20th ICCRTS

EVOLUTIONARY DEVELOPMENT OF COMMAND AND CONTROL SYSTEMS OF SYSTEMS SIMULATIONS

Topics:

Topic 9 - C2-simulation Interoperability (Primary)

Topic 5 – Modeling and Simulation (Alt)

Topic 4 – Experimentation, Metrics, and Analysis (Alt)

Topic 3 - Data, Information, and Knowledge (Alt)

Authors:

Mary Ann Cummings (POC)

Naval Surface Warfare Center, Dahlgren Division Dahlgren, VA 22448

Mary.cummings2@navy.mil

540-653-5419

Clint Winfrey

Naval Surface Warfare Center, Dahlgren Division

Dahlgren, VA 22448

Clint.winfrey@navy.mil

ABSTRACT:

This paper describes an M&S software framework that allows the evolutionary building of weapon SoS simulations by allowing the adding, changing, and removing of models (C2 and/or component weapon systems) without making changes to the rest of the simulation. Using this framework, we illustrate how C2 can be simulated in a weapon SoS and show how it can be virtually tested with the component weapon systems. This framework uses Dr. Zeigler's (University of Arizona) work in M&S theoretical frameworks as a basis for creating a software framework that separates the Experimental Frame (objectives of the simulation) and the Simulator from the models themselves. This same Simulator and/or Experimental Frame can be used in multiple SoS simulations without change. The Simulator and Experimental Frame are reusable and swappable. This means that the Simulator can be reused for unrelated projects. In addition, the Simulator can be exchanged for another without having to change the models. This framework allows the development of these models from multiple organizations to seamlessly work together. It allows these models to be reused (not recompiled with modifications) as new simulators and experimental frames are built that use these models to develop other weapon SoS's.

EVOLUTIONARY DEVELOPMENT OF COMMAND AND CONTROL SYSTEMS OF SYSTEMS SIMULATIONS

Mary Ann Cummings
Naval Surface Warfare Center, Dahlgren Division
Dahlgren, VA 22448
Mary.cummings2@navy.mil
Clint Winfrey
Naval Surface Warfare Center, Dahlgren Division
Dahlgren, VA 22448
Clint.winfrey@navy.mil

ABSTRACT

This paper describes an M&S software framework that allows the evolutionary building of weapon SoS simulations by allowing the adding, changing, and removing of models (C2 and/or component weapon systems) without making changes to the rest of the simulation. Using this framework, we illustrate how C2 can be simulated in a weapon SoS and show how it can be virtually tested with the component weapon systems. This framework uses Dr. Zeigler's (University of Arizona) work in M&S theoretical frameworks as a basis for creating a software framework that separates the Experimental Frame (objectives of the simulation) and the Simulator from the models themselves. This same Simulator and/or Experimental Frame can be used in multiple SoS simulations without change. The Simulator and Experimental Frame are reusable and swappable. This means that the Simulator can be reused for unrelated projects. In addition, the Simulator can be exchanged for another without having to change the models. This framework allows the development of these models from multiple organizations to seamlessly work together. It allows these models to be reused (not recompiled with modifications) as new simulators and experimental frames are built that use these models to develop other weapon SoS's.

Keywords: System of Systems (SoS), framework, Simulator, Experimental Frame

INTRODUCTION

We live in a complicated world - a place where simple systems are no longer the norm. Our weapon systems and the warfighter capabilities they provide are also complicated. Today's weapon systems have become very complex, and a growing number of military capabilities are achieved through a System of Systems (SoS) approach.[5] Many of these SoS's incorporate formerly standalone legacy weapon systems (with their own purposes) with new developments to create a more powerful weapon system of systems. All of these weapon SoS's have a Command and Control (C2) component. For example, the U.S. Navy's Aegis Combat System (ACS) and the U.S. Army's Terminal High Altitude Area Defense (THAAD) have been integrated with the newly developed Command, Control, Battle Management and Communications (C2BMC) system to become part of a system of systems known as the Ballistic Missile Defense System (BMDS), shown in Figure 1. The mission of the BMDS is an integrated, layered defense system to defend the United States, its deployed forces, and allies against all ranges of enemy ballistic missiles in all phases of flight.

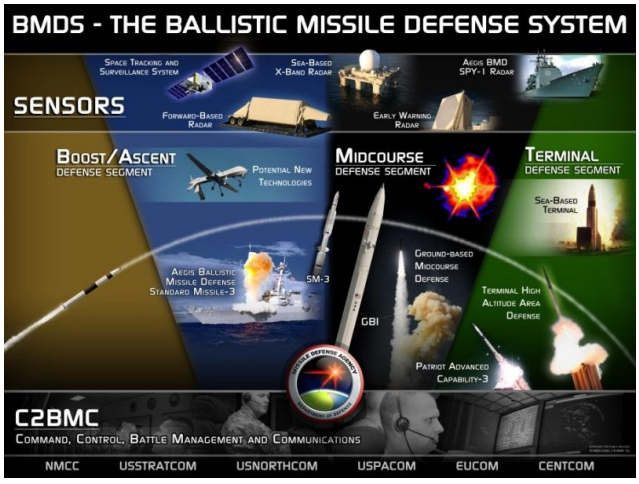

Figure 1: Ballistic Missile Defense System Elements

In order to develop these complicated systems, the Policy for Systems Engineering in the Department of Defense (DoD) [4], issued by the Under Secretary of Defense for Acquisition, Technology & Logistics in 2004, calls for the application of a rigorous systems engineering discipline that meets the challenge of developing and maintaining needed warfighting capability through the integration of complex systems in SoS's. This discipline calls for the use of capabilities-based analysis, which has become the basis for defining the warfighter's (also known as the user) needs. A key goal of engineering an SoS is to specify the value-added capabilities and ensure that these capabilities are correctly implemented. This may force the previously stand-alone systems to provide functionality or interfaces that had not been considered in their individual designs [6]. In order to validate an SoS, the complex behaviors resulting from the interactions between the component systems must be known, tested and accepted.

Modeling and Simulation (M&S) is an affordable validation alternative to flight testing all aspects and interfaces in a weapon SoS. The Department of Defense Acquisition Modeling and Simulation Master Plan calls for the use of M&S in defining, developing, testing, producing and sustaining warfighting capability that supports the spectrum of DoD missions. The use of M&S can be thought of as the virtual test for SoS's, thus reducing the need for flight testing these weapons in all possible configurations. With changes in a few parameters, M&S can be used to test many configurations (that are cost prohibitive to test via flight tests). Through this virtual testing, we are also able to identify behavior that emerges from the interactions among the component systems in the simulation, termed as emergent behavior. This virtual testing is not a replacement for flight tests, but it does reduce the number of flight tests needed. If done properly, the flight tests "feed" data to the M&S, and the M&S shape the flight tests needed to reduce the overall cost.

Emergent behavior in an SoS [3] is defined as a behavior or pattern that occurs as a result of interactions of its components (or parts). A defining characteristic is that the behavior does not appear within one component alone; it only appears in the interactions. It allows the SoS to be greater than the sum of its components. Examples of emergent behavior are:

- The working of the human mind which emerges from neurons collaborating together
- The flocking of birds
- Schools of fish

Emergent behavior is contrasted with resultant behavior. This resultant behavior is one in which it is the sum of the behaviors of the components instead of being greater than the sum of the components (emergent). An example of a resultant behavior is an increase in sales resulting in an increase in profits.

In [16], the authors describe the need for a never ending M&S process (as shown in Figure 2) that ties the models, simulation and real system together. The process is driven by objectives being generated from outside the system boundaries. These objectives do not happen just once, but appear and change over time. As new objectives come in, the modeling activity is started again. The available knowledge for building the model comes from the model base. Model construction is followed by the simulation of the model. The data used in simulation comes from the database which stores and organizes data gathered about the real system. The validation of the model and simulation leads to new experimentation on the real system (otherwise known as flight testing). As the process is traversed, refined objectives may be formulated as deficiencies in the current knowledge base become more apparent.

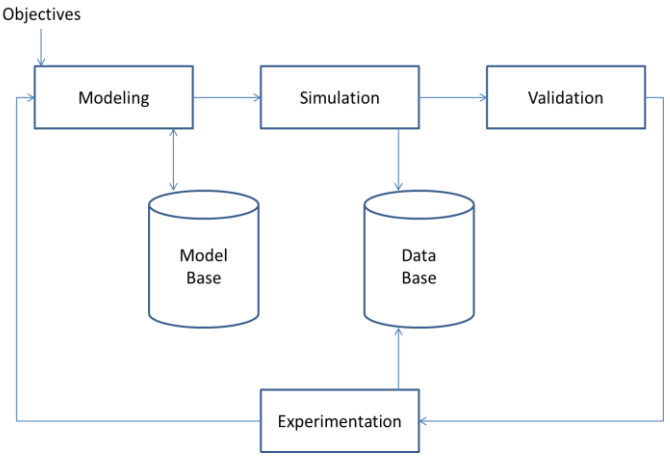

Figure 2: Modeling and Simulation Process

According to [13], the term "modeling" in M&S is the act of developing a static representation of some real world object. A model can also be thought of as a well defined representation of some real world object. The term "simulation" is the execution of models over time representing the attributes of one or more entities or processes. Most often, simulation is performed by a computer. Thus, simulation is facilitated by advances in system engineering and software agents, among other things. According to [13], the purpose of M&S in dealing with complexity is not to predict the outcome of a system necessarily, but rather it is to reveal and understand the complex and aggregate system behaviors that emerge from the interactions of elements involved.

Figure 3 shows that the simulation plays a vital (circular) role in the development of a system. According to [14] the simulation is driven by data taken from the real system under control and needs to meet the requirements of the system.

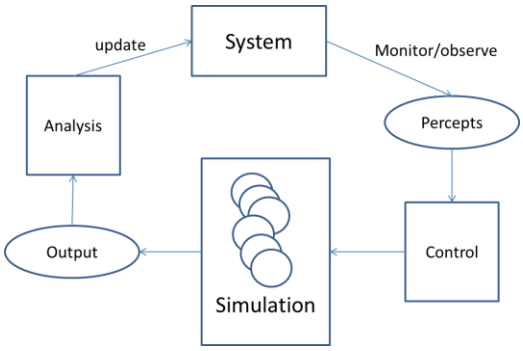

Figure 3: Circular role of simulation in system development

MOTIVATION

Today, many weapon System of Systems, such as the BMDS, have a large set of M&S developed by different organizations that define the different component weapon systems. They also have a set of M&S that provide the environments, threats, C2, debris, etc. They then tie these together with a framework. Many of the M&S have their own event drivers and timing. Because of these limitations, a scenario (objectives of the M&S) is defined for each component M&S and most are executed independently with the output data saved off. The scenario for the overall SoS is then defined. The execution of the SoS simulation is the setting up of the scenario, reading in and manipulating the output of each of the proprietary simulations, and running the simulations that are not proprietary. The problems with this are:

- Every time any of the component M&S systems change, any M&S that interface with that component also have to change.
- Any new component M&S will cause the other M&S and the framework to change.
- Because the outputs are different for every M&S, the framework has to change every time any of the component M&S change inputs or outputs.
- Any scenario changes cause changes to all M&S and the framework.

All of this results in high costs and change in schedules for any changes or additions to this SoS framework.

EXISTING SOLUTIONS

Is there an M&S methodology that can address these problems? Dr. Bernard Zeigler, in [15], describes a theory that has brought coherence and unity to M&S. This was the first work to describe approaches to M&S as system specification formalisms. [17] defines the specification of a set of objects in an unambiguous manner. Although it has been over 30 years since this work was distributed (and two update editions have come out since then), this work on formalisms and frameworks (note that this is a theoretical framework and not a software framework) is still heavily influencing M&S today The formalisms described in this work are:

- Discrete Event System Specification (DEVS)
- Discrete Time System Specification (DTSS)
- Differential Equation System Specification (DESS), also known as continuous time

In [17], the author provides the definition of a formalism as a convention in communication that specifies a class of objects under discussion in an unambiguous and general manner. It allows us to focus on features of objects which are relevant to our discussion. This process is called abstraction. Formalisms can be hard to grasp because the abstraction takes us out of the concrete world.

A formalism contains a list of parameters which are a set of theoretic constructs and a list of constraints. If an object is in the class described by the formalism then it has a set of parameters which each take an assignment of values which satisfy the constraints. The structure of the formalism refers to its parameters and constraints.

Dr. Zeigler's work, in [17], describes the basic objects of the M&S theoretical framework as the source (or real) system, the model, the Simulator and the Experimental Frame. The source system provides the data we want to observe (otherwise known as the system we want to model). To observe it, we must determine the conditions we want to observe in the real system. These conditions are specified in the Experimental Frame, which documents the objectives of the model we want to build. A model is some representation of the real system. The model has the ability to execute instructions and generate behavior. The Simulator triggers these instruction executions and behavior through time. Figure 4 shows the basic M&S framework objects and their relationships as proposed in [17].

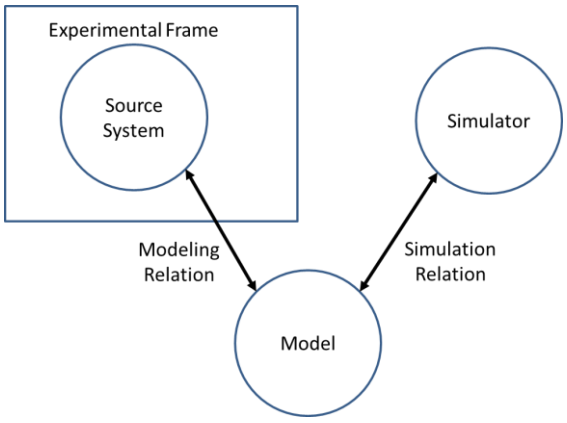

Figure 4: M&S theoretical framework entities and their relationships

[17] also states that there is one Simulator per model (or class of models). This is one of the elements that makes this M&S theory stand out because it pulls the event driver and timing out of the models. This helps the current M&S situation described above, because it allows the overall SoS M&S engineer to build the Simulators while understanding the overall purpose/function of the SoS M&S. It also removes the need to build event drivers and timing into each separate component M&S system.

Although this is a great improvement, this Simulator is tied to the model (or class of models). This means that there are unique Simulators for each class of models. For a weapon SoS M&S, this can mean a Simulator for every model (which could include weapon models, environment models, and threat models) because one can expect each component M&S system to be a different class of models. Taken further, this also means that the model may have to change when the Simulator changes.

[17] states that the Experimental Frame is the operational formulation of the objectives (resulting in the scenario) that drive a particular M&S. This helps the situation above by allowing the development of one Experimental Frame (holder of the scenario), built by the SoS M&S engineer, to be used by all of the component M&S systems instead of having a separate scenario for each M&S.

This means that there is a unique Experimental Frame for each M&S. Also, there is a unique Experimental Frame for each objective or purpose of a single M&S. In a weapon SoS, this could mean a unique Experimental Frame for testing the range of the weapons, for measuring the performance of the weapons against a threat, and for analyzing environment changes on the weapons. According to [17], the Experimental Frame is the formulation of the objectives that motivate the M&S project. This indicates that the model and Simulator may change if the objectives change.

Taken together, for one System of Systems M&S (and this is true for our example above), we could have many Simulators and many Experimental Frames. For example, for a BMDS simulation with twelve weapon and satellite models, we could have twelve Simulators and fifty or more Experimental Frames to define different simulation objectives. In addition, we may have many variations of a model based on the simulation and the objectives used. If we look at multiple System of Systems M&S's that an engineer may build to perform a virtual test of all the configurations of its system of systems, we may end up with many Simulators and many Experimental Frames that might be slightly different or may be vastly different. We may also end up with many variations of the models.

PROPOSED SOLUTION

Is it possible to look at reusing the Simulator and Experimental Frames such that we only need one Simulator to drive all the models together and one Experimental Frame that allow the user to make parameter changes to the models over many configurations? [2] points out that the goal of effective reuse for both simulations and other software is connected to the adaptation of components to fit a new scenario. If we do not have to change the Simulator and Experimental Frames, then the adaptation is left to the models. This calls for software engineering techniques that are specific to M&S.

Let us take it a step further. Using the definition of "swappable" to mean the ability to exchange one item for another, is it possible to create swappable Simulators and swappable Experimental Frames so that the models do not have to change when the Simulator and/or the objectives of the model change? The work described in this paper strives to design the Experimental Frame and the Simulator (to include their interfaces) such that the Simulator and/or the Experimental Frame can be chosen at runtime by the user to allow multiple objectives in one execution of the SoS M&S.

This research seeks to design a Simulator Exchange that allows one Simulator per simulation method (DEVS, DTSS, DESS) to be used by all SoS M&S in order to provide maximum reusability for all SoS simulations. In addition, this research seeks to design an interface to this Simulator such that Simulators are swappable. This means that we are not just tied to the three simulation types listed in [17]. A programmer can create other types, such as human in the loop, etc., that have not yet been identified without changing the other elements of the M&S. We will call this the Simulator Exchange, since it can be swapped or exchanged with other Simulators without causing change to the other elements of the M&S.

Because we can do this same exchange method with the Experimental Frames, we will seek to design an Experimental Frame Exchange that allows the reuse of an Experimental Frame to be used by all SoS M&S. We will also allow these Experimental Frames to be swappable. We will call this set of Experimental Frames the Experimental Frame Exchange. This work also seeks to design this swappable Experimental Frame Exchange in such a way that any objective can be chosen at runtime for an SoS M&S without changing any of the other elements in the simulation.

Because we can now have multiple models interacting together and driven by one Simulator, we will call the collection of models the Component System Model Collection.

We believe that this research can create exchanges that will allow for a metrics collector that will interface with the Simulator and Experimental Frame in such a way that it can allow for the identification of emergent behavior among the interactions of the models (component systems). For the purpose of this paper, emergent behavior will be defined as

behavior of one component that is affected by another component.

Let's look at how this helps BMDS. The Simulator Exchange will allow the development of a BMDS simulation that will have one DEVS Simulator and one DTSS Simulator that can be interchanged without changing any of the weapon models. This interchange will allow one BMDS simulation to be driven by both Discrete Events and Discrete Time without any changes to the other elements. The Experimental Frame Exchange will allow the selection of multiple objectives for the BMDS simulation without changes to the other elements.

Figure 5 is a depiction of how the experimental exchange, the Simulator Exchange, the metrics collector and models all interface together to produce an SoS simulation.

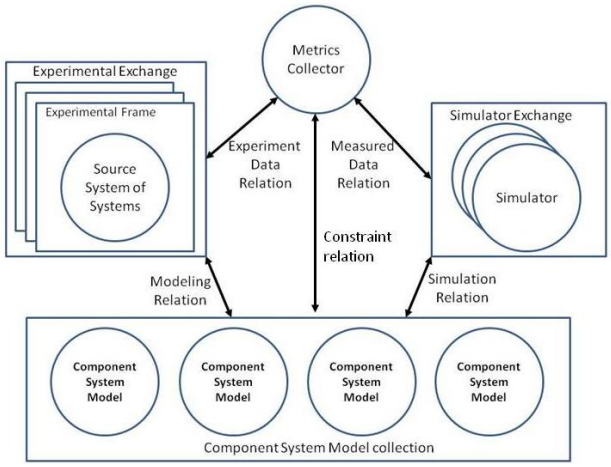

Figure 5: Modified M&S theoretical framework entities and their relationships

FRAMEWORK

In order to create simulations that can use reusable and swappable elements, a framework must be identified that was built using an evolutionary architecture methodology. [8] defines evolution as the change over time that occurs in the SoS as component systems are added, removed, or replaced.

[8] identifies two principles that must be met in order to allow the SoS to evolve as the component systems are added, removed or replaced. These are:

- The complexity of the system of systems framework does not grow as constituent systems are added, removed, or replaced.
- The constituent systems do not need to be re-engineered as other constituent systems are added, removed, or replaced.

In order to meet these two principles, the SoS framework must have standard interfaces and interface layers. The standard interfaces and interface layers are to ensure that the component systems do not have to be changed as components are replaced, added or removed.

Our proposed framework uses the Orchestrated Simulation Through Modeling (OSM) framework because it meets the two principles identified above. [12] shows that the OSM framework allows Discrete Event System Specification (DEVS) modeling and simulation (M&S) frames to be developed separately as plug-ins and combined with output visualization plug-ins and model plug-ins to form a complete simulation. OSM follows Dr. Zeigler's theoretical formalism [17] to separate the Experimental Frame, Simulator and models. OSM allows input plug-ins (model, experimental), execution plug-ins (Simulator), and output plug-ins to be developed separately and put together to form a unique system while allowing development of these elements separately (even by different organizations).

The OSM framework is a government-owned M&S framework that was developed by Naval Surface Warfare Center Dahlgren Division in the Strategic Software department. This framework is "model agnostic" and enables the integration and execution of multiple independently developed models. It uses "plug-ins" as a method of putting all the components together. It facilitates the development of a rich set of re-usable plug-ins by providing examples of plug-ins already developed as part of its developer's toolkit (can be thought of as a user's guide to developing an SoS simulation). It can be used for any type of M&S project because of its model agnostic methodology. It allows the sharing and tying together of models from various organizations to create a system of systems simulation. It allows the development of agents from scratch or can import existing models if the models are "wrapped" to produce agents from the inputs/outputs of the models. This works best if the models were developed to interact or provide data throughout the simulation.

Because of the flexibility of this framework and its foundation built on Dr. Zeigler's theoretical framework, it is a good choice for the addition of swappable Simulators and swappable Experimental Frames.

COMPONENTS AND CONNECTORS ARCHITECTURAL STYLE

In order to develop swappable Simulators, a software architectural style must be chosen that allow the seamless change of these elements without changing the rest of the elements in the simulation. The Components and Connectors (C2) architectural style provides interoperability between components of different types (i.e., developed with different programming languages). [9] shows that it was created to allow applications to work with new platforms by extracting the components from calls to platform services such as operating systems and user

interfaces from the body of the components that need these services. It also allows for substituting one component for another to achieve applications that are similar but still different.

Applications which use the C2 style are layered networks of concurrent components connected by message routing connectors. In this style, there is no direct component to component communication. It only allows the connections of components to connectors. This enforces layering which promotes the ease of modifying an application to work with different platforms, and promotes the ability to swap components. C2 uses domain translation to make this happen. Domain translation is the transformation of requests from a component into a form that is understood by the recipient of that request. This principle is also used for the notifications provided by a component.

The C2 style manages the flow of control and flow of data for an application. The flow is managed by using one or more channels to link the interacting components. This control and data flow will be used to determine/modify the timing of an event (convert the event to a timed event) being sent to the event driver of the Simulator.

The C2 style provides conversion services to transform the data involved in an interaction to be in a format required by another component. This type of connectors is called adaptor connectors. In our case, we convert discrete events to discrete timed events.

We use the C2 style to provide a Simulator that can act either as a DEVS Simulator or a DTSS Simulator without changing the models themselves, as described in Figure 6. The benefit of extracting components from calls to platform services allow us to switch between various simulation types without having to change the model itself. The benefit of substituting one component for another is the feature that allows us to develop a Simulator that can convert between Discrete Events and Discrete Time without changing the models themselves.

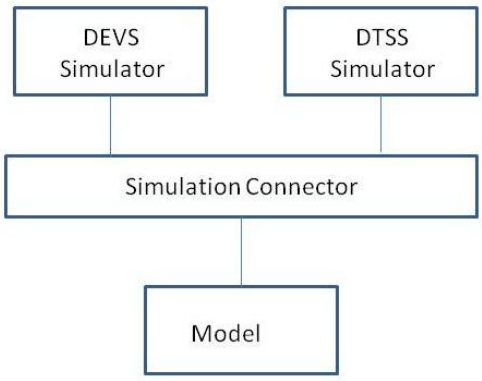

Figure 6: Simulator Connector Using C2 Style

THE METHODOLOGY

Using Zeigler's theoretical framework, we separate the simulation into an Experimental Frame, a Simulator and a set of models. We extend this further and represent each model as an agent. We are able to do this because Dr. Zeigler allows the models to be represented by a hierarchy of atomic models coupled together.

In order to provide a way to identify emergent behavior, we create a metrics collector, which interfaces with the Experimental Frame and the Simulator, to collect data to allow us to identify emergent behavior. We define emergent behavior to be that behavior that comes about as one component that is affected by another component. Figure 6 shows the Experimental Frame, Simulator, and Metrics Collector connected to the agents (models).

SWAPPABLE EXPERIMENTAL FRAMES

Through the framework described in this paper, we developed a methodology for the swappable Experimental Frames by developing three Experimental Frames that can be used in multiple simulations and can be swapped in the same simulation. These are:

- Single run Experimental Frame
  - This allows the setting of input parameters without any randomization. See Figure 7 for an example of this.
- Lattice
  - This provides the setting of 1 or 2 parameters to run through a set of numbers in between 2 limits. This type of panel would allow the user to set the number of runs based on the number of values in the input set. An example of this is as follows:
    - variable = Lower limit 1, upper limit 4 – runs 4 times with variable =1, variable = 2, …
  - See Figure 8 for an example that picks a random number between a lower and upper limit for a set number of runs

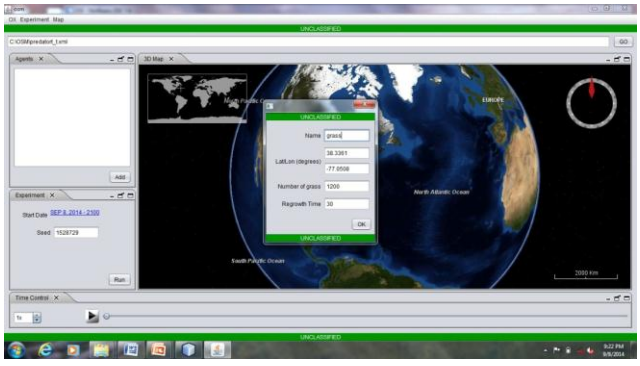

Figure 7: Example of a Single Run Experimental

- Stochastic

- This allows the setting of one parameter using a random number between a lower and upper limit for a set number of runs. An example of this is as follows:
    - variable = 5, + or – 2 – 10 runs; variable is set to some random number between 3 and 7 for 10 runs

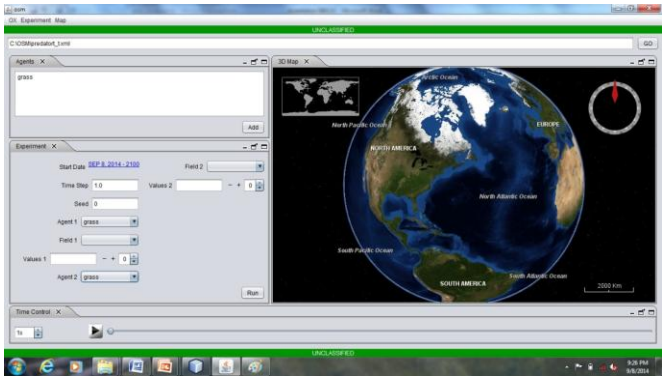

Figure 8: Example of a Lattice Experimental Frame Input Panel

INTENT OF SWAPPABLE EXPERIMENTAL FRAMES

The intent of this work is to provide a methodology for developing an Experimental Frame, as called out in Dr. Zeigler's book that is swappable. We call this an Experimental Frame Exchange. Let's first define what an Experimental Frame is. It defines the objectives of the simulation. For a SoS simulation, it must define these objectives such that they encompass all of the component systems.

The term "swappable" means to trade or exchange something for another. In the case of an Experimental Frame, this means to be able to trade or exchange the objectives of a model without changing any of the other elements of the simulation. This means that the Simulator and the models do not change as a result of swapping or exchanging one Experimental Frame for another.

MOTIVATION FOR SWAPPABLE EXPERIMENTAL FRAMES

As we described earlier, many M&S are being written for users where every time the objectives of the simulation change, the simulation has to be rewritten in some areas or the whole simulation has to be rewritten. We believe that simulation objectives can be defined in such a way that the models and the Simulator do not have to change. In addition, when multiple simulations are written by an organization, they should not have to recreate an Experimental Frame.

This methodology should be used whenever more than one simulation is written by an organization. It should also be used whenever a simulation may have multiple objectives.

SWAPPABLE SIMULATORS

Through the research described in this paper, we are developing a methodology of swappable Simulators by developing Simulators that can be used in multiple simulations and can be swapped in the same simulation. These are:

- Discrete Event System Specification (DEVS)
- Discrete Time System Specification (DTSS)
- For both DEVS and DTSS Simulators, the models will not change. Therefore, the user can select (through the Experimental Frame) the simulation type, and get possibly different results without changing the rest of the simulation modules.

INTENT OF SWAPPABLE SIMULATORS

The intent of this methodology is to provide a methodology for developing a Simulator, as called out in Dr. Zeigler's book, that is swappable. Let's first define what a Simulator is. A Simulator executes a model through time.

In the case of a Simulator, this means to be able to use a different simulation algorithm (or simulation type) without changing any of the other elements of the simulation. This means that the Experimental Frame and the models do not change as a result of swapping or exchanging one Simulator for another.

MOTIVATION FOR SWAPPABLE SIMULATORS

As we described earlier, many M&S are being written for users where the Simulator is either embedded in the model or the Simulator is written to drive one model or one class of models. We believe that Simulators for a particular simulation algorithm or type can be written such that they can drive all models and they will not require changes to the models. This means that the models can be used without change for either Discrete Event, Discrete Time or Differential Equations simulations.

The motivation behind a swappable Simulator is that DEVS provides a computational basis for implementing behaviors that are expressed in the other basic systems formalisms (i.e., DTSS and DESS). [17]

PROOF OF CONCEPT

This research used swappable Simulators and Experimental Frames to develop a weapon system SoS simulation. This simulation is of the communications between all the different elements in a Ballistic Missile Defense theater raid scenario. In this simulation, there are the following agents:

- Threat missiles firing at a defended area
- Aegis Ships (each instance a separate agent)
- Weapon Control System (WCS) on each ship

- Vertical Launch System (VLS) for each missile on each ship
- Sensor (SPY1) on each ship
- Sensor (TPY2) looking for the threat missile during flight
- Sensor (SBIRS) search for threat missile right after launch
- C2BMC controlling all weapon systems

This simulation used a Simulator that allows events caused by some action or caused by time. We used both the multi run Experimental Frame and the Lattice Experimental Frame. Figure 9 shows the simulation with the multi run Experimental Frame. This figure shows the list of agents in a panel on the left with the Experimental Frame panel underneath it. The simulation visualization is shown to the right. The red lines show the threat trajectory; the blue lines show the interceptor (Aegis missile) trajectory, the yellow cones are the SPY1 sensor range, and the green cone is the TPY2 sensor range. All of these agents are controlled by messages sent from the C2BMC.

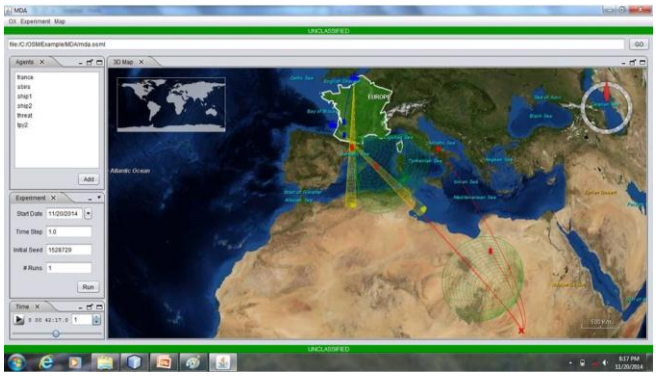

Figure 9: BMDS Communications Simulation with Multi Run Experimental Frame

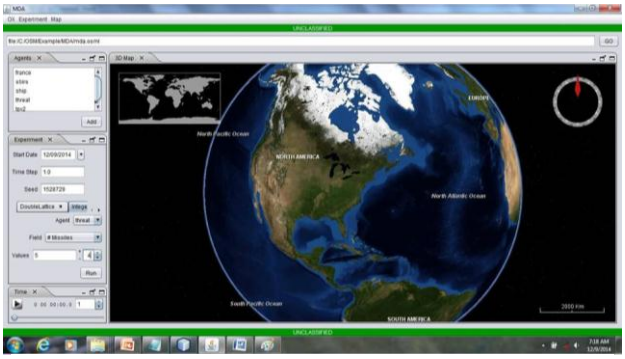

Figure 10: BMDS Communications Simulation with the Lattice Experimental Frame

CONCLUSION

Weapon systems (like the BMDS) have become very complicated and are no longer just one system, but a system of systems involving many component weapon systems operating together. All of these weapon SoS's are controled by a C2 component. These SoS's must be modeled and simulated before they are tested and prior to and during their operational deployment. In order to do this, we must have a Modeling and Simulation software framework that allows us to connect simulations of these component weapon systems together, without having to change the framework each time. We also do not want to change the component simulations every time we make a change to the objectives or to the simulator algorithms. To do this, we need swappable Experimental Frames and swappable Simulators that allow us to change the objectives (also can be thought of as scenarios or experiment data) and change the simulation types (for example, changing from Discrete Event to Discrete Time simulations) without changing the other elements in the simulation. Because of these swappable elements, we can now better identify emergent behavior of a System of Systems through metrics collection. This paper has described a framework and a method for providing these swappable components and metrics collection using a framework that is an extension of Dr. Zeigler's theoretical constructs. This paper has shown a proof of concept of these swappable components using a BMDS communication simulation where C2BMC controls the messages among the component weapon systems and sensors.

DISCLAIMER

The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements of the US government.

REFERENCES

1. Bajrachaya, K., & Duboz, R. (2013). Comparison o three agent-based platforms on the basis of a simple epidemiological model. *Proceedings of the Symposium on Theroy of Modeling & Simulation.*
2. Carnahan, J. C. (2005). Simulation-Specific Characteristics and software Reuse. *Proceedings of the Winter Simulation Conference '05.*
3. Chan, W. K. (2011). Interaction Metric of Emergent Behaviors in Agent-Based simulation. *Proceedings of the 2011 Winter Simulation Conference.*
4. Deputy Under Secretary of Defense (Acquisition, Technology & Logistics). (2004). *Policy for Systems Engineering in DoD.*
5. Director, Systems and Software Engineering and Deputy Under Secretary of Defense (Acquisition and Technology). (2008). *Systems Engineering Guide for*

Systems of Systems (Vol. 1.0). Washington DC: OSD(A&T)SSE.

6. Garrett, R., Baron, N., Anderson, S., & Moreland, J. (2011). Managing the Interstitials; A System of Systems Framework suited for the Ballistic Missile Defense System. *Systems Engineering* , 87-109.
7. Office of the Under Secretary of Defense (Acquisition, Technology and Logistics) Defense Systems. (2006). *Acquisition Modeling and Simulation Master Plan.*
8. Selberg, S., & Austin, M. (2008). Toward an Evolutionary System of Systems Architecture. *Proceedings of Eighteenth Annual International Symposium of The International Council on Systems Engineering (INCOSE).* Utrecht, The Netherlands.
9. Taylor, R. N., Medvidovic, N., & Dashofy, E. M. (2010). *Software Architecture: foundations, theory, and Practice.* Hoboken: John Wiley & Sons, Inc.
10. Wilensky, U. (1999). *http://ccl.northwestern.edu/netlogo*. Retrieved from Center for Connected Learning and computer-Based Modeling, Northwestern University, evanston, IL.
11. Winfrey, C., Baldwin, B., Cummings, M. A., & Ghosh, P. (2014). OSM: An Evolutionary System of Systems Framework for Modeling and Simulation. *Proceedings of the Spring Simulation Multi-Conference.* Tampa.
12. 11. Wilensky, U. (1997). *NetLogo Wolf Sheep Predation model*. Retrieved from http://ccl.northwestern.edu/netlogo/models/WolfSheepPredation.
13. Yilmaz, L., & Oren, T. (2007). Agent-Directed Simulation and Systems Engineering. *Proceedings of the 2007 Summer Computer Simulation Conference.*
14. Yilmaz, L., & Oren, T. (2004). On the Need for Contextualized Introspective Modelsto Improve Reuse and Composability of Defense Simulations. *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology* .
15. Zeigler, B. (1976). *Theory of Modeling and Simulation.* John Wiley.
16. Zeigler, B. (1984). *Multifacetted Modeling and discrete Event Simulation.* London: Academic Press.
17. Zeigler, B. (2000). *Theory of Modeling and Simulation.* John Wiley.