How Does an Analyst Select M&S to Support the Entire DoD Acquisition Lifecycle Process? Examine ARL's Executable Architecture Systems Engineering (EASE) Research Effort

Christopher McGroarty
US Army Research Laboratory Human Research and Engineering Directorate Simulation and Training Technology Center
Orlando, FL
USA
christopher.j.mcgroarty.civ@mail.mil

Christopher J. Metevier
US Army Research Laboratory Human Research and Engineering Directorate Simulation and Training Technology Center
Orlando, FL
USA
christopher.j.metevier.civ@mail.mil

Scott Gallant
Effective Applications Corporation
Orlando, FL
USA
scott@effectiveapplications.com

Lana McGlynn
McGlynn Consulting Group
Cary, NC
USA
lana.mcglynn@gmail.com

Joseph S. McDonnell, PhD
Dynamic Animation Systems, Inc.
Fairfax, VA
USA
joe.mcdonnell@d-a-s.com

1.0 INTRODUCTION

Modeling and Simulation (M&S) users who require complex M&S typically do not have a long lifecycle for an experiment, analysis initiative or simulation-based event. To reduce cost, they need to use well-established simulation architectures and robust models that are easy to integrate with other simulations. This desire for a short lead time for system design, development, integration, execution and data analysis forces the system definition and design to happen very quickly.

In addition to having limited time and financial resources, analysts are being forced to address ever increasingly multifaceted problems. These problems require resources far beyond the simple spreadsheets of the past. With the advent of multicore desktop computers, cloud architectures and data mining tools, analysts have the opportunity to leverage vast amounts of data in order to conduct their analyses. But manipulating output data is not the same as analyzing data. Truly analyzing data requires understanding the linkages among the input data, the design assumptions and the intricacies of the systems producing the data.

The United States (US) Army Research Laboratory (ARL) has developed tools and processes that will help M&S users with their goals of understanding the simulation capabilities that are available and executing complex M&S environments as needed rather than when technical staff is available. A description of the users' needs will provide the context of our efforts.

2.0 NEEDS OF THE USER

The majority of analysts will agree that there never seems to be enough time when preparing for an experiment, test, analysis initiative or simulation-based event. A long planning cycle is a luxury they are not afforded. The analysts desire the ability to obtain key information in an effortless manner and to be able to employ tools that do not require a steep learning curve. Ultimately, the analysts want to spend more time examining the findings and less time learning to utilize the simulation tools.

There is seldom a single simulation that will accomplish the analysts' goals on its own; rather engineers will integrate multiple systems together. Each system represents specific aspects of the synthetic environment being used. These M&S users rely on standards and simulation developers to get the systems to communicate using the same syntax. This often works to instantiate a System of Systems (SoS) architecture [1] and to get models to share information. A SoS environment is an assembly of applications that together provide more capability than the sum of their individual capabilities. Within the M&S community, the applications assembled are each focused on representing a specific warfare function (or functions) based on data and models from an organization considered to be the center of excellence for that aspect of warfare. The SoS architecture provides many benefits when compared to executing a single monolithic model, including performance, model management and information transparency for analysis.

The United States Department of Defense (DoD) acquisition community is focused on creating viable materiel solutions. Figure 1 shows the DoD Acquisition Life Cycle [2]. While a formal Materiel Solutions Analysis occurs prior to Milestone A, a Project Manager (PM) can be faced with the challenge that the materiel solution they are developing is not meeting its required specification(s). However, this materiel may arguably be better than what is fielded for the same purpose. The challenge becomes how to make that case to senior acquisition decision makers who determine if a system is acquired or not.

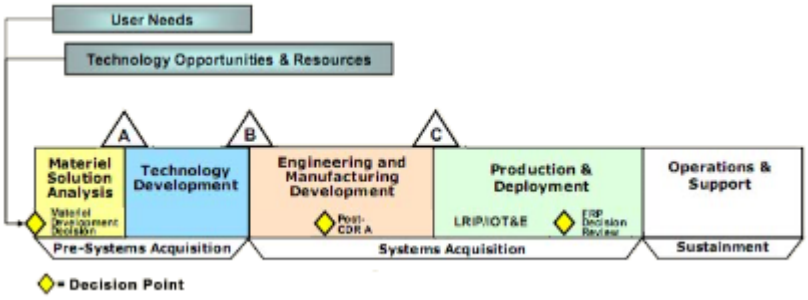

Figure 1 – DoD Acqusition Life Cycle.

To assist in making a compelling argument with supporting data, the PM calls upon the analyst. In some cases, enough developmental test data exists to help illustrate the value of a system. When it does not, simulation is frequently used as a tool. The difficulty in the intricacy of the problem and the maturity of the simulations available then increases the level of complexity when actually using these simulations. When things become too complex, M&S experts can be required. In some cases, new models or simulations need to be developed. Of course, this assumes that the PM actually knows what simulations exist for their problem space and how to get them. We will walk through an acoustic sensor example in section 5.4 below.

3.0 THE TECHNICAL BARRIERS TO ROBUST USE OF SIMULATION

There are many obstacles for using M&S within the US DoD. This paper focuses on the technical barriers rather than the issues that relate to bureaucracy, financial resources or any other non-technical considerations. Those issues are very important and should not be overlooked, but our project, Executable Architecture Systems Engineering (EASE), is focused on technology solutions for bringing together distributed M&S for the appropriate purposes (hopefully despite many possible non-technical considerations).

The sheer breadth and depth of warfare to be represented adequately is massive. Understanding exactly what parts of warfare need to be represented is based on a detailed breakdown of the Measures of Performance (MoPs) and Measures of Effectiveness (MoEs) [3] for an event's goals. Once the modeling requirements are known though, it is impossible to know what exactly exists throughout US DoD in order to help. There have been efforts to catalogue the existing M&S assets but the information gathered is almost always limited to textual descriptions. Much work remains to be accomplished in order to understand whether the application fits the needs per fidelity, resolution and interoperability, along with many other factors.

A major problem with using multiple systems together is the interoperability among those systems. Interoperability among distributed M&S is complex, tedious and often difficult to evaluate. Integrating models that were developed for various purposes with disparate technologies and managed by independent organizations is often the goal. The effort required to meet this goal is frequently underestimated due to misunderstood commonalities between those applications. Common compliance with middleware architectures, modeling goals and object models gives a false impression of complete interoperability. There are numerous considerations when developing a distributed simulation environment. The event's objectives drive the necessary simulation functions but how those simulation functions interact needs to be meticulously designed for true interoperability. The semantics of the information transmitted, the behavior necessary across multiple applications and fidelity and resolution synchronization are only a subset of the systems engineering necessary for a coherent SoS.

Once the appropriate M&S applications have been procured, configured and integrated, there is a significant

workforce requirement to learn how to use, setup, manage and execute the M&S applications for both the current event as well as future events. Reuse of M&S environments can provide cost avoidance, but retaining organizational knowledge is difficult with workforce turnover, particularly in this era of smaller budgets and shorter execution time periods. Once a M&S event concludes, we have often seen computers repurposed, configurations and software modifications completely lost and engineers moved on to other projects. It becomes impossible to build on the previous event with small changes so the organization must start almost from the beginning spending nearly the same resources as spent originally.

Towards this end, we have established a data-driven systems engineering infrastructure which allows SoS design encapsulation and connected an interview subsystem which allows a user to launch a distributed M&S execution based on functional and scenario choices. We have implemented generative programming techniques [4], which automatically generate executable computer programming artifacts from a higher level source, in order to quickly deploy a SoS architecture for military analysis. The flexibility required to implement our goal requires systems architecture qualities and objectives. This includes encapsulation of functionality into appropriately sized portions to be able to manipulate and construct larger capabilities, as needed, with as little engineering effort as possible. We aim towards an architecture that is fully compliant with US Army Verification and Validation guidance [5], and robust enough for decision-oriented analysis, while maintaining flexibility and quickness in order to save the DoD tremendous amounts of time and effort when constructing distributed M&S environments for various uses.

4.0 WHAT WAS: MODELING ARCHITECTURE for TECHNOLOGY, RESEARCH AND EXPERIMENTATION (MATREX)

To understand the impetus for solutions provided by the EASE research, it is important to understand where we have been. The Modeling Architecture for Technology, Research and EXperimentation (MATREX) program [6] had the mission to research and develop an M&S environment that included a collection of multi-fidelity models, simulations and tools which were integrated into an established architecture to conduct analyses, experimentation and technology trade-offs. The MATREX program was made up of many US Army Research, Development and Engineering Command (RDECOM) labs, centers and activities providing simulation solutions into the overall system architecture. A number of different customers used the simulation environment for varying purposes. Any particular instantiation of the MATREX system could be dramatically different than the next based on the user requirements and the subsequent model selections and system design choices made to satisfy the functional requirements. However, the flexibility of the system created a complex system design problem by allowing many different possible configurations.

The numerous and often generic potential uses of the system offered a difficult systems engineering challenge to link system requirements to detailed system design and technical dependencies. The MATREX Environment needed to retain the flexibility of the technical solution set while providing a rigorous and thorough systems engineering product set that could be used to design a system instantiation, provide technical design contracts and link low level data elements to high level user functional requirements. This need drove the initial creation of the System Design Description (SDD) [7], which is a data-driven systems engineering tool that linked operational and technical requirements to design decisions, allowing engineers to collaborate on system integration and provide traceability to event objectives. This tool was extended within the EASE project to support research goals as described in Section 5.0.

Other tools developed within the MATREX project included tools to support rapid software development including a software library (ProtoCore) that abstracted away middleware details and allowed applications to run across different middleware protocols. It also included an over-the-wire testing tool (Advanced Testing

Capability (ATC)) that provided stimulus and validated applications based on sequence diagrams that were imported from the systems engineering tool. These tools will be further explained in section 5.3 when extensions to support EASE research goals are described.

These tools enabled a more accurate and quicker process for developing and integrating M&S applications, applying systems engineering throughout development and testing. While useful, these tools still expected an M&S expert to employ them, leaving that expertise specialized and perishable. The next logical step was to build on that with automation by capturing additional details about the M&S environment, including how to install, configure, launch and capture data from those same applications. We could then orchestrate the execution of the M&S environment based on the same systems engineering data already used to ensure the correct warfare representation while accomplishing true interoperability. This next step is the EASE project.

5.0 WHAT IS: EXECUTABLE ARCHITECTURE SYSTEMS ENGINEERING

5.1 BACKGROUND

The goal of EASE is to lower the barrier of entry to the use of M&S. EASE provides a single interface for systems engineers, software developers, information technology professionals and analysts to work together. These individuals define the simulation systems engineering data and execute the appropriate applications in order to support the M&S user's goals. EASE provides an interface to M&S users to select the capabilities they require and the scenario necessary to stimulate the appropriate warfare circumstances. The selection criteria are used to filter and display the most appropriate executions for the user to choose from. The user can then adjust configuration elements that have been exposed by the developers, select the number of runs they need to execute, schedule runs and hit the "Go" button to execute. The web-based interface provides a mechanism to launch potentially complex M&S in the cloud or on specific computing hardware. The systems engineers, developers and integrators can centrally manage all aspects of EASE and the execution of the proper M&S systems to achieve the M&S users' requirements. Having a data-driven and easy to use interface keeps the systems engineering technical information (i.e. interface specifications) current. In turn, each user can be assured that they're referencing and updating the latest information.

5.2 NEEDS DERIVATION

Simply learning which M&S and analytical tools exist within the DoD is challenging enough let alone actually obtaining them for use. Once users receive these systems, they still need to be trained and/or read lengthy and complicated user manuals on how to configure the systems and which execution options to use for a desired effect. This process is painful, time consuming and costly; so much so that users will opt for a simpler, but less effective solution. In order to ensure that the best tools DoD has to offer are used there is a need to quickly and easily find execution options for specific M&S needs.

After users become educated in the systems they use, that knowledge is generally not documented and remains only in their head. The complexity and nuances of running highly technical systems is if often too difficult or too time consuming for them to share the information with their peers. Each system is also delivered with its own types of documentation and few seem to follow existing standards when creating this documentation. There needs to be a method for capturing systems technical information in a common format for Systems Engineers (SEs). This method should connect functions across systems, understand the warfare capabilities of each element within the system and link the M&S solutions to experiment goals without adding more cost when compared to activities already being conducted to execute the experiment. In order to maximize the user's derived knowledge and time expended, there is a need to link systems engineering information with execution details.

Currently, the warfare functions of each M&S system are described through brochures, slides or user manuals in

human readable text. This is only a precursor to what engineers and analysts need. Specifically, there needs to be more detailed information available and captured within a common systems engineering tool. Items, such as object model elements, middleware types, versions and execution options, need to be linked and the consequences of choosing each option understood as it relates to the warfare functions represented. For example, configuring a system to have the right resolution for the function under analysis is a configuration option and needs to be linked to the correct function. This requirement leads to the need to determine necessary technical systems, object models and middleware based on warfare functions required.

Knowing that two simulations represent warfare functions that seemingly compliment a larger analytical goal (e.g. a weather simulation and a chemical agent dispersion simulation to model a chemical release) does not necessarily imply that they will work together semantically. Even if two systems work on the same middleware and use the same object model, they still might not be interoperable when it comes to which data elements within the object model each system sends or receives. These important distinctions lead to the need to capture technical interface details to facilitate identification of integration gaps and understanding the data provided for analysis.

The semantics or reasons for systems communicating are also very important in order to determine that the two systems are indeed sharing the appropriate data. The M&S user needs the capability to easily capture these technical details and have better visibility to discover gaps for interoperability and how systems can be integrated. Providing a tool that assists users in integrating systems with true interoperability is the objective.

Software development schedules are often delayed. In turn, when multiple applications are designed to share data the development teams become reliant on others' schedules. This has major impacts to overall schedule and cost. Having the ability to quickly generate a surrogate application to replicate the functionality of a missing system allows the other systems to integrate into the distributed system and test their interfaces, timing and so on. This provides cost avoidance in those cases when a simulation system is unable to integrate. This leads to a requirement to create surrogates when key systems are delayed.

The simulation community needs a rapid application development mechanism to quickly generate the software for connecting distributed simulations. This technology can be generic enough to be applicable across any model use case. Having the ability to generate source code will greatly reduce the software development cost of developing new models and integrating existing models into distributed environments. The generated code includes the ability to connect to the appropriate middleware, send and receive the right messages and even has software constructs that will simplify a modeler's learning curve into distributed simulation environments. This leads to the need to quickly, easily and more cost effectively modify a model to work within a distributed simulation environment.

Managing computers in a laboratory can be time consuming, redundant and tedious. Executing simulation systems across a laboratory can add to that burden. Launching a large distributed simulation environment can often take over an hour wherein the users have to manually script how the systems will be launched or even worse, walk around the lab and launch each system on each computing device manually. A system to manage the computers and launch applications according to the correct execution details and order is required. This leads to the need to launch complex computing assets easily from a single point.

In laboratories that execute many simulation environments, each one can be slightly different from the previous one. Managing how each system needs to be modified for changing scenarios or even technical constraints like middleware or object model differences requires engineers to spend much of their time configuring and testing systems. This leads to the need to orchestrate the order and cooperation of systems as appropriate to the scenario

and technical interoperability details.

Hardware requirements change depending on the applications, the scenarios they need to represent and the exercise architecture, among other things. Having to procure additional hardware can be expensive and unnecessary. Moreover, each computer in the lab has a finite useful life. Once the systems and scenarios grow, the hardware becomes unable to support the execution without upgrades. Having a cloud-based system to dynamically add and allocate processors, memory and network bandwidth will help alleviate the lab management of limited life time hardware. This leads to the need to flexibly allocate computing resources (memory and processors) to simulation systems based on scenarios, configurations and application-specific details.

Software integration with middleware specifications, such as the High Level Architecture (HLA) [8], can be complicated and error prone. Once integrated and tested, other software developers can reuse the software library for their own use. Making the software library generic to work across any object model and adding plug-ins to work across multiple middleware specifications allows this library to be reused across a wide spectrum of simulation systems. It additionally facilitates interoperating simulations which were not originally planned to work with other simulations. This leads to the need to abstract away technical middleware details from business logic to facilitate reuse and remove errors.

Requirements written in human readable text and provided to software developers can often be misinterpreted, especially if those requirements do not include enough detail or the semantics of the requirement. When system developers arrive to integrate their system for an analysis, any misinterpretation of the requirements will be discovered through trial and error. Another problem that occurs frequently is that system developers write their own simulation test procedures so any errors that they have in their minds will also be in their tests. These problems include erroneous encoding and decoding of simulation communication messages and middleware specification errors. Instead of discovering problems at the exercise site while personnel are on travel and using funds for hotel, per diem and other expenses, it would be useful if tests could be generated for the developers that properly test everything possible prior to developers traveling to the exercise site. These generated tests should test an application's middleware connection as well as the object model elements it needs to receive and send. This leads to the requirement to test systems prior to integration events based on an agreed upon system design.

The design of an analysis changes frequently as analytical goals are modified as well as between analyses that may leverage elements of the same simulations. Having to manually update test cases for simulations involved will lead to configuration management problems and be a time and cost driver. Being able to automatically update test cases based on a systems engineering tool that captures the methods and means of the analysis will save time and reduce errors. This creates a requirement to quickly update tests from design via automation and a data-driven export mechanism.

5.3 COMPONENTS

EASE consists of the following components and associated software:

- Interview Component
- System Design Document
- Surrogate Generation Capability
- Deployment Management System
- ProtoCore

- Advanced Testing Capability

The Interview component of EASE is the interface for the M&S users, systems engineers, integrators and software developers to access and manage their respective areas of complex simulation. The user has the ability to search the system for scenarios that are applicable to their specific needs, configure those scenarios and execute the simulation environment on dedicated hardware assets by simply clicking on a button within the web browser. They can later return to the Interview interface to access the data artifacts that resulted from the scenario they previously executed. Systems Engineers enter into the framework what applications can perform what functionality, which then informs how scenarios can be created. Integrators create adjustable configuration fields for M&S users to configure complex simulation applications through an easy interface. This allows constraints to be put on the models, simulations and tools that ensure that the systems do not operate outside of their limits. Following the rules laid out by the systems engineers, the developers can upload, configure and approve their software for future execution within EASE.

The SDD is a systems engineering tool used by the systems engineer to capture the design details of a distributed computing environment. The SDD links high level requirements to subsystem specific details through Modeling Design Decisions that describe how the simulations will communicate, including sequence diagrams and architectural strategies. The SDD is a database driven tool which stores all of its information in the form of database fields with links across the database tables. This database driven approach allows the system to quickly generate systems engineering artifacts with database queries and templates for their output and subsequent use by the systems engineer. If a change is made to any of the systems engineering data, this artifact generation can be repeated automatically by the systems engineer. This ensures that systems engineering artifacts remain current with little effort, compared to most projects that need systems engineers to constantly update and configuration manage Microsoft Word, Excel or PowerPoint documents to ensure currency and consistency.

The Surrogate Generation Capability uses the SDD's ability to generate artifacts based on the SDD database. An SE can enter simulation business logic into the SDD and export a working software application that will execute within a distributed simulation environment based on the appropriate middleware and object model. This capability eliminates the need for the SEs generating a surrogate to: understand the simulation middleware details; know how to write interface details that are often repeated; or, know how to write a multi-threaded software application optimized for distributed simulation. The Surrogate Generation Capability includes an interface that is already filled in by the SDD based on which warfare function is to be surrogated. The correct events have already been included, with fields available for the user to manipulate and/or add their own simulation business logic. Once completed, the systems engineer can save their work back into the SDD, export the software application to their local desktop for further development or use and can have the surrogate they created automatically deployed to EASE for use by users in future executions.

The Deployment Management System component of EASE is responsible for the automated orchestration of simulation executions using dedicated hardware assets. In any distributed simulation environment, there is a specific order and configuration of the components for them to execute properly. This is often known only by a handful of integrators on each project. The Deployment Management System component captures this knowledge and automates it so that anyone can execute complex simulation environments. As a part of that orchestration, applications must be configured for the middleware, the application's performance data and for the specific scenario to implement, among other areas. Each component is executed in an emulated computing environment, known as a virtual machine, and via a virtual machine management interface. This allows EASE to dynamically partition processors and memory to each virtual machine, as appropriate, rather than be tied to the limitations of an existing piece of hardware with its associated operating system. Instead, each application gets the operating system and hardware required to properly execute. Those virtual machine executions can also be

scheduled, repeated, started, stopped and monitored by the Deployment Management System component. A video stream is provided to the user to monitor each virtual machine while it runs, which is key to supporting Human-In-The-Loop (H-I-T-L) simulations [9]. After a simulation run has been completed, the data artifacts are gathered and exposed to the Interview component for the users to get data for their analysis. This implementation allows for easy scaling and management of hardware, software and their connection to requirements and goals of the simulation execution.

ProtoCore [10] is a software library developed to allow software developers to create simulations capable of communicating with other simulations in a distributed architecture without having to be experts on distributed simulation. Most distributed simulation middleware architectures have very similar concepts such as joining, subscribing, publishing and exiting. Distributed simulation environments also have some common simulation business logic, such as dead reckoning, time representation and coordinate conversions. These types of common concepts and utilities are included within ProtoCore so software developers do not have to write their own implementations. This saves developers time and it also helps ensure accurate implementation since the logic has been peer reviewed and used across many different simulations. An additional benefit of ProtoCore is its ability to provide these capabilities across a variety of middleware architectures due to its plug-in architecture. Plug-ins exist for HLA 1.3, HLA 1516, Distributed Interactive Simulation (DIS) [11] and Test and Training Enabling Architecture (TENA) [12] so a software developer using ProtoCore can write their code once and choose which middleware that it will use at run-time. This allows software developers that support multiple projects on different middleware architecture to write their software once and allow it to work across several environments.

The Advanced Testing Capability [13] is a software tool that is used to test distributed simulation applications under controlled conditions without needing every simulation involved in a scenario. First, ATC is started along with any necessary middleware architecture components. Then, the simulation under test is started and connects to the middleware and ATC. ATC provides the stimuli to the application that is required for the scenario being tested and verifies the application's responses as sent over the middleware. This type of testing ensures that the application can properly join the middleware, transmit the data based on the middleware architecture's guidelines and publish and subscribe to the correct events. The ATC tests are presented as sequence diagrams where a tester can edit details, such as the events' attributes and the timing of each event. The ATC stores the test cases into an eXtensible Markup Language (XML) file called the Test Case Markup Language (TCML). The TCML file storage allows other tools to read, manage and export test cases. Additionally, the SDD can export TCML files based on system design information captured within its database.

5.4 USE CASE

Beginning with a hypothetical problem, assume an acoustic sensor has a requirement to detect and discriminate targets, such as manned and unmanned ground vehicles, in urban environments with a specified false alarm rate [14]. During developmental testing, this acoustic sensor appears susceptible to background noise that could appear in some urban environments and, in turn, is not able to detect and discriminate targets in these environments with the required false alarm rate. It is, however, able to discriminate all required targets in non-urban environments, as well as a subset of urban environments that may be relevant to future operations, within the required false alarm rate. The current fielded acoustic sensor is significantly less reliable in the urban environments of interest. The PM wishes to make the argument that this new system should pass Milestone B due to the gains it provides to the force.

The analyst creates an experimental design that compares the current acoustic sensor to the one under development including operational scenarios in relevant urban environments. While he would like to use available empirical data, he is also interested in using physics-based models that replicate the acoustical

phenomenology at hand and show how the sensors will perform as background noise is varied. The analyst logs in to EASE and sees that he already has models for the current system from when it was developed and fielded. Moreover, he has models of the system being developed from Pre-Milestone A. Using EASE, he modifies the scenario he had from Pre-Milestone A to reflect the operations in Milestone B and adds both sensors for comparison. He then modifies parameters within the simulations reflecting the background noise as input. Finally, he schedules multiple replications due to the stochastic nature of the physics-based models being used and hits the "Go" button. EASE then runs the simulations using available resources and provides the analyst with data when complete. Conveniently for this analyst, he was able to load his simulation post processors which modify the data for use as information after the runs are complete into the EASE system further automating the process. Through this analysis, he is able to show a comparison of the developmental system to the current system and operationally make the argument that there is utility to the developmental system. It is then up to the decision makers whether the operational utility outweighs the cost and sustainment footprint for a new system that is not meeting all requirements.

It should be stressed that there is no magic in this hypothetical situation. In our example, M&S professionals developed models that represented the acoustic sensors in question and systems engineers took the time to integrate them into EASE. Moreover, the analyst knew how he wanted to present the data and built post processors to facilitate the process. The key here was that as these models were developed, they were put into the EASE framework. In doing so, the constraints and capabilities were known as well as how to execute them. This allowed our analyst to take advantage of work done previously, possibly on an analysis of another weapon system for another PM, without having to call on the M&S experts or become an M&S expert himself. EASE also allowed the analyst to easily modify parameters, schedule runs and receive data. If the data looked incorrect, for whatever reason, the analyst could easily change the inputs and run again. Normally, this process is done by hand and is error prone, but the rigor of EASE ensures that this is not an issue. Should there be the need for a new model, the experts would then be called upon. Furthermore, should the question change from a comparison of acoustic sensors in environments for which the PM understood to a more SoS-like situation where the acoustic sensors had to interface with other operational systems, additional models may need to be entered into EASE. If these models were entered into EASE and a SoS question arose, various PMs would have the ability to leverage models from other PMs (presumably with some level of accreditation) to answer analytical questions that do not have just one PM. EASE provides that ease of use access to the M&S while facilitating re-use.

5.5 MAPPING OF NEEDS TO COMPONENTS

Need	EASE Component
Quickly and easily find execution options for specific M&S needs	Interview
Link systems engineering information with execution details	Interview
Determine necessary technical systems, object models and middleware based on warfare functions required	Interview
Capture technical interface details to facilitate identification of integration gaps and understanding the data provided for analysis	SDD
Integrate systems with true interoperability	SDD
Create surrogates when key systems are delayed	Surrogate Generation Capability
Launch complex computing assets easily from a single point	Deployment Management System
Orchestrate the order and cooperation of systems as appropriate to the scenario and technical interoperability details	Deployment Management System

Flexibly allocate computing resources (memory and processors) to simulation systems based on scenarios, configurations and application-specific details	Deployment Management System
Quickly, easily and more cost effectively modify a model to work within a distributed simulation environment	ProtoCore
Abstract away technical middleware details from business logic to facilitate reuse and remove errors	ProtoCore
Test systems prior to integration events based on an agreed upon system design	ATC
Quickly update tests from design via automation and a data-driven export mechanism	ATC

Table 1 – Mapping of Needs to Components of EASE.

6.0 WHAT SHOULD BE: COMMON MODEL FRAMEWORK

While the M&S community works hard to produce solutions that support the needs of the analytical community, modelers and simulation developers often fall into the trap of focusing on their particular domain. They may or may not attempt to leverage existing representations of phenomena because they are so focused on what they need to model or simulate for the analyst. Reuse is always a hot topic, as is composability, but there are barriers to these two ideals that have kept them from becoming a reality.

The idea of a framework that brings models together as needed is not novel. Some might argue that various simulations have been defacto frameworks to that end. For example, we continue to develop specialized terrain to support the needs of simulations and recreate physical representations that support kinetic warfare. We do this because "our" particular simulation was not built to use "your" model, due to issues such as fidelity, format or data. Software programming, in general, relies on libraries that become canonical representations of their functions. These libraries can also be changed as necessary. Why aren't we using this approach for simulation development?

Imagine a paradigm where an analyst is able to pull together models that represented phenomena necessary to replicate the problem space being explored. These models would be produced by experts in those particular fields. This indeed would require some level of regulation and a serious Verification, Validation & Accreditation discussion, but we save that topic for a future paper. From the point of view of the analyst, if he is trying to have a fair comparison of two systems in a relevant operating environment, having a common source for models would be key. Furthermore, having the ability to pull together those same models for the next analysis, or being able to run updated models using the parameters of the original analysis and then performing a new analysis, would provide great analytic rigor. A potential solution might be a repository that literally houses these models and allows another analyst to leverage what was previously done, instead of the current perishable description of a model or simulation.

The paradigm of distributed simulation in general arguably provides a level of reuse models and simulations; however, as discussed, taking a black box approach to simulation interactions leads to interoperability issues and does not support reuse of fundamental models. Part of the challenge lies in defining the primitives of what those fundamental models would be. There is additionally still a challenge in the breadth of uses of M&S to support acquisition. The types of models for a system-level analysis normally differ from the models used for a force-on-force analysis, but is that required? We need to derive environment representations from a canonical source without having data translation errors that plague terrain generation, simulation gateways, etc. This is an area for

serious research and demonstration to prove where the state-of-the-art really is.

Furthermore, models and simulations are worth little without the data that drive them. There are numerous activities in the US and North Atlantic Treaty Organization (NATO) discussing data generation, collection and storage. What remains to be seen is how the M&S space can effectively tie in to these efforts, especially in context of taking advantage of the data as it emerges from the battlespace. Would a common framework for models better support this linkage and in turn, better support the lifecycle? Furthermore, while we have discussed the need for representing kinetics of warfare, research is needed to better support scenario development. The lack of standardization in scenario generation across simulation environments does not allow us to easily sketch out a mission for execution. Would a common model framework further improve this problem?

It is our belief that advancements in technology are beginning to solve the problems in computational power, data storage and distributed access to models and simulations. What remains is a concerted effort to define what capabilities an analyst would actually desire to do his job independent of the current methods and means used to produce and use M&S. Arguably, better defining how M&S could best support the acquisition lifecycle will allow us to move forward rather than continue a slow evolution.

7.0 CONCLUSIONS

While there remain challenges to enabling analysts to select M&S throughout the entire DoD Acquisition Lifecycle Process, by examining where we have been and where we are now we can make recommendations for where we should be. The challenges that we have identified with current methods will continue to be challenges as long as M&S is designed, developed and employed in the same way it has been. The EASE research project attempted to implement solutions to many of the challenges that we identified through our experience with the MATREX program and has found success in many. Unfortunately, while EASE can provide many benefits, it cannot fully enable true composability and reuse as long as M&S continues to be developed with disparate timelines and purposes. If the analysts define their ideal system, the Science and Technology community can demonstrate what technologies can achieve this vision driving towards a more useful paradigm in the future.

8.0 REFERENCES

[1] M. Jamshidi. 2008. System of Systems Engineering. 1st ed. Wiley.

[2] Defense Acquisition University (DAU) ACQuipedia, 22 January 2008, "Acquisition Life Cycle" available via https://dap.dau.mil/acquipedia/Pages/ArticleDetails.aspx?aid=30c99fbf-d95f-4452-966c-500176b42688

[3] Roedler, Garry and Jones, Cheryl. Technical Measurement. A Collaborative Project of PSM, INCOSE, and Industry. 2005 INCOSE-TP-2003-020-01. (https://www.incose.org/ProductsPubs/pdf/TechMeasurementGuide_2005-1227.pdf)

[4] Czarnecki, K, 2000, "Generative Programming: Methods, Tools, and Applications", Addison-Wesley Professional.

[5] Headquarters Department of the US Army, Army Regulation 5-11. Management of Army Modeling and Simulation. 2014. (http://www.apd.army.mil/pdffiles/r5_11.pdf)

[6] Metevier, Chris et al. Modeling Architecture for Technology Research and Experimentation (MATREX): M&S Tools and Resources Enabling Critical Analyses. 2009 Modeling and Simulation Information Analysis Center (MSIAC) Journal Summer 2009. (https://www.matrex.rdecom.army.mil/front/msiac_journal_july_2009.pdf)

[7] Beauchat, Tracey et al. A Collaborative Tool for Capturing the Design of a Distributed Simulation Architecture for Composable Execution. 2012. Fall Simulation Interoperability Workshop – Spring Conference.

[8] 1516-2010 IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Framework and Rules. 2010. (http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5553440)

[9] Rothrock, Ling and Narayanan, S. Human-in-the-Loop Simulations: Methods and Practice. 2011. Springer.

[10] Keith Snively, Phil Grimm "ProtoCore: A Transport Independent Solution for Simulation Interoperability." SISO Fall SIW 2006.

[11] 1278.1-2012 IEEE Standard for Distributed Interactive Simulation – Application Protocols. 2012. (http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6387564)

[12] Powell, Edward and Noseworthy, Russell. The Test and Training Enabling Architecture (TENA). 2012. (https://www.tena-sda.org/download/attachments/6750/TENA-2012-Paper-Final.pdf)

[13] McCray, Paul and Snively, Keith. Functional Component Testing for Distributed Simulations. 2008. Simulation Interoperability Workshop Spring Conference, April 2008.

[14] Kaushik, B; Nance, Don; Ahuja, J, 23-25 May 2005, "A Review of the Role of Acoustic Sensors in the Modern Battlefield", Proceedings of the 11th AIAA/CEAS Aeroacoustics Conference (26th AIAA Aeroacoustics Conference).