A Path Forward to Protocol Independent Distributed M&S

Christopher Metevier, Chris Gaughan
SFC Paul Ray Smith Simulation & Technology Training Center
Orlando, FL
Chris.Metevier@us.army.mil, Chris.Gaughan@us.army.mil

Scott Gallant, Kiet ("Jeff") Truong
Effective Applications Corporation
Chuluota, FL
Scott@EffectiveApplications.com, Jeff@EffectiveApplications.com

Gary Smith
Dynamic Animation Systems
Orlando, FL
gsmith@d-a-s.com

ABSTRACT

In order to keep up with the rate of technology advancement, solutions must be created that transcend hardware, middleware, protocols and data structures to allow for a sustainable implementation. Distributed Modeling & Simulation (M&S) includes applications executing their intended use for the collective goal of a System of Systems (SoS) M&S environment. Distributed M&S is fundamentally based on the exchange of information between functions that may not have been built to work together.

The Modeling Architecture for Technology, Research and EXperimentation (MATREX) program has been a synchronization point for M&S across the U.S. Army Research, Development and Engineering Command (RDECOM) for many years. It has brought together models, simulations and tools from the RDECOM labs, centers and activities into a common architecture and environment. The program's focus has been on integrating these disparate applications into a harmonious solution for engineering model development and evaluation, technology tradeoffs, capability assessments, concept development, experimentation, testing and training. MATREX has developed tools that support this integration through multiple simulation middleware protocols. It has also developed tools that abstract away the integration details from the application developer. In particular, it allows the modeler to develop models while MATREX provides the tools to handle the M&S integration intricacies such as interest management, encoding/decoding, dead-reckoning, etc.

This paper will describe these Government-owned middleware agnostic tools along with a SoS Systems Engineering approach and infrastructure that can link M&S functional requirements to model-specific data requirements and code generated testing. MATREX tools and Systems Engineering process have been used across several programs, including the Training and Doctrine Command Battle Lab Collaborative Simulation Environment (BLCSE), Army Test & Evaluation Command Operational Test Command (OTC) Analytic Simulation and Integration Suite (OASIS) and Brigade Combat Team Modernization Simulation Environment (BSE), to name a few. This methodology could benefit many more M&S programs.

ABOUT THE AUTHORS

Christopher Metevier is the Technology Program Manager (TPM) of the Modeling Architecture for Technology, Research, & EXperimentation (MATREX) program at the Simulation and Training Technology Center (STTC) Human Dimension, Simulation and Training Directorate, Army Research Laboratory (ARL). He has over 20 years of experience with the Navy and Army in the Modeling & Simulation (M&S) field. His M&S experience extends across the acquisition lifecycle and includes the research, development, adaptation, integration, experimentation, test, and fielding of numerous simulation technologies and systems. He received his Master of Business Administration from Webster University and Bachelor of Science in Electrical Engineering from the University of Central Florida.

Chris Gaughan is a Chief Engineer working at the US Army STTC Human Dimension, Simulation and Training Directorate, Army ARL. He is currently the Deputy TPM for RDECOM's MATREX, a program focused on creating a composable M&S environment wherein a collection of multi-fidelity models, simulations, tools and resources can be integrated and mapped to an established architecture for conducting analysis, experimentation and technology tradeoffs for RDECOM and others. From 2004-2009 he worked at the Edgewood Chemical Biological Center (ECBC) where he served as the Configuration Manager of the Chemical, Biological, Radiological, Nuclear (CBRN) Simulation Suite, a tool suite for CBRN materiel and hazard effects simulation. He received his Master of Science and Bachelor of Science in Electrical Engineering from Drexel University in Philadelphia, PA.

Scott Gallant is a Systems Architect with Effective Applications Corporation. He has over 15 years experience in distributed computing including US Army M&S. Scott has led technical teams on the MATREX program for distributed software and federation design, development and execution management in support of technical assessments, data analysis and experimentation. He has also led the systems engineering team for the implementation of the described Systems Engineering product infrastructure. Scott currently serves as the System Architect and Lead Systems Engineer for the MATREX program.

Kiet ("Jeff") Truong is a Principal Systems Engineer with Effective Applications. He has over 20 years of Systems/Software Engineering and Technical Management experience in distributed M&S, embedded systems, telecommunication/networking systems, and network management systems. He has led a team responsible for defining and managing the interface requirements for the Brigade Combat Team Modernization (BCTM) M&S System of Systems (SoS). He also chaired a Federation Object Model Working Group to define object model and federation agreements for MATREX and the BCTM SoS simulation events. Jeff currently serves as a Systems Engineer for the MATREX program responsible for the requirement definition of the MATREX M&S tools. He is also working with other RDECOM Labs and Centers to mature their M&S models and integrate them into the MATREX Federation.

Gary Smith is a Principal Software Engineer with Dynamic Animation Systems, Inc. He has over 10 years experience in DoD M&S as a technical lead engineer, software engineer and software developer. He has led technical teams on the MATREX program for systems engineering, distributed software design and development efforts as well as integration and testing efforts that support an execution for a customer in support of data analysis, experimentation, and fielded testing. Gary is currently the Technical Lead for the MATREX program.

A Path Forward to Protocol Independent Distributed M&S

Christopher Metevier, Chris Gaughan
SFC Paul Ray Smith Simulation & Technology Training Center
Orlando, FL
Chris.Metevier@us.army.mil, Chris.Gaughan@us.army.mil

Scott Gallant, Kiet ("Jeff") Truong
Effective Applications Corporation
Chuluota, FL
Scott@EffectiveApplications.com, Jeff@EffectiveApplications.com

Gary Smith
Dynamic Animation Systems
Orlando, FL
gsmith@d-a-s.com

EFFECTIVE INTEROPERABILITY

Most integrators concentrate on integrating models by focusing on the syntax for the interfaces. Although that is a critical step, interoperability between models needs to be focused first and foremost at the functional level. Models within the Department of Defense (DoD) Modeling & Simulation (M&S) community are developed along disparate timelines and for disparate purposes. Programs that fund the development of models cannot afford to coordinate across the community on requirements so they must concentrate on a narrow set of modeling requirements serving their highest priority capabilities. Even One Semi-Automated Forces (OneSAF), a model meant to be used across the DoD M&S community, has an Operational Requirements Document (ORD) that does not fulfill everyone's requirements.

When migrating a High Level Architecture (HLA) (IEEE 1516-2000) application to a Federation Object Model (FOM) it was not developed for, developers map internal model attributes to the fields found in the FOM. Most of the attributes do not map one for one and have the same underlying intent. Due to the ill-matched attributes, a message, an attribute or an object can be misused by the model. For example, if we had one model representing the hull of a tank, a different model representing the turret, and they used a FOM with only one platform orientation field, the developers would need to decide which model should fill that field in. If the field is used for graphical representation of platform movement, then the field should be the hull's orientation so it wouldn't look like the tank was moving sideways. However, if the orientation field was used for fires modeling, then the orientation of the turret is necessary.

As illustrated in the tank example, if we rely on developers making object model choices without a broader understanding of the functional requirements, these types of interface discrepancies will occur and the event's goal(s) will be compromised. A higher level of functional requirements and functional design early in the event's lifecycle can help avoid these interface interoperability issues. (Tolk, et al., 2003)

Overview of Functionally Oriented Integration

During a System of Systems (SoS) experiment, analysis, event, etc., the SoS requirements must be understood and continually focused on while designing the implemented solution. In the Modeling Architecture for Technology, Research and EXperimentation (MATREX) distributed architecture (Tufarolo, et al., 2004), different models will fulfill many different pieces of the required functionality, but will need to integrate with the rest of the models to accomplish the SoS goals.

When integrating models together, we must have an understanding of the functions that each model implements and their data and functional dependencies within a higher level understanding of the system. In the above tank example, a requirements decomposition from a semantic interoperability perspective would identify the need for a FOM change early in the development lifecycle. Many management teams will attempt to reuse a FOM in order to save costs and provide configuration control, however, if interface issues are identified, the changes must be made to the

FOM in order to have a traceable and valid event. Using a FOM as a base for integration is acceptable, but assuming that an object model will be suitable for all Army events is naïve.

MATREX SYSTEM OF SYSTEMS (SOS) SYSTEMS ENGINEERING APPROACH

The MATREX suite of models, tools and architecture allow for many different possible configurations of the system to achieve the user's functional requirements. The goal is to work with the user to develop an event-specific System Design Description (SDD) that contains their exercise requirements, data decomposition requirements, system architecture guidelines, scenario, configuration choices and model selection. Systems Engineering (SE) for distributed systems (Jamshidi, et al., 2008) is complex and unique from Systems Engineering for stand-alone software systems. The MATREX Systems Engineering (SE) process assists the user by mapping their functional requirements to model behavior and mapping the interfaces to the existing FOM fields as best as possible. If necessary, FOM changes are identified and submitted to the FOM Management Focus Group for inclusion in the next FOM release. As an aside, many organizations throughout the military M&S community use the MATREX FOM, including RDECOM, the Training and Doctrine Command (TRADOC), the Army Test & Evaluation Command (ATEC) and the Program Executive Office for Integration (PEO I).

The data-driven approach to mapping technical detail to the high level requirements produces the mapping between the data collected during the exercise, initial exercise goals and functional capability data decomposition.

The MATREX architecture includes many disparate models (differences with functionality, fidelities, resolutions, technical dependencies, etc.) that need to work together. These models can be combined within an architecture that scales by using complicated techniques, such as Data Distribution Management (DDM) (Van Hook, et al., 1998). Those technical complexities force design, development and testing collaboration to be well coordinated and as automated and data-driven as possible. Simulation requirements such as "fair fight" issues, scalability concerns and data element analysis force the design to have additional architectural strategies that must be uniformly followed. These architectural strategies must be captured and enforced with an SDD that is linked to the design.

The MATREX program has developed the SDD as a product and process to capture the system design at a functional level and subsequently link the functional design to the technical design. This allows the functional requirements to be linked to system design and allocated to specific models. The low level requirements, object model and test cases can then be auto-generated based on model allocation to functions. The product is data-driven, easing information maintenance duties by linking the products and simplifying the editing of the system design. The product also allows for auto-generating low level specifications, Department of Defense Architecture Framework (DoDAF) (DoD CIO 2009) views and test cases.

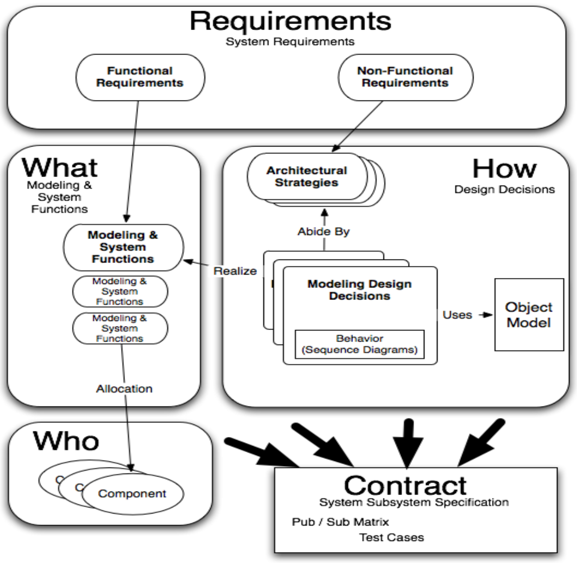
Requirements
System Requirements
Functional Requirements
Non-Functional Requirements
What
Modeling & System Functions
Modeling & System Functions
Modeling & System Functions
Modeling & System Functions
Allocation
How
Design Decisions
Architectural Strategies
Abide By
Realize
Modeling Design Decisions
Behavior (Sequence Diagrams)
Uses
Object Model
Who
Component
Contract
System Subsystem Specification
Pub / Sub Matrix
Test Cases

Figure 1. MATREX SDD Operational View.

Figure 1 shows the overview of the MATREX SDD which captures the system design within the MATREX Integrated Development Environment (IDE). The MATREX IDE is a content management system that provides various views into the MATREX system design. Since the SDD captures not only the object model information, but also the semantics of the data exchanges, test threads can be generated from the SDD and traced back to high level functional system requirements. In addition, the test generation process, which is described in the next section, uses a transport abstraction layer to allow these tests to be translated into various protocols, ensuring portability of the tests along with the models. Therefore, the tests evolve and migrate to new transport protocols with the SDD.

While developing mechanisms for object model integration is valuable, they are not always the best solution. By developing a way to map concepts and warfare functions to data elements and then mapping those data elements to various FOMs, we can directly map FOM elements to data within the warfare elements being represented within the M&S environment. If we can automate and generate the data transformations and middleware technical details, we can shorten the time and effort necessary to integrate models. The next sections will describe how this can be done by evolving the SDD and its export mechanism to the MATREX ProtoCore where the applications' data transformations are eased or replaced by the exported ProtoCore business logic and Application Programming Interface (API).

Finally, applying a manual integration and testing methodology to complex distributed simulation systems will produce errors, typically discovered too late in the schedule to be properly fixed. The following will describe the MATREX SE products and process, which enable the seamless interoperability between simulations in a distributed SoS test environment.

MIDDLEWARE ARCHITECTURE AGNOSTIC TOOLS FOR MODELING & SIMULATION

More and more middleware architectures are being used within the Army and DoD M&S community for various purposes. Existing infrastructures continue to be used for legacy simulations that have invested a lot of resources developing and maturing their environment. Switching middleware is difficult and expensive. The major three middleware architectures being used in DoD M&S are HLA, Distributed Interactive Simulation (DIS), and Test and Training Enabling Architecture (TENA). There are even version differences between standard versions, such as variants on HLA 1.3-Next Generation (NG) and HLA Institute of Electrical and Electronics Engineers (IEEE) 1516. Projects create their own changes to suit their needs resulting in branches within each of the already multiple choices. The M&S community continually diverges in its use of standards, tools and models. This costs the M&S community time and money porting a multitude of applications to different middleware architectures and their variants every few years.

Recognizing the continual technology advancements and community changes in direction, MATREX has strived to build middleware agnostic tools to support M&S projects across DoD. ProtoCore and the Advanced Testing Capability (ATC) are two such tools.

ProtoCore

ProtoCore provides simulation developers a single modern Object Oriented (OO) and type safe API (Snively, et al., 2006) to distributed simulation services. This API design makes it easier to use and less error prone than the native API of some supported protocols. The type safety allows more errors to be caught at compile time rather than runtime. The mechanism provided to connect the API to various network protocols is done via code generation from a common Object Model (OM). The underlying plug-in architecture used by the ProtoCore allows an application binary to run over various protocols without modifications. Legacy middleware architectures, used in many simulation environments, do not make use of modern programming practices and can be cumbersome and error prone to use. ProtoCore provides an object-oriented and event-based software library to ease the burden (to work on functions like data encoding) on software developers to use middleware architectures such as HLA.

Existing tools and applications are preserved, as they remain portable and easily deployable. Existing object models are preserved because they can be used with and migrated to newer protocols. ProtoCore currently supports plug-ins for HLA 1.3-NG, HLA 1516, TENA and OneSAF's Simulation Object Runtime Database (SORD).

As new functionality is added by and for the DoD M&S community, ProtoCore stays abreast providing the benefit of the improved functionality to all applications using ProtoCore. All users will benefit from the improvements driven by other users. ProtoCore is government-owned and available for use by DoD projects.

ProtoCore will be critical to our future plans for code generation and distributed simulation management and automation. More information on the path forward will be described later in this paper.

Advanced Testing Capability

The primary purpose of ATC (McCray, et al., 2008) is to provide model developers the capability to perform meaningful and repeatable "black box" analysis on individual models. The model, or system in this context, is treated as a black box; all other components, or actors in this context that would normally interact with the system are represented by the ATC. The system/actor relationship allows individual model component testing without having to stand up the whole simulation environment. Specific use cases can be explored helping bound the problem space when testing elements of a proposed federation. This makes debugging easier and lowers the cost of testing.

Through an advanced graphical user interface (GUI), the user is allowed to graphically define tests that may be saved and recalled at a later time or sent to other users. ATC also performs the function of documenting specific test cases in order to provide reproducibility. This allows for automation, and thus regression testing becomes much easier. The ATC allows the users to graphically create a sequence of actions or events to stimulate the system under test. The system responses generated by actor actions are validated. The ATC generates source code, which is then run to execute the test and verify results.

Enhancements have been made to integrate the ATC with our SoS SE approach and tools. Further improvements are planned and will be described later in this paper.

AUTOMATIC GENERATION OF ENGINEERING ARTIFACTS

Current Community Issues

There are many critical qualities that managers of a simulation environment must achieve: traceability from requirements to implementation and the resultant data collected; alignment of data semantics across applications; ease of maintenance; and change propagation throughout the architecture. Aligning data semantics is referring to ensuring applications are communicating based on a consistent understanding of the context and connotation of the information being shared.

When integrating existing applications that are chosen because they share a common syntax, or even for political reasons (e.g. someone with the authority orders the use of a model), the integration of applications must be backward engineered to the functionality required. Systems are often chosen because of the object model and middleware protocol that they are compatible with. However, compatibility is more than the ability to communicate without compilation errors or crashing. The applications' capability must provide necessary portions of a high level capability and they must provide that functionality in semantic harmony with the other applications within the architecture.

Events and exercises are notorious for making changes to the implementation throughout the integration and preparation. Most of the time, heavy change is still required up to only a few days or hours before the start of execution. Engineers often pull off technical miracles at the last second including working through the night or using one-time fixes that they know are not good long term solutions. Sometimes those changes work out, but frequently they are the cause of reduced availability, reliability and effective modeling.

Benefits of Automatic Generation of Engineering Artifacts

MATREX has made great strides in implementing some of the core building blocks for generative programming techniques (Czarnecki, et al., 2000) to the distributed M&S domain. It has become clear that there are exponential returns on investment when supporting the design and implementation of distributed M&S environments.

Capturing the Systems Engineering data within a database-driven infrastructure has allowed for full traceability from the top-level functional requirements through the design and implementation choices through to the detailed technical engineering artifacts used by all phases of the exercise implementation. The engineering artifacts include detailed technical requirements, systems engineering views for design discussions and even executable test cases. The next step will be to generate artifacts that can be used by the simulation and management applications as shown in Figure 2.

Top-Down Systems Engineering Benefits

Capturing the modeling requirements through a top-down decomposition ensures that the engineers understand the functions and information exchanges that are required to accomplish the high level modeling functions. Whether there are applications that can meet those needs or not, the engineering staff understands where there are weaknesses or workarounds necessary.
Applications and interface messages can later be allocated to the functional decomposition. The applications' ability to meet the functional requirements ensures traceability from the implementation back to the functional needs which can then be tied back to the purpose of the system as a whole.

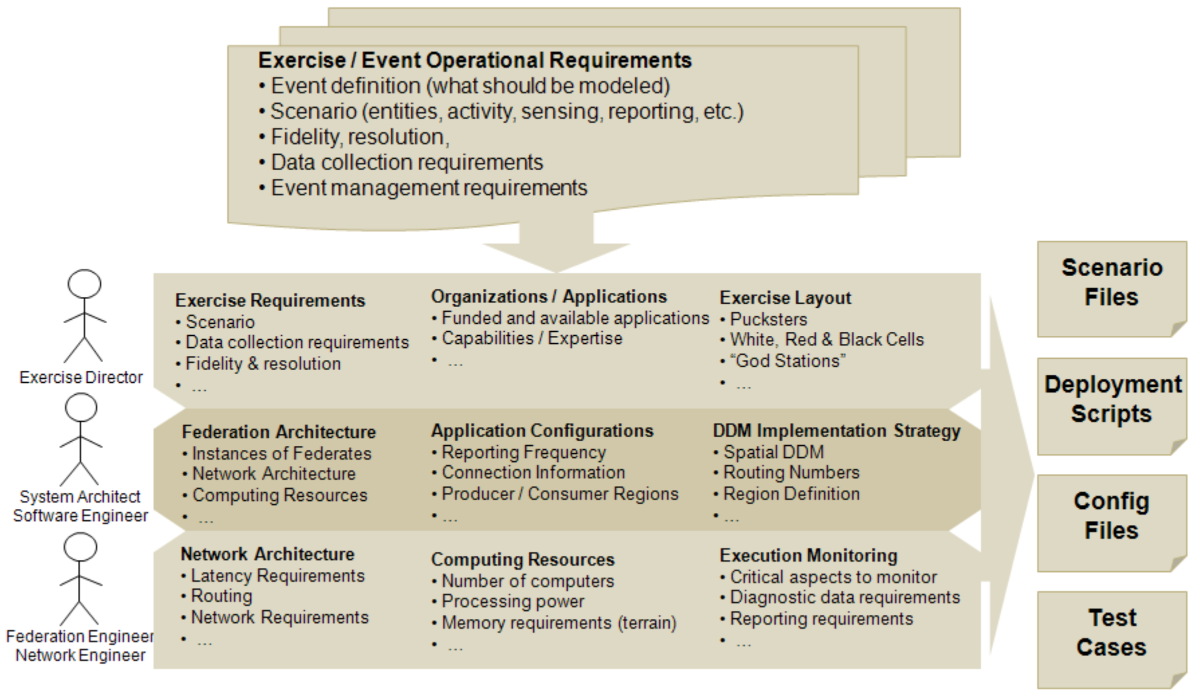
Exercise / Event Operational Requirements
• Event definition (what should be modeled)
• Scenario (entities, activity, sensing, reporting, etc.)
• Fidelity, resolution,
• Data collection requirements
• Event management requirements
Exercise Director
System Architect Software Engineer
Federation Engineer Network Engineer
Exercise Requirements
• Scenario
• Data collection requirements
• Fidelity & resolution
• …
Organizations / Applications
• Funded and available applications
• Capabilities / Expertise
• …
Exercise Layout
• Pucksters
• White, Red & Black Cells
• "God Stations"
• …
Federation Architecture
• Instances of Federates
• Network Architecture
• Computing Resources
• …
Application Configurations
• Reporting Frequency
• Connection Information
• Producer / Consumer Regions
• …
DDM Implementation Strategy
• Spatial DDM
• Routing Numbers
• Region Definition
• …
Network Architecture
• Latency Requirements
• Routing
• Network Requirements
• …
Computing Resources
• Number of computers
• Processing power
• Memory requirements (terrain)
• …
Execution Monitoring
• Critical aspects to monitor
• Diagnostic data requirements
• Reporting requirements
• …
Scenario Files
Deployment Scripts
Config Files
Test Cases

Figure 2. An approach for automating much of the event life cycle.

Code Generation for the Object Model

The MATREX ProtoCore includes a code generation capability to turn the FOM into a set of programming classes that can be used by application developers. By centralizing the generation of classes based on the evolving FOM definition, there is less reliance on application developers to all make the same changes accurately. This also saves time and eases developer participation within the environment. For instance, if every developer was relied upon to make the right changes and make them quickly, there would be a much larger management burden than just providing the new classes out to the developers in a single distribution.

Advanced System Black-Box Testing

The ATC described above stores test cases in an Extensible Markup Language (XML) file called the Test Case Markup Language (TCML). The XML file format is currently exported by the MATREX SDD so that ATC test cases are explicitly generated from systems engineering decisions and design captured in a systems engineering database. This alleviates a great deal of time for integration and testing staff by avoiding the need to manually change hundreds of tests and test processes due to a few small design changes. The code generation integration of ProtoCore with the ATC also means that object model changes are easy to adapt to over the evolving versions and instantiations of the MATREX architecture.

The ATC can test each system individually according to the design captured within the systems engineering infrastructure. These independent tests eliminate the complexity of a SoS architecture and can isolate interface details in an easy to execute testing environment. Since the tests are automatically generated from the design phase, test cases can be distributed to the developers the same day that the engineering decisions are being made. Development teams can code to the provided tests rather than spend their own time developing independent and possibly erroneous tests.

Test cases can be developed that test a subset of the systems to be integrated to increase the scope of the testing while maintaining an appropriate level of isolation from the complex SoS environment. Testing threads can further diagnose problems when integrating disparate applications built by numerous development teams.

We have also developed the capability to export sequence diagrams from the commercial tool MagicDraw and import the sequence into ATC for test case generation. This effort demonstrates the ability to pull in sequence diagrams in standard XML Metadata

Interchange (XMI) format from existing commercial tools to incorporate within DoD M&S community.

Architectural Design Agreements

In SoS environments like the MATREX, the Federation Agreements Document (FAD) (MATREX FAD, 2010) is used to explain architectural interoperability agreements to all the developers that need to integrate their applications. That document captures agreements on the use of coordinate systems, dead-reckoning and the heartbeat timing and distance thresholds for objects.

The MATREX ProtoCore software library is already used by many applications as their tool for interoperability with the simulation middleware protocol, so adding architectural compliance was the next obvious step, including capabilities such as coordinate conversions and dead-reckoning (Aronson, 1997). The additions can be made more flexible if they are driven by the systems engineering infrastructure. If architectural design decisions could be automatically driven by the systems engineers it would further decrease the software modification time and chance for discrepancies in application adjustments as the technical characteristics of the system are changed.

The design for dead-reckoning was kept dynamic so an operator could change the dead-reckoning distance and timing thresholds during run-time. This allows the operator to control the execution performance and accuracy from a central point. This can be used to recover the system from technical issues of slow performance, to accelerate the scenario without flooding the network or to change the dead-reckoning attributes of forces based on spatial considerations such as prioritizing updates for entities within an area of interest for the analysis of the scenario.

APPLICATION OF AUTOMATED MATREX CAPABILITIES

The MATREX program has made major strides in the execution of its objective goals. This section will walk through an example to demonstrate current progress and to illustrate the current system.

The design information is captured in a relational database and focused on components and the events between components. The SDD traverses the information within the database to automatically draw the sequence diagrams. This alleviates the need for a human to draw a picture and upload it to the system. The user can simply change the interaction or component allocation information and the changes are automatically seen within sequence diagrams.

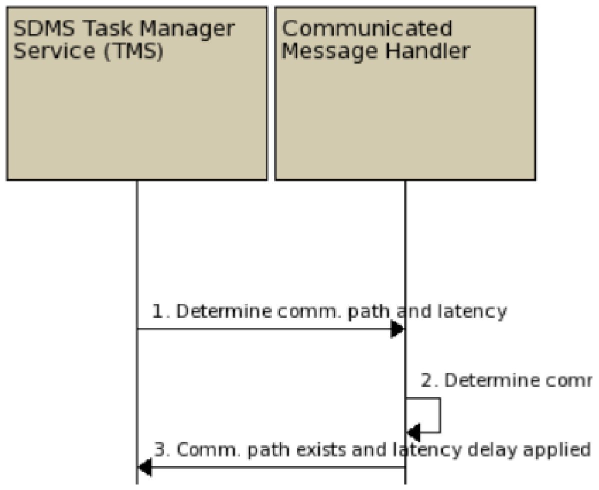
SDMS Task Manager Service (TMS)
Communicated Message Handler
1. Determine comm. path and latency
2. Determine comr
3. Comm. path exists and latency delay applied

Figure 3. Sample functional sequence diagram in the MATREX SDD

The sequence diagram above in Figure 3 is the functional view which is void of any implementation details. This can be developed without knowledge of the technical solution and be focused on the decomposition of the functional requirement. The sequence diagram below in Figure 4 is the allocated version of the same sequence diagram. This has models assigned to some components and abstract components that are linked to other discrete sequence diagrams. The example here shows interactions of the Sensor Data and Management Services (SDMS) (Mayott, et al., 2010), which is currently being integrated into the MATREX environment, with an external communications effects server.

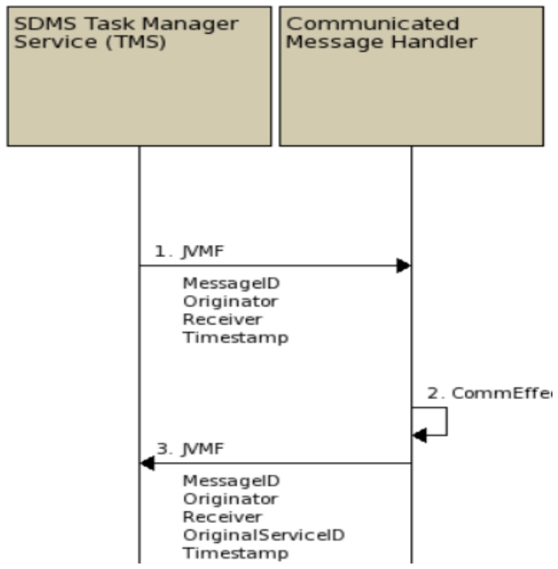

Figure 4. Sample component sequence diagram in the MATREX SDD

The SDD can generate the TCML file based on test set data assigned to this sequence diagram. Figure 5 shows a portion of the generated TCML file from the SDD sequence diagrams in figures 3 and 4.

```
<TestCase>
<Name>Determine communication path and latency</Name>
<Description/>
<ObjectModelName>fom_v6_0_31MAR2010</ObjectModelName>
<TCMLVersion>1.0-beta</TCMLVersion>

<Component>
   <Name>SDMS Task Manager Service (TMS)</Name>
   <Notes>The TMS serves as a Gateway between the Sensor Data Management
   <OrderIndex>1</OrderIndex>
   <Stereotype_ATC>System</Stereotype_ATC>
</Component>
<Component>
   <Name>Communicated Message Handler</Name>
   <Notes>The generic function for all functions that send communicated
   <OrderIndex>2</OrderIndex>
   <Stereotype_ATC>Actor</Stereotype_ATC>
</Component>
<Event>
   <Name>Determine comm. path and latency</Name>
   <Type>INTERACTION</Type>
   <EventID>707</EventID>
   <Notes>Determine comm path exists and latency</Notes>
   <Originator>1</Originator>
   <Receiver>2</Receiver>
   <Delay>0</Delay>
   <MinWait>60</MinWait>
   <MaxWait>300</MaxWait>
   <OrderIndex>1</OrderIndex>
   <TestDataSet_ATC>
      <Name>Determine communication path and latency - Test set 1</Name>
      <OMType>interactions.JVMF</OMType>
      <Attribute>
         <Name>Originator</Name>
         <Value>
            <ValidationFlag>false</ValidationFlag>
            <Value>TMS-1</Value>
         </Value>
      </Attribute>
```

Figure 5. Test Case Markup Language File

The TCML file is ingested by the ATC, which will allow the user to either tweak the test case and/or generate a working HLA federate. The federate publishes the prescribed stimuli and validates the model under test send the prescribed response. Figure 6 shows a screen shot of the ATC with this TCML file loaded.

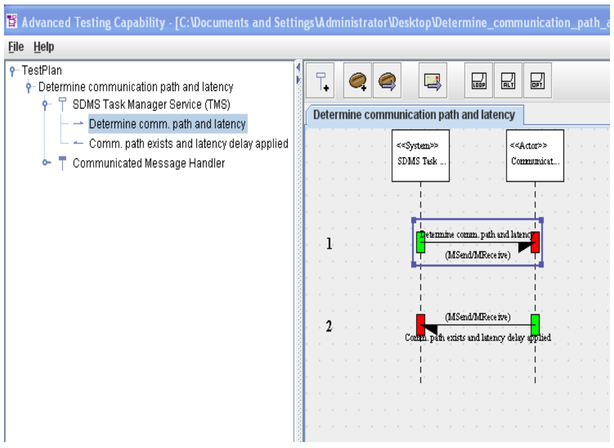
Advanced Testing Capability - [C:\Documents and Settings\Administrator\Desktop\Determine_communication_path_
File
Help
TestPlan
Determine communication path and latency
SDMS Task Manager Service (TMS)
Determine comm. path and latency
Comm. path exists and latency delay applied
Communicated Message Handler
LOOP
ALT
OPT
Determine communication path and latency
<<System>> SDMS Task ...
<<Actor>> Communicat...
1
Determine comm. path and latency
(MSend/MReceive)
2
(MSend/MReceive)
Comm. path exists and latency delay applied

Figure 6. ATC User Interface

The self-addressed event is not included in the ATC because it is merely descriptive of the action to be taken internally by the right-hand component.

The federate that ATC generates can be edited to change the internal business logic of the federate to increase the complexity of the test and/or be the basis for a surrogate federate when a model is not available. The benefit of the developing surrogates based on an SDD export is that the interfaces are identical to the design decisions made during the Systems Engineering process. The surrogate federate will abide by the same interface as the model it is replacing. In the SDMS example, the passed test case will ensure the ability of SDMS to interact with this external communications effects server by publishing the interactions subscribed to by SDMS without SDMS needing to actually use the external server for all testing.

MATREX has completed the described work and will be continuing to improve the functionality of our process and tools to ease the implementation of distributed simulation. We describe some of our plans for the future in the next section.

A VISION OF THE FUTURE

Developing distributed simulation environments to answer analytical questions has proven difficult for decades within the DoD M&S community. The MATREX program has experienced large amounts of the interoperability challenges and witnessed community peers struggle as well. The program has made great strides in SoS SE for distributed simulation and plans on continuing to push the state-of-the-art for SoS SE for distributed simulation. Some plans and ideas for the future are described below.

An overarching goal of MATREX is to reduce the cost while increasing the accuracy of distributed simulation using modern software development and integration techniques. Tools for developing models and generating software already exist. MATREX will apply proven tools and techniques to distributed modeling and simulation.

Integrated Application Deployment

It is currently possible to remotely deploy applications to lab machines and remotely launch applications based on a set of well defined configuration files. MATREX plans to expand this capability with a look at virtualization technology to improve the automation and flexibility based on the systems engineering design and configuration choices.

The deployment of models will need to be flexible to change with each purpose for the modeling and simulation instantiation. The models that need to be deployed, the number of instances required, the types of machines, the network topology and many more details are driven by the scenario, functions performed and the models themselves. Linking the application details with scenario qualities and functional interoperability will be the key to automating the deployment of the simulation environment.

System Configuration

Configuration of the participating systems within a SoS architecture are both application-specific as well as based on the scaling strategy and scenario that will be executed. By capturing and linking application configuration requirements to scenarios and functional requirements for the architecture, configuration options are mapped to the appropriate use and the proper configuration is automatically exported depending on the scenario and functions the systems engineers require for any given instantiation of the architecture.
The M&S infrastructure, such as the middleware and its configuration, is also based on the scale and architecture of the implementation required. As engineers are making design decisions within the systems engineering infrastructure, the middleware configuration is predictable and can also be automatically exported from the systems engineering tool.

Integrated Scenario Development and Initialization

The MATREX SDD is currently limited to the functions required rather than the scenario in which those functions will be used. To facilitate deployment and configuration becoming automated, the basis for both must be captured in our SoS SE infrastructure, the SDD. As scenario information is added within the SDD, predicting the appropriate system configuration to execute the scenario becomes more robust. The SoS environment can also be remotely initialized from a central point.

The MATREX design pattern for simulation initialization is to provide the scenario details, such as initial platform attributes and force laydown, over the simulation middleware at the beginning of the exercise. The structure of the information and the design paradigm are already in place to expand the SDD to incorporate scenario information and export the necessary engineering artifacts to configure simulation management tools, such as simulation initialization, data collection and execution monitoring (Kolek, et al., 2000).

A subtle benefit for incorporating the scenario information within the systems engineering tool is that in many cases the functional design depends on how the scenario will be executed. Similar to how military operations depend on the mission, the execution of the mission with M&S applications also depend on many parameters, many of which are based on the scenario.

Active Design-Based System Monitoring

Active monitoring and system management can help engineers recognize issues early so they can fix them without a lot of wasted execution time and cycles. As more information is captured about the execution environment, more information will be available to recognize when the system is performing adequately and when the system is beginning to fail.

Monitoring information exchanges at run-time ensures that the implementation does not deviate from the design. Part of the monitoring includes monitoring performance via response time of applications, queue lengths of applications sending and receiving information and machine diagnostics such memory footprints and processor loads.

Surrogate Model Generation

MATREX already has the technical ability to generate working applications within the ATC. Those applications can be generated to subscribe and publish object model elements based on the sequence diagrams defined within the SDD. The next logical step will be to expand the generation of the test federates to include more complex execution behaviors such as the generation of working code blocks within the generated federates from pseudocode (Roy, 2006).

Data Collection and Mapping to Requirements

In order to deploy, configure, initialize and monitor the execution of the simulation environment, high level requirements (the purpose of the simulation) must be mapped to the technical design and ultimately to the object model details and application details (Fogus, et al., 2006). The data collection plan is based on information that is already available to the system: object model object, interactions, parameters and attributes that are required in order to have the functional capabilities implemented.

There is then enough information to tie object model elements collected with the high level functional requirements. MATREX does not organically manage a data collection and analysis tool but does use a few that can be given the data collection requirements and ultimately the information necessary for the system to provide an analyst focused and direct data results from the simulation run.

The DoD Standard Practice Documentation of V&V for Models and Simulations (DAU, 2008) is the basis.

User Interface to an Automated M&S Environment

One of our objective goals is to provide the previously explained SoS SE infrastructure with an interface easy to use and instantiate the desired M&S environment rapidly and provide the resultant data in the user's inbox overnight. The intended interface will be a directed interview akin to the Turbo Tax® interface. The choices for the users will be derived by their choices to previous steps within the process.

The implied requirements for this goal are to have machine understanding of the warfare functional capabilities and their mapping to the possible technical solutions, the interoperability of multiple technical solutions, and the implementation of the solution through automated deployment, configuration, initialization and data collection of the relevant information and its application to the analyst use cases.

Ultimately, as the automation increases throughout the process, the ability to execute analysis events without large efforts from engineers improves. The goal is to allow an analyst to use this systems engineering tool to design, deploy, configure and manage the SoS architecture based on accredited models. The execution run could occur on a representative set of lab machines that can be setup on the fly to accommodate the captured execution configuration. These machines could be at various geographic locations with results compiled and sent to the analyst automatically when complete.

These are non-trivial accomplishments that MATREX is looking forward to tackling. Preliminary design plans exist for execution in FY11 and beyond and the program plans to report its successes and lessons learned in future publications.

ACKNOWLEDGEMENTS

This paper is based on work executed over the last four years for which we'd like to acknowledge some of the personnel most involved in its development and maturation. Those personnel are: Sid Antommarchi, Tracey Beauchat, Jon Clegg, Jim Gallogly, David Itkin, Rich Leslie, Richard Mangieri, Lee Mangold, Tom McAfee, Paul McCray, Rachel Offutt, Keith Snively, Jeff Swauger, and John Vintilescu. We'd like to also acknowledge our management personnel that helped provide vision, resources and encouragement: Chris Bailey, Tom Hurt, Tom Mathis, Joe McDonnell and Dave Poole. Finally, we'd like to acknowledge the Office of the Assistant Secretary of the Army (Acquisition, Logistics, and Technology) and RDECOM for providing the support necessary to continue these critical efforts.

REFERENCES

Acquisition Community Connection at Defense Acquisition University (DAU). 2008. *MIL-STD-3022 DoD Standard Practice Documentation of V&V for Models and Simulations*. Available via https://acc.dau.mil/CommunityBrowser.aspx?id=205916.

J. Aronson. 1997. *Dead Reckoning: Latency Hiding for Networked Games*. Available via http://www.gamasutra.com/view/feature/3230/dead_reckoning_latency_hiding_for_.php.

K. Czarnecki and U. Eisenecker. 2000. *Generative Programming: Methods, Tools, and Applications.* 1st ed. Addison-Wesley Professional.

Department of Defense (DoD) Chief Information Officer. 2009. *DoD Architecture Framework 2.0*. Available via http://cio-nii.defense.gov/docs/DoDAF%20V2%20-%20Volume%201.pdf.

M. Fogus, H. Borum, D. Prochnow and K. Penn. 2006. *MATREX Data Collection and Analysis: Linking Simulation Results to Military Analyst Requirements.* Simulation Interoperability Workshop Fall Conference, September 2006, 06F-SIW-048.

Institute of Electrical and Electronics Engineers. *Standard for Modeling and Simulation (M&Amp;S) High Level Architecture (HLA) - Framework and Rules.* Available via http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=893287.

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

S. Kolek, S. Boswell and H Wolfson. 2000. *Toward Predictive Models of Federation Performance:*

Essential Instrumentation. Simulation Interoperability Workshop Fall Conference, September 2000, *00F-SIW-085*.

MATREX Federation Agreements Document. 2010. Available via https://www.matrex.rdecom.army.mil.

G. Mayott, W. Self, J. McDonnell and G. Miller. 2010. *SOA approach to Battle Command to Simulation interoperability*. SPIE Defense, Security, and Sensing, April 2010.

P. McCray and K. Snively. 2008. *Functional Component Testing for Distributed Simulations*. Simulation Interoperability Workshop Spring Conference, April 2008, 08S-*SIW*-063.

G. Roy. 2006. *Designing and Explaining Programs With a Literate Pseudocode*. Journal on Educational Resources in Computing (JERIC) by Association for Computing Machinery (ACM). Volume 6, Issue 1 (March 2006). ACM, New York, NY USA.

K. Snively and P. Grim. 2006. *ProtoCore: A Transport Independent Solution for Simulation Interoperability*. Simulation Interoperability Workshop, September 2006, 06F-SIW-093.

A. Tolk and J. Muguira. 2003. *The Levels of Conceptual Interoperability Model*. Simulation Interoperability Workshop Fall Conference, September 2003, 03F-*SIW*-007.

J. Tufarolo, R. Leslie and D. Lewis. 2004. *Distributed Integration for the V0.5 MATREX*. Simulation Interoperability Workshop, Spring 2004, 04S-SIW-131.

D. Van Hook and J. Calvin. 1998. *Data Distribution Management in RTI 1.3*. Simulation Interoperability Workshop Spring Conference, Spring 1998, 98S-SIW-206.