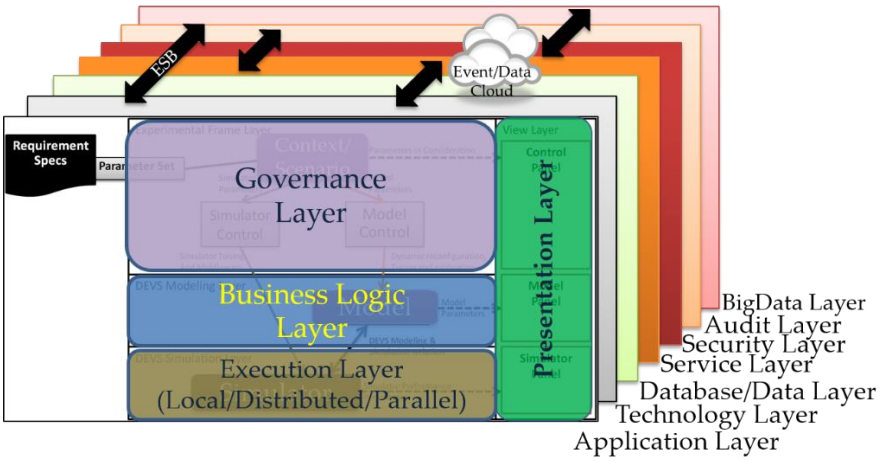

Figure 3: MSVC Framework and Enterprise Architectures

3.1. Architecture Frameworks

Enterprise frameworks are organized in various layers where each layer can have its own architecture. These layers are identified on the function they provide in serving the enterprise. Clearly, separation of concerns is at the heart of an enterprise framework, which in turn results in specific technologies being developed at each layer to enhance its functionality. Some of the identified layers are:

- Governance:
- Business Logic
- Execution
- Application
- Technology
- Database
- Service
- Security
- Audit
- Big Data

These layers, which have their own architectures, communicate using enterprise service buses (ESBs) that provide atomicity and various other message-based reliable communication services. This communication is largely in a netcentric domain where many Standards exist, facilitating solid integration. Event-driven Architectures (EDAs) are the latest in the list of enterprise architectures that provide a shared memory-store (an Event cloud) of user-specified queries and patterns that can access various other connected layers (Taylor, Yochem, and Phillips 2009).

Figure 3 shows the relation of enterprise framework with the MSVC pattern. The Controller is analogous to the Governance layer, the Model to the Business Logic layer, the Simulator to the Execution layer, and the View to the Presentation Layer. All the other layers are orthogonal to the MSVC layers. The MSVC is coupled together as an Application layer, as shown in Figure 3.

A framework enables the development of an architecture and it is imperative to have a framework to manage and utilize various architectures. The reader is encouraged to review frameworks like:

1. DoDAF (CIO 2010): Department of Defense Architecture Framework.
2. MoDAF (MoD 2012): Ministry of Defense Architecture Framework
3. TOGAF (TOG 2011): The Open Group Architecture Framework
4. Zachmann Framework (Zachmann 2008)
5. NAF (2013): NATO Architecture Framework
6. DNDAF (2010): Department of National Defense/ Canadian Armed Forced Architecture Framework
7. FEAF (2012): The Federal Enterprise Architecture Framework

While it is encouraging to know that these framework exists, the major problem that now arises is that there are too many frameworks based on the domains they serve. The problem initially thought to be contained within the confines of a framework, seems compounded by the presence of multiple architectures and multiple frameworks. Again, no wonder why various agencies who like to share information and data are plagued with the inherent complexity. Various ontology and data-dictionary-based solutions are utilized that try to harmonize the knowledge between two or more different frameworks. Efforts are ongoing (Gorman, Haperen & Bailey 2013).

4. COMPLEXITY AND SOS

Section 2 provided an overview on the fundamental problem of V&V with multiple M&S architectures after becoming components within an enterprise framework. While there are many examples of enterprise frameworks as enumerated in Section 3.1, currently there exists none for enabling an M&S enterprise architecture. To that end, Joint Live, Virtual and Constructive (JLVC) Vision 2020 brings together Cloud-enabled Modeling and Simulation (CEMS) services (Weter 2012) and Joint Training Enterprise Architecture (JTEA), are under development (Edgren 2012) for enterprise level Joint Force training (Irwin 2012). These assume that the underlying LVC environment is indicative of enterprise M&S architecture. Technically, the integration problem is solved but semantic interoperability remains elusive with V&V from M&S perspective still questionable. Moreover, it is still not clear how causality, control and emergent behaviors need to be addressed in a distributed discrete event simulation environment (Rainey and Mittal 2014). Indeed, a cloud-based simulation framework continues to be a vision where fundamental issues of ownership of state, shared environment, truth and simulation remain unaddressed (Tolk and Mittal, 2014). There is a fundamental difference between software-based discrete event simulation and systems-based discrete event simulation. While the former is strictly based on object-oriented software engineering paradigm, the latter enforces mathematical System Theory on the object-oriented discrete event simulation engine. A successful solution must incorporate ideas from Systems Theory and M&S engineering practices. Even though all the above mentioned can be used for developing an M&S architecture, the Execution Layer (Figure 3) needs to be strengthened by distributed simulation architecture based on Systems Theory in a netcentric domain as described by Mittal and Martin (2013a, b). Until system-based approaches are encouraged, the vision of developing an executable architecture, where an architecture model can be executed over an enterprise simulation infrastructure, will continue to be a vision.

The problem of emergent behaviors when such architectures are brought together as a SoS has gained widespread attention in recent years. In addition to the five SoS characteristics defined by Maier (1998), where the constituent systems are operationally independent, managerially independent, have evolutionary nature

within their own systems development life cycle, are geographically distributed and together they portray emergent behavior, *an SoS also has a purpose*, to begin with. Assuming each constituent architecture is a 100% functional system, passing all the verification and validation tests in isolation, it cannot be determined a priori if the same system can perform with an SoS and can deliver: (a) the emergent behavior that is desired of an SoS and (b) fulfills the overall goal i.e. validation of SoS. Each constituent system or an architecture becomes an 'agent' that has a unique level of autonomy.

Finally, the biggest issue of all is the presence of humans as an actor in an enterprise solution thereby giving a dynamic nature to the entire landscape wherein everything is evolving and adapting (which includes, the technology, the users, the developers, the knowledge, the architectures and the systems themselves). This is how a complex adaptive system in an enterprise environment comes into being and there exist no mechanisms to perform T&E before such SoSs are made live with real users.

5. CYBER COMPLEX ADAPTIVE SYSTEM (CYCAS)

Complex Adaptive Systems (CAS) are systems that display strong emergent behavior, have positive- and negative-feedback loops and have large number of adaptive agents. Examples of such systems include ant colonies, the biosphere, the brain, the immune system, the biological cell, businesses, communities, social systems, stock market, financial systems, High Frequency Trading systems, enterprise systems, etc. Some of these examples can also be classified as *open systems*. One of the characteristic features of these systems is the generation of new knowledge within the system that has causal properties resulting in adaptive behavior of the system components itself. CyCAS occurs in contemporary society where computational systems interact with both live human agents and virtual agents (software bots). The resulting complex system, for example, the World Wide Web, evolves in unknown ways not initially thought out by designers. Various types of communities emerge, that use the system in ways unthinkable by the system designers. Usage of Facebook to organize protests, Twitter to coordinate and You Tube to showcase demonstrations were critical in the Arab Spring revolution. On another note, usage of You Tube as a learning system makes everybody an educator. We frequently use do-it-yourself (DIY) videos on You Tube for our personal advancement and Massive Open Online Courses (MOOCs) give these videos a formal structure. Learning amidst such CAS happens in leaps and bounds and aptly so, the technology to create such systems advances in lock-step fashion. While we have technologies to engineer such netcentric complex systems today, we are short of methodologies to understand the emergence of adaptive stance in these engineered systems. They are difficult to study due to unavailability of a 'CAS' sandbox.

The engineering process is a formal process that is applied to solve a particular problem and deliver a system that addresses it. With CyCAS, even the initial objective takes a spiral upturn or downturn (based on one's perspective) and continuously evolves. In contemporary society, every Cyber-system must have a component of computational emergence that studies the life-cycle of such systems.

I identify eight characteristics of CyCAS as follows:

5.1. Human-in-the-System

Every cyber system has a human as a critical component of the system. A CyCAS model accordingly must have infrastructure to manage user model profiles, the usage spread (distribution of users with respect to their roles), the capability to understand system responses for various usage spreads and the mechanisms to perform role switching for each user. Various human behavior modeling approaches and context quantification approaches (Mittal and Zeigler, 2014) need to be incorporated from fields like cybernetics, cognitive psychology, agent-based modeling etc.

5.2. Multi-agent-system

A CyCAS is a multi-agent system (MAS) where the agents can be heterogeneous, homogeneous, deliberative or communicative in varying degrees and combinations. MAS (Weiss 2000) is a fairly new discipline of Distributed Artificial Intelligence (DAI) (Bond, Gasser 1988) that focuses on behavior management and design of agent-based systems. Canonically, such systems are multi-level algorithms subjected to multi-level control in a real-time environment.

5.3. Control and communications in a netcentric environment

The communication aspect of CyCAS cannot be understated as it is the source of all emergent behavior. This Standards-based communication in a netcentric environment can be point-to-point, inter and intra-enterprise, and may include blackboard architectures with shared memory infrastructure. The presence of communication links at various levels establishes positive and negative feedback loops and unless such aspect is dealt per Systems Theory-based "closure under composition" principle, it is hard to establish who is in control of state and time for the resulting non-linear system.

5.4. Resource-constraints and economy of scale

Each constituent system operate within the confines of real world and has finite resources that need to be used efficiently and in a priority-based manner as dynamic nature of pragmatics within CyCAS re-purpose existing resources. Scalability and resourcefulness need to be addressed in a formal manner as failure of a constituent system may result in cascaded failures. Constraint-graphs facilitate the evaluation of resource availability and limitations.

5.5. Emergent Attention and Second order cybernetics

As the CyCAS design is motivated by the pragmatics (overall SoS goal), various observers (models) and auditing systems need to exist within the CyCAS to ensure that the resultant emergent behavior is truly the intended one. Usage of BigData technologies to engineer advanced pre- and post-analytics system must conform to Second-order cybernetics philosophy (Heylighen and Joslyn, 2001) where both the observer and the observed define the actual system. The emergent attention (Mittal and Zeigler, 2014) becomes a critical aspect of such systems that indicates the most active constituent system (or a group of sub-systems) in a resource-limited and time-constrained environment.

5.6. Phase Transition

As SoS manifests emergent behaviors, the constituent systems may respond in unknown ways due to their black-box nature. This may lead to emergence of unknown behaviors not specified in the design. Sometime these behavior are benign and sometimes puts the entire SoS into an undesired trajectory. Consequently, a CyCAS model must be able to attend to these phase transitions through computational emergence studies that provides information about the adaptive stance of CyCAS and whether it continues to display the intended emergent behavior.

5.7. Structure of knowledge

Each of the constituent system being a black-box implements its own knowledge-base. While they can be syntactically integrated successfully, achieving semantic interoperability is the holy-grail of systems interoperability and CyCAS. Before semantic interoperability is achieved, knowledge and ontological structures have to be ensured and validated by the SMEs. Issues like knowledge sharing, transformation, evolution, computational representation (ontology), interoperability, context switching according to emergent attention, and new knowledge synthesis have to be addressed at multiple operational levels. Knowledge management and evolution roadmap is a critical component of CyCAS and demarcates the difference between a *weak* or a *strong* emergent system (Figure 4) (Mittal, 2013; Mittal and Zeigler, 2014).

5.8. Resilient or Anti-fragile

Finally, CyCAS undergo a lot of dynamic reconfiguration due to their component nature. It needs to be formally determined if the engineered CyCAS is fail-fast, resilient/robust or anti-fragile where it continues to function, albeit evolving, in unknown ways.

6. CYCAS FRAMEWORK

Based on the eight characteristics identified in the previous section, this section presents a framework to engineer a CyCAS. Following are the assumptions:

1. CyCAS is a digital complex adaptive system where the notion of Object is ubiquitous at every level of specification, and is rapidly being replaced by the notion of a 'smart' object that may have sense-making capabilities.
2. System behavior is indicative of events that are manifested externally by a constituent system and are communicated using structured messages in a netcentric environment
3. Human in the most complex and the least predictive element. However, it is the most critical element that transforms a netcentric system into a CyCAS.

CyCAS Framework consists of eight views corresponding to each of the characteristics. Table 1 lists the eight views, the capabilities of tools that implement these views and the metrics that ensure that the tools are valid.

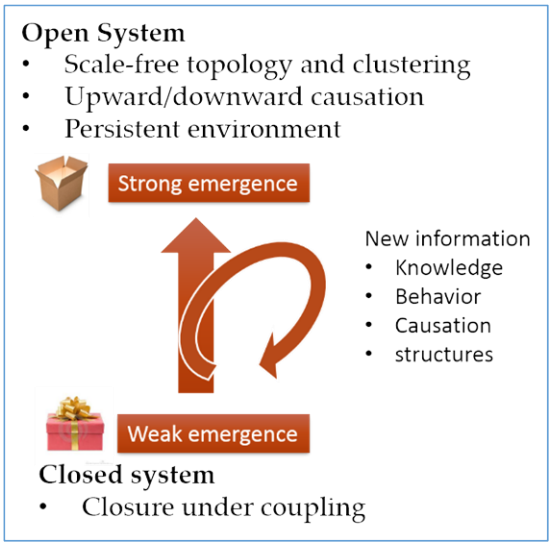

Figure 4: Strong and Weak Emergence

7. SANDBOX REQUIREMENTS AND THE ENGINEERING PROCESS

As we strive to adhere to Systems Theory for a CyCAS sandbox, we must be able to classify the eight CyCAS views into structure and behavior. This will allow us to separate the static and dynamic nature of the views. However, one has to remember the fact that the CyCAS structure continuously changes as the CyCAS behavior evolves. In a CyCAS, variable structure modeling is a critical capability (Mittal, 2013). Following is the classification:

I. Structure
   a. Control and Communication View
   b. Resource and Constraints View
   c. Knowledge View
   d. Resilience View

II. Behavior
   a. Human View
   b. Multi-agent View
   c. Emergence View
   d. Phase Transition View

Table 1: CyCAS Views, tool capabilities and various metrics for model engineering

ID	CyCAS Views	Tool Capabilities	Metrics
5.1	**Human View**	Cognitive architectures, Live, Virtual and Constructive (LVC) environments, user behavior modeling	Behavior quality, spread and quantification, cognitive plausibility, contextual realism, quantized context
5.2	**Multi-agent View**	Agent structure, behavior and interactions with other agents or environment, closed-under-composition	Ease of model-transformation and model integration, partial observability, group cohesion, shared goals
5.3	**Control and Communication View**	Hierarchical organization, logical structures and supervisory control (similar to DoDAF OV-7 and SV-7)	Degree of control (from centralized and totalitarian to completely decentralized and autonomous), feedback loops
5.4	**Resource and Constraints View**	Similar to DoDAF 2.0 System View 4, 5 resources. Constraints similar to policy considerations in DoDAF OV-6a	Utilization, availability, limitations, affordance
5.5	**Emergence View**	Multi-level instrumentation, Big Data, expected behaviors, causal behaviors, novel behaviors	Multi-level behavior validation and recognition, computational emergence, agent adaptation
5.6	**Phase Transition View**	System behavior transition matrix (similar to DoDAF SV-3)	Multi-level transition probabilities, credit assignment and new behavior detection and encoding
5.7	**Knowledge View**	Ontologies (data and its relationships)	Semantic network, semantic validity through SME and keyword-rank
5.8	**Resilience View**	Experimental frames	Degree of robustness at multiple-levels

While providing further details on each of the views is outside the scope of the article and is left for future work, I would like to describe the Multi-agent View in a bit more detail. As a CyCAS is a MAS at the fundamental level, there are many agent-based modeling (ABM) tools available. For example, JAMES II, Repast, NetLogo, Mason, Flame, Soar, Swarm, etc. To the best of my knowledge, none conforms to the Systems Theory's closure under composition principle and infact, many leverage the flat nature of agent based systems with limited encapsulation and hierarchy. The problem is rooted in usage of only the Software Engineering practices and ignoring the Systems Theory.

In the classical paradigm, and not going too far back for the purpose of this discussion, let us look at the 4th generation languages such as C. There is the *main* function and there are subroutines that invoke each other. Then came the Object-Oriented paradigm which brought forward concepts like inheritance, data encapsulation, and polymorphism, and grouped the set of subroutines within an Object as its 'behavior'. Capitalizing on this, the ABM community made the Object an Agent by adding constructs like purpose, behaviors and capabilities (i.e. a smart object). They also provided an agent execution environment where these agents could manifest their behavior with a virtual environment. All the existing ABM tools with the exception of JAMES II, have this notion of software agent with varying implementations and integration with a real or synthetic environment. Sometimes, there is a scientific theory behind the ABM tool, for example SOAR has cognitive psychology as its foundation (Laird, 2012). On the other hand, a Systems-based agent, for example, a DEVS agent, has the notion of a system attached to it (Figure 5). It is built on formal semantics and adheres to the Systems Theory (Zeigler, Praehofer and Kim, 2000, Mittal, 2013). It also provides a simulator, a simulation protocol and a distributed high performance engine for agent/system model's execution and ensures that Systems Theory is not violated. The DEVS M&S theory (Zeigler, Praehofer and Kim, 2000) is applicable to both discrete and continuous hybrid systems and has the notion of abstract time. It models the dynamical behavior and has the notion of elapsed time. Work by Mittal and Martin (2013a,b) packages all these functionalities in a netcentric DEVS Virtual Machine (VM) that provides an agent-execution environment in a CyCAS setting.

To develop a CyCAS sandbox, Table 2 provides an overview of the DEVS M&S elements as applicable to implementation of the structural and behavioral components of CyCAS. The DEVS Unified Process and the related work by Mittal, Martin and Zeigler in the last decade (Zeigler 2013) bring technologies like DEVSML Stack, DEVSVM, metamodeling, domain specific languages (DSLs) to be integrated with the DEVS M&S environment based on dynamical Systems Theory towards the pursuit of multi-paradigm modeling, a prime requirement for integrating multiple modeling architectures. The CyCAS sandbox based on such ideas will be reported in future work.

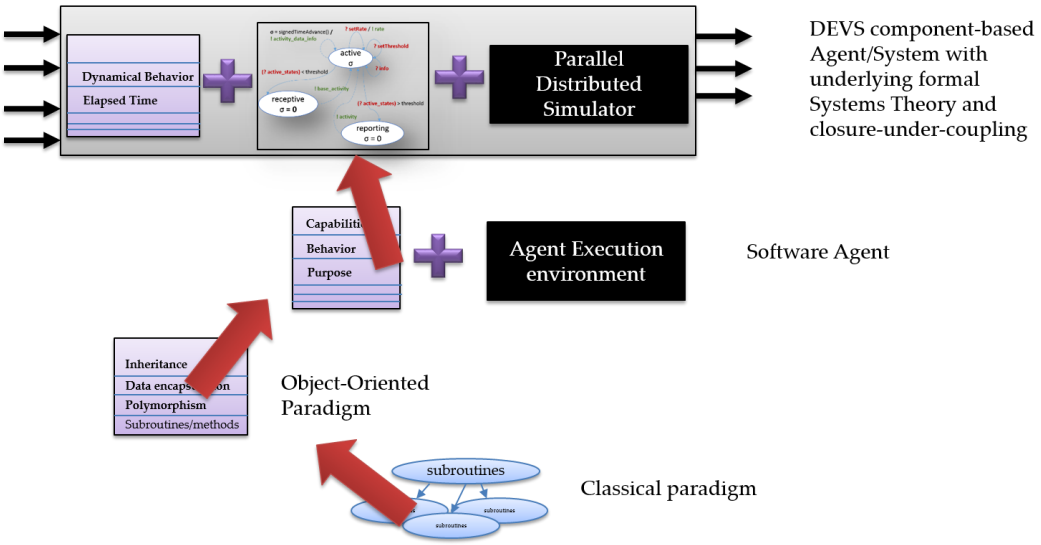

Figure 5: From a Software Agent to a DEVS Agent

Table 2: CyCAS and Model Engineering

CyCAS View	DEVS Element	Alternatives
Human	Dynamical agent behavior modeling in Atomic DEVS subject to SME validation	Cognitive agents, software agents, DEVS behavior models, SOAR agents
Multi-agent	Atomic, Coupled and Multi-Level DEVS	Object interaction diagrams
Control and Communication	Levels of Systems specifications	Subsumption architectures
Resource and Constraints	Atomic DEVS and network-delay models	External systems with defined I/O interfaces, constraint graphs
Emergence	Atomic observers at higher levels of hierarchy, Experimental Frames	Pattern matchers
Phase Transition	Atomic observers at multiple levels of system specifications, Experimental Frames	Pattern matchers
Knowledge	Representation through pragmatic frames within the System Entity Structure (SES) theory	Triple stores, RDF stores, etc.
Resilience	Experimental Frame, Atomic DEVS	-

8. CONCLUSIONS

Many enterprise architecture frameworks provide various views and perspectives to facilitate architecture development and capture the true purpose behind the enterprise architecture. Enterprise IT Systems underwent a lot of transformation in the last decade and robust technologies exist that deliver scalable enterprise solutions. Multiple enterprise architecture frameworks exists that serve different niche and efforts are underway to consolidate them towards a unified framework and none mandate the formal M&S engineering methodology to develop an executable architecture. Enterprise systems modeling is a different problem altogether and new approaches are needed to model such complex multi-faceted systems. Cyber complex adaptive systems are specialized cases of CAS in the cyber domain where both the systems and the system users adapt in unknown ways. Consequently, it is impossible to conduct studies before-hand as both sides of the equation are in constant flux. In this article, I introduced the CyCAS concept, essential elements of CyCAS and various perspectives that are needed to understand it. The CyCAS perspectives can be used in conjunction with the existing architecture frameworks and as of this article, they are by no means the only perspectives that are needed to understand the holistic behavior of CyCAS. I also introduced the sandbox requirements for modeling and simulation of CyCAS and related it to the fundamental Systems Theory. I also stressed the importance of using frameworks like DEVS Unified Process that is based on formal discrete event systems (DEVS) theory to be applied to CyCAS M&S. Further details will be presented in the future work.

REFERENCES

Bond, A., L. Gasser, 1988. A Survey of Distributed Artificial Intelligence., Available at: http://www.exso.com/nsurvo.pdf, Last accessed: July 31, 2014

CIO, DoD, The DoDAF Architecture Version 2.02, Available at: http://dodcio.defense.gov/TodayinCIO/DoDArchit

ectureFramework.aspx, Last accessed, July 31, 2014

DNDAF, 2010. DND/CF Architecture Framework (DNDAF) Version 1.7 Home Page. Available at: http://trak-community.org/index.php/wiki/DNDAF, Last accessed: July 31, 2014

Edgren, M. 2012. Joint M&S Strategy, WJTSC M&S Working Group, Defense Technical Information Center. Available at: http://www.dtic.mil/doctrine/training/conferences/wjtsc12_2/wjtsc12_2_jmsc_msstrategy.pdf, Last accessed July 22, 2014

FEAF, 2012. The Common Approach to Federal Enterprise Architecture. Available at: http://www.whitehouse.gov/sites/default/files/omb/assets/egov_docs/common_approach_to_federal_ea.pdf, Last accessed: July 31, 2014

Gorman, P., K. Haperen, I. Bailey, 2013. An Update on the Convergence of MOD and NATO Architecture Frameworks, Available at: http://nafdocs.org/wp-content/uploads/2013/07/Gorman-Bailey-Van-Haperen-IEA14-v2.pdf, Last accessed: July 31, 2014

Heylighen, F., and C. Joslyn, 2001. Cybernetics and Second Order Cybernetics, in: R. A. Meyers (ed.), *Encyclopedia of Physical Science & Technology*, **4**, 155-170, Academic Press, New York

Irwin, T., 2012, Enabling Training Technologies for Joint Force 2020, Available at: http://www.ndia.org/Resources/OnlineProceedings/Documents/21M0/Joint_Forces_2020/Intro_and_Overview_Thomas_Irwin.pdf, Last accessed, July 22, 2014

Laird, J. 2012, *The SOAR Cognitive Architecture*, MIT Press, Cambridge, MA

Maier, M.W. 1998. "Architecting Principles for Systems-of-Systems." *Systems Engineering* 1 (4): 267-284.

Mittal, S., E. Mak, and J.J. Nutaro. 2006. "DEVS-Based Dynamic Model Reconfiguration and Simulation Control in the Enhanced doDAF Design Process." *Journal of Defense Modeling and Simulation* 3 (4).

Mittal, S., B.P. Zeigler, J.L.R Martin, F. Sahin, and M. Jamshidi. 2008. "Modeling and Simulation for System of Systems Engineering." In *System of Systems Engineering for 21st Century*, by M. (ed.) Jamshidi. Wiley.

Mittal, S., 2013. Emergence in Stigmergic and Complex Adaptive Systems, *Journal of Cognitive Systems Research*, 21, 26-39.

Mittal, S., and J. L. R. Martin. 2013a. *Netcentric System of Systems with DEVS Unified Process.* Boca Raton, FL: CRC Press.

Mittal, S., and J. L. R. Martin. 2013b. Model-driven Systems Engineering in a Netcentric Environment with DEVS Unified Process, Winter Simulation Conference

Mittal, S., and B. P. Zeigler, 2014. Context and Attention in Activity-based Intelligent Systems", ITM Web of Science, 2.

MoD, UK. 2012. MOD Architecture Framework. Available at: https://www.gov.uk/mod-architecture-framework, Last accessed: July 31, 2014

NAF, 2013. NATO Architecture Framework: MODEM Review. Available at: http://nafdocs.org/wp-content/uploads/2013/07/20130816_MODEM_to_NAF_Review_V1_1-U.pdf, Last accessed: July 31, 2014

Rainey, L., S. Mittal, eds., 2014. Agent-based modeling for understanding complexity: Lessons learned from defense domain where lack of adherence to Systems theory lead to unpredictable behavior, Special Issue Journal of Defense Modeling and Simulation, Call for Papers. SCS.

Taylor, H., A., Yochem, L. Phillips, F. Martinez, 2009. *Event-Driven Architecture: How SOA enables the real-time enterprise*, Boston, MA., Addison-Wesley

TOG 2012. An Introduction to ToGAF 9.1, The Open Group. Available at: https://www2.opengroup.org/ogsys/catalog/w118, Last accessed: July 31, 2014

Tolk, A., and S. Mittal, 2014 "A Necessary Paradigm Change to Enable Composable Cloud-based M&S Services", Winter Simulation Conference, Savannah, GA, USA

Weiss, G., (ed.) 2000. Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, MIT Press, Cambridge, MA

Weter, D., 2012, Joint Staff J7 Modeling & Simulation Program Update, available at: http://www.rolands.com/jtls/j_iuc2012/JSJ7_Weter.pdf, last accessed: July 22, 2014

Zachmann, J.A., 2008. The Zachmann Framework Evolution, Available at: https://www.zachman.com/ea-articles-reference/54-the-zachman-framework-evolution, Last accessed: July 31, 2014

Zeigler, B.P. 1976. *Theory of modeling and simulation.* Wiley Interscience.

Zeigler, B.P., H. Praehofer, and T.G. Kim. 2000. *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems.* New York, NY: Academic Press.

Zeigler, B. P. 2013. "Grand Challenges in Modeling and Simulation, What can DEVS Theory do to meet them? Parts 1 and 2" *Seminar to School of Automation Science and Electrical Engineering*, Beihang University, Beijing, China.

BIOGRAPHY

SAURABH MITTAL is the founder and president of Dunip Technologies LLC, CO USA. He earned his MS (2003) and PhD (2007) in Electrical and Computer Engineering from the University of Arizona, Tucson. His current interests include modeling and simulation of complex natural systems, distributed artificial intelligence, multi-agent systems and complex adaptive systems. He can be reached at smittal@duniptech.com