Decreasing Computational and Operator Complexity with Aggregate Entity Interactions

Daniel J. Lacks, PhD, John Stevens, Rod Choi, Kevin Kelly
Cole Engineering Services, Inc.
Orlando, Florida
Daniel.Lacks@cesicorp.com, John.Stevens@cesicorp.com, Rod.Choi@cesicorp.com, Kevin.Kelly@cesicorp.com

ABSTRACT

Historically, whether for training or analysis, there has been a tension between the desirable detail of entity simulations and the reduced overhead of aggregate simulations. Often both are used – entity to provide detail in areas of particular interest and aggregate to provide an appropriately complex battlespace. When so used, care must be taken to control the interactions between entity and aggregate simulations due to inaccurate or even impossible interactions. For example: entities often cannot sense aggregates while aggregates often use completely different engagement models (fire/detonate vs Lanchester's equations). In this paper we present an innovative approach to aggregate simulation based on validated entity performance data. Our approach provides low overhead computation and operation while allowing entity level mobility, sensing and engagement compatible with more detailed entity simulations. We also present results of using this methodology during test and user events.

ABOUT THE AUTHORS

Dr. Daniel Lacks is the Chief Scientist at Cole Engineering Services, Inc. (CESI) and Conceptual Modeler on the One Semi-Automated Forces (OneSAF) Integration and Interoperability Support (I2S) program. He received Computer Engineering BS (2001), MS (2002), and PhD (2007) degrees from the University of Central Florida specializing in distributed computing, software simulation, and software engineering. Dr. Lacks has spent the last 15 years working as a software and systems engineer in the DoD M&S industry on Live, Virtual, and Constructive programs to include Warfighter's Simulation (WARSIM), Joint Simulation System (JSIMS), One Tactical Engagement Simulation System (OneTESS), Joint Land Component Constructive Training Capability (JLCCTC), Live Virtual Constructive-Integration Cell (LVC-IC), Live Virtual Constructive-Integrating Architecture (LVC-IA), and OneSAF. Dr. Lacks lead the final CESI Internal Research and Development (IRAD) initiative to complete the final implementation phase and prepare the Unit Level Entity (ULE) baseline for handover to the OneSAF program.

Mr. John Stevens is the CTO and co-founder of Cole Engineering Services Inc. He has worked in the simulation industry for more than 20 years. He architected the FOM agnostic HLA version of OTB as part of JVB and MATREX-participating in the 2002 C4ISR event at UAMBL. He is the principle concept developer of advanced concept constructive models including Goal Based Behaviors and Unit Level Entities.

Mr. Rodney Choi is currently a Sr. Systems Engineer on the Integrated Live-Virtual-Constructive Test Environment program. He is a retired Marine officer who served in the infantry and communications-data occupational fields. He holds a BS from the US Naval Academy and an MS in computer science from the Naval Postgraduate School. He has worked on a variety of projects for CESI including the ULE development effort.

Mr. Kevin Kelly is the Lead Systems Architect for the One Semi-Automated Forces (OneSAF) Integration and Interoperability Support (I2S) program. He holds a BS in Computer Engineering from Purdue University and a MS in Modeling and Simulation of Intelligent Systems, from University of Central Florida. He has been in the simulation industry for 18 years working primarily on WARSIM and OneSAF. He helped design many of the original modeling frameworks for the OneSAF program. He wrote an early paper on the ULE concept and developed the original prototype. He also oversaw the integration of the capability into the OneSAF program.

Decreasing Computational and Operator Complexity with Aggregate Entity Interactions

Daniel J. Lacks, PhD, John Stevens, Rod Choi, Kevin Kelly
Cole Engineering Services, Inc.
Orlando, Florida
Daniel.Lacks@cesicorp.com, John.Stevens@cesicorp.com, Rod.Choi@cesicorp.com, Kevin.Kelly@cesicorp.com

PROBLEM STATEMENT

As simulations evolve over time to perform training, test and evaluation, analysis, intelligence, acquisition and experimentation activities, the user community desires for an expansion of capabilities and force size to model complex scenarios. In cases for some simulations, increasing the force size is not as simple as adding additional computational hardware as there are limits to data structures, network communications, and other factors which prevent scenario growth. Likewise as force size is increasing, the user community is facing pressure to limit or reduce the number of simulation operators available to run a simulation. The Unit Level Entity (ULE) capability presents a technique to expand the simulated force size and reduce the computational footprint by aggregating the entity modeling into a single entity which controls and queries its subordinate entities. This technique allows for reducing user operations of units in some cases while providing increased fidelity typically not available in aggregate simulations.

The initial impetus for the ULE effort was representation of "wrapper" forces for larger events. The objective is to provide a means to represent the adjacent "wrapper" units realistically without requiring excessive resources. One common solution is to interoperate with another simulation. The primary forces of interest are simulated on the primary simulation while the wrapper forces are modeled, possibly as aggregates, on another simulation that is joined via an appropriate protocol. The activity of interoperating simulations causes problems because it is labor and resource intensive, it introduces fair fight issues, and it may not provide entity ownership transfer between simulations. The ULE use case combines wrapper and primary forces into a single simulation modeling them at different resolutions. Using a single simulation alleviates technical concerns such as mismatched protocols or data models, non-correlated terrain, differing models, common exercise and technical control tools, and differing time management policies.

Large Scale Simulation Events Needed to Train Staff

Military simulations have been used for generations by military professionals to train tactical and strategic thinking (US Army, 2008). These have ranged from board games (ala chess or go) and rock drills to the highly complex and sophisticated computer based simulations in use today. What is traditionally viewed as entity simulations are developed to support lower level echelon training such as individual trainees in aircraft and vehicle simulations up through platoon commanders and unit collective training such as Aviation Collective Combat Tactical Trainer (AVCATT) and the Close Combat Tactical Trainer (CCTT). Aggregate simulations such as Core Battle Simulation (CBS) and EAGLE represent higher echelon training. Joint Land Component Constructive Training Capability (JLCCTC) Multi-Resolution Federation (MRF) is a hybrid approach using combinations of entity and aggregate federated system of systems. For upper echelon level staff training, details below two echelons are generally not needed and simple models are quite sufficient. The simplicity drives a number of factors discussed below.

Computational Limits

Initial simulation systems were built and executed using human operators and probability data. Lanchester's Equations (Joyner, 2007) address this type of war gaming. Calculations could be as simple as dice (remember the Avalon Hill war games of the 1960's and 1970's? (Wikipedia, 2016)). As computers became available, portions of these "games" became automated, but the computers were quite limited in memory and computational capability. Initially only individual computers (pre-networking) were used and the models had to be simple enough to execute in usable (staff level training may be compressed for clarity) time. In the 1980's (Strickland, 2011) the beginnings of entity level autonomous behaviors for individual trainees began to be introduced with programs such as the General Electric

Advanced Maneuvering Logic (AML) model for air-to-air combat. The load on existing computers was prohibitive at first, but computers were getting faster.

Networking multiple computers together introduced the ability to divide and conquer the problems and distributed simulation was born (ibid.). Early efforts to connect individual simulators into groups included the development of SIMNET (evolving to the DIS standard we have today). Now groups of simulators could be gathered into exercises and units could be trained, but the OPFOR still needed to be operated by people. Eventually computer models of humans were introduced by systems like the Modular Semi-Automated Forces (MODSAF) and JCATS. Individual computers could simulate a few 10's of vehicles or a few aircraft. These platforms evolved with the computing hardware but large scale simulation events required lots of hardware (a single brigade size event in the early 2000's required over a hundred Linux computers to simulate.

Operator Cost

At each of these computers is an operator. Entity level simulation provides great detail, and as combat effectiveness of individual systems grew, this level of detail was increasingly necessary even at the command staff level. UAS systems have only added to this problem by making individual entities not only visible but critical at the battalion and brigade TOC. But skilled operators have always been required to handle things the computer models cannot.

Mixed Resolution

The networking and federation of simulation models led to mixed resolution federations. The idea is simple and makes good sense: use entity level simulation in areas where the details matter and aggregate to lower to computational and operator overhead detail is not as important. An example of this type of approach was the C4ISR Experiments conducted at the Unit of Action Maneuver Battle Lab (UAMBL) on Ft. Knox during 2002. The "wrap" providing context was provided by the aggregate level simulation EAGLE and the entity level simulation was provided by the Joint Virtual Battle Space federation using the OneSAF Testbed Baseline (OTB). This provided the span and detail needed to successfully conduct the experiment.

Different Models

But it wasn't easy. EAGLE used a very simplistic set of timed models for mobility, area sensing probabilities, and Lanchester based attrition models. OTB used vehicle and dismounted soldier models with ACQUIRE based sensing, soil, slope and terrain based physics mobility and individual weapon target paired PH/PK attrition models. In early testing runs EAGLE units could drive right through OTB units with neither sensing nor engaging them.

The solution for the UAMBL experiment was a geographic division of entity and aggregate players. The exercise planners chose terrain where the entity level "game" was played out in a central valley while the aggregate wrap was played out in the adjacent valleys. The mountain ranges provided a natural buffer to avoid having the aggregate and entity meet. This is a common way to separate the incompatible models to avoid unrealistic outcomes. Also employed are operational divisions: aggregate models for indirect fire units, for example, and entity based for maneuver units. It is rare for maneuver units to actually encounter indirect fire units on the battlefield. Finally, domain divisions may be employed. If the primary training or analysis target is ground, air & maritime may be aggregate or low resolution and maneuver units entity or high resolution.

Artificial Exercise Constraints

The problem with these approaches is that they introduce artificial constraints into the exercise which may affect the results. Commanders are not allowed the full freedom to innovate tactics. In the UAMBL example cited above, the commander was constrained to the central valley. If a flanking tactic had been employed, the exercise would have been stopped due to interoperability problems.

SOLUTION

Our solution to the issue of balancing aggregate and entity simulation fidelities into a single simulation is the ULE. The concept applies to extending either an aggregate or entity simulation to support ULE. It preserves the "traditional"

existing task organization structure to allow transition between ULE and traditional modes. In ULE mode, extra software supports the existing infrastructure for balancing the level of detail in simulation while allowing certain traditional aspects to turn off their modeling capabilities in order to be more efficiently controlled by the ULE. ULE modeling is turned off and traditional modeling is turned back on when transitioning out of ULE mode. This approach allows for using the existing simulation data (engagement, vulnerability, etc.) without having to create special information for the ULEs.

Solving the problem of representing aggregated and entity forces is both an art and a science. From the art perspective, the presentation of ULEs such as fire lines, formations, supply data, etc. not only had to be accurate, but needed to be represented in a way that was honest in presentation to the user and the simulation. For example, does a fire line originate from the ULE or from the platform that is shooting but is invisible to user? Does the fire line end at a ULE or at the entity being shot at who is not visible? While the simulation handles "who shot who" without ambiguity, the presentation of that data gets confusing when aggregate icons are displayed especially when mixed with entities and aggregates in cross-resolution firefights.

From the science perspective, the entity simulation is expected to be highly accurate and correct due to the breadth and depth of high fidelity authoritative data sources and algorithms; however aggregation saves compute power possibly at the cost of fidelity. For example, if we sense from one entity in a unit rather than each entity in a unit, should the sensor be placed with the unit leader or the center of the unit? What if the unit leader is in the middle of a formation, on the left or right, blocked by trees while the rest of the unit is exposed? We approached these types of issues in a way that balances accuracy, realism, and presentation while still preserving the application of validated algorithms and data.

ULE Concepts

The following concepts apply to simulation from the perspective of ULE modeling regardless of the simulation the ULE concept is implemented within.

- **Simulation Engine.** Manages, at a minimum, the creation, destruction, distribution, time management, processing time, data, persistence, fault tolerance, and communications for simulated objects and agents.
- **Agent.** Facilitates the modeling of simulated objects.
- **Simulated Object.** Anything that can be represented in simulation that is managed by a simulation engine.
- **Entity.** A simulated object capable of modeling itself and/or modeled by an agent.
- **Entity Simulation.** A simulation which primarily represents and manages at an entity level of detail even when the entities are combined into units and aggregates. Exceptions are made, for example, when a unit level order is simulated differently than issuing the same entity level order to the unit's individual entities.
- **Traditional Entity.** A simulated object actively modeling itself or modeled by an agent other than a ULE.
- **Ghost Entity.** An entity modeled by a ULE as a puppet master or inquisitor.
- **Puppet Master.** The ULE controls ghost entity subordinates rather than the ghosts controlling themselves.
- **Inquisitor.** The ULE queries ghost subordinates for data rather than accessing or storing data directly. This allows ghosts to be reactivated when switching between ULE and traditional modes.
- **Unit.** Pertains to an organization of personnel and platforms as represented in any hierarchical fashion indicating a command structure. Units are composed of subordinate entities.
- **Traditional Unit.** A simulated unit not in a ULE mode of operation.
- **Aggregate.** A group of units and/or entities which may or may not match a unit's organization.
- **Aggregate Simulation.** A simulation which primarily represents and manages at the aggregate or unit level of detail including circumstances when an aggregate or unit only contains one entity. Exceptions are made, for example, for specialized lifeforms and platforms whose tasks are not representative of the aggregate representation of its unit or who are typically tasked individually in an "entity mode."
- **Unit Level Entity (ULE).** An entity representing a unit's agent capable of controlling (as a puppet master) or querying (as an inquisitor) subordinate ghost entities in order to reduce computational power and user operations. The ULE balances the subordinate level of detail between aggregate and entity level of simulation in order to produce accurate and realistic representations while improving performance compared to entity level simulation and fidelity compared to aggregate level simulation. Incompatible unit compositions exist, for example, when performance degrades compared to traditional entity modeling.

- **ULE Unit.** The unit coordinating with a ULE within the software.
- **Representative Entity**. Used for determining mobility and sensing capabilities. The representative entity type is calculated based on an algorithm of the unit composition of subordinates; it is generally the most common platform or non-crew lifeform in the unit. For a simple example, an M1A1 tank platoon ULE has a representative entity type of M1A1 because it consists of four M1A1 tanks. The approach for determining the representative entity type is described in the aggregate representation section below.
- **Wrapper Forces**. Provide context for the primary force of interest. For example, in a battalion-level training event, the battalion being trained would likely be adjacent to four to five more maneuver and supporting battalions from its parent brigade. The primary ULE objective is to provide a means to represent the adjacent units realistically without requiring excessive computational resources.
- **Entity Count vs**. **Force Size**. Entity count is the number of entities. Force size represents the number of entities that compose an organization of military, paramilitary, insurgent, or other ad hoc forces.

Aggregate Representation

The ULE concept can ideally be implemented in any military echelon based simulation. If an aggregate simulation is chosen, then the entity detail may need to be worked on in order to de-aggregate into an entity mode where the entities are controlled and queried by the ULE. Likewise, the aggregate capabilities must be improved in an entity simulation to represent units. We chose to implement the ULE concept using the OneSAF simulation because it is a Government Open Source Solution (GOSS) used by a wide audience of training, test and evaluation, analysis, intelligence, acquisition and experimentation communities (PEO-STRI, 2016). OneSAF already has rich entity level data and the constructs to support unit representation. OneSAF units perform just as well as entities because the units are just a collection of entities; but there is no computational benefit by entities being grouped into a unit, and there is no loss of capability either. A modeled entity can be configured to be represented at various resolutions such as low, medium, and high; this concept is extensible to include a new ULE representation. OneSAF units have a limited set of behaviors, but can be controlled with basic capabilities. By understanding the existing ground rules of entity and unit simulation in general, ULE ground rules emerge:

Traditional entity and unit capabilities are unchanged. ULE adoption cannot contaminate the way the simulation works when ULEs are not played. This is particularly true because OneSAF models are formally validated. Our approach cannot break validation of existing traditional models, and we also want to leverage the traditional entity models for ULE. ULE builds on top of existing only code changing what is necessary and leveraging what is already there as applicable.

Reuse traditional capabilities. ULEs reuse the existing traditional modeling capabilities as much as possible and in some cases exactly. OneSAF provides flexibility to change or keep what is necessary to make ULE successful. Some capabilities, like sensing, start out as a ULE capability when units are in ULE mode, but once entities are sensed, the traditional sensing model is used while still in ULE mode. Vulnerability is always calculated by the entity in ULE or traditional unit mode. Supplies are tracked in the entity, but supply counts are controlled by the ULE in ULE mode.

Improve performance. ULE subordinate entities disable modeling capabilities as their modeling fidelity is controlled by the ULE when in ULE mode. Since ULEs must not contaminate OneSAF unit capabilities, a new ULE software class is created adding a new entity to counterpart the unit class. But, while the overall entity count increases by one, the count of entities consuming CPU in order to model decreases for each ghost entity. For example, suppose a tank platoon of four tank entities is now converted to a ULE. There are now four tank entities and one ULE entity counted, but the four tank entities are just data containers and the ULE is the only entity modeling. If the ULE needs to calculate something on behalf of the entity, it either instructs the entity to do it as a puppet master or asks the entity for data as an inquisitor. Two examples are (1) the ULE instructs the entity to move and (2) the ULE asks the entity what the damage outcome is from a particular engagement.

Provide a fallback plan. In the case of ULE being a new feature, if a bug or inconvenience is getting in the way of a user performing tasks, we allow the user to be able to switch back to traditional unit and entity mode in real time to achieve mission success. This approach also gives breathing room for the developers to identify limits of ULEs. For example, in most cases it makes sense to not use ULEs in urban operations.

Minimize user training. We want ULEs to look like and operate like units to the operator, but we do not want to trick the operator for confusing traditional higher fidelity OneSAF units with ULEs. Both icon representations use MIL-STD-2525-B (Department of Defense, 1999), the ULE icon includes a health bar and an orientation arrow. The user can select ULE icons to see the ghost entity representation and formation where the entity color is combined with white in order to give a ghostly look to the icons versus their counterpart modeled icons.

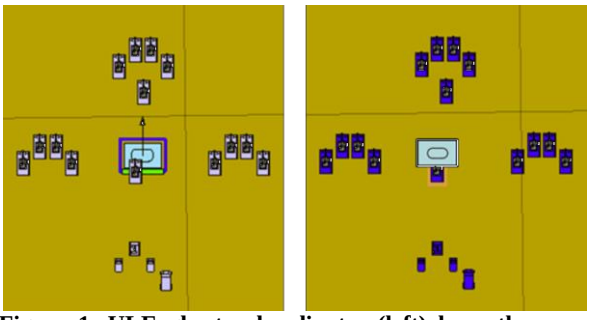

Figure 1. ULE ghost subordinates (left) have the same color as their traditional entity counterparts (right), but are blended with the color white to appear most ghostly. This assists with providing seamless integration.

Seamlessly integrate ULE. ULEs are optimized to be created at any point in time from any unit and vice versa, so long as the unit is compatible with ULE.

In order for the ULE to perform more efficiently than traditional units, trade-offs are made at the modeling level to save CPU and memory consumption while still maintaining results of validated models and data. A traditional unit is represented with a collection traditional entity subordinates where the entities perform the modeling in an entity simulation. The ULE capability instantiates as a unit-level entity with traditional subordinate entities (ghosts) modeling capabilities turned off. The ghosts help with tasks like declaring an entity composition, holding entity state, and performing passive mobility, sensor, weapon and vulnerability calculations against the validated set of OneSAF parametric data. The ULE created is composed of specialized ULE mobility, sensor, weapon, and vulnerability aggregate models which facilitate puppet master and inquisitor subordinate ghost entity modeling. Without ULE, each subordinate traditional entity would have one or more active mobility, sensor, weapon, and vulnerability models rather than aggregating these active models one time within the ULE.

Each ULE has a representative entity type for reducing sensing and mobility computation. Choosing representative entity types is complex for heterogeneous unit compositions, but is simplified for the user when the echelon representation is chosen for converting units to ULEs. The unit and each subordinate entity must have compatible compositions with the conversion semantics in order for the unit to convert to a ULE: ground or lifeform, with or without weapons, and weapons must be direct fire weapons. ULE units as implemented in OneSAF cannot be composed of incompatible entity compositions including fixed wing aviation, rotary wing aviation, sea surface, sea sub-surface, space, and indirect fire entities. The representative entity type cannot be designated with entities that cannot move on their own (like trailers or supply caches). The user is alerted when a unit cannot convert to a ULE to minimize user training.

Supplies

The ULE capability accounts for supplies at the unit level making decisions based on supply levels at the unit level, but the individual supplies are accounted for in the ghost entities. This technique of keeping the supplies on the entities allows for seamless switching between ULE and traditional unit modes so that the subordinate entities keep their supply levels. Also, if a ghost entity is retrieved for medical or maintenance operations, the proper supply levels can be accounted for when the subordinate returns to duty with or without resupply.

Interoperability

The ULE approach leverages the OneSAF InterOp to communicate with other simulators such as virtual simulators, UAVs/stealth viewers, and other entity and aggregate simulations. An option is implemented to enable whether or not to publish ghost entities to external simulators. ULE logic assists in determining which ghost entity is targeted by an external entity (a UAV for example) so that damage can be dealt to the correct ghost entity when ghost entities are published to external simulators. When the ghost entity publication is enabled, the ghost formations are reflected in the same manner that the ULE manages the formations with the puppet master paradigm. This allows external simulator operators to identify the unit formation even though the fidelity of the formation is simplified. ULE interoperability assumes that the receiving system will ground clamp the entities.

Sensing
The ULE sensor model inherits capabilities from a validated acquisition model, but bases its sensing capabilities on the best sensor available in the unit represented to capture perception information for an entire unit when possible. The best sensor is determined based on the range, illumination, and atmospheric conditions. This single sensor is used by the ULE to sense to save performance. If a ULE is sensed by another entity (traditional or ULE), the sensed ULE returns the representative entity type as the sensed entity type. However during the shooter and target adjudication and selection processes, the entities will sense with their sensors using data defined for those entities to provide realism and valid results during engagements.

Mobility
The ULE mobility model inherits capabilities from a high fidelity ground vehicle mobility model, but bases its movement operations on the ULE representative type. The puppet master approach allows the ULE to control the real time mobility of the subordinate ghost entities in formation with one agent and less computational processing power. This design saves the computation of modeling movement of each entity within formation, still accounts for fuel burn rate, but at a loss of some fidelity. A computational improvement is that the ULE mobility model covers the whole unit's formation control model at the unit level rather than each entity independently finding their location in formation based on formulas and offsets. ULE subordinates move at the same speed based on the representative entity type selected in order to stay in formation. The ULE icon is placed at the centroid of the formation which is only calculated when a unit converts to a ULE; the location of the ULE icon does not change if, for example, an entity suffers a mobility kill and is left behind or if the commander is wounded or killed.

Engagement
The ULE weapon model provides the framework needed to simulate the firing of all types of weapons and choosing a target. Each weapon has a model that supports a delivery accuracy model and a rate of fire model. Weapons provide functionality for aiming, firing, and selecting munitions. The ULE weapon model replaces the traditional weapon entity level model and neither improves nor degrades the computational performance of the ULE weapon compared to the traditional entity level weapon modeling. But, the ULE weapon model aggregates information by weapon type which would typically be modeled by individual shooter entities such as what subordinate entities have weapon fire capability, are not firepower killed, and are ready to shoot. It also manages munition counts for subordinate entities.

Vulnerability
The ULE vulnerability model adjudicates engagements between ULEs and ULEs, and between ULEs and entities where either combination is the shooter or target. When a detonation event is received by the ULE Vulnerability Model, it loops through all of the subordinate vulnerability models to find the best fit for the munition type shot to the target type and location. Vulnerability models are completely reused from existing validated data due to this approach and thus require no mapping from traditional entity vulnerability models to ULE vulnerability models. The ULE vulnerability agent only process detonations relevant for its subordinates. Ghost entities suffering from mobility or catastrophic kills are visible on the map to pick up for medical or maintenance activities.

Behaviors and Interventions
ULEs support the following behaviors (semi-automated), interventions (immediate actions), and magic behaviors (immediate actions requiring elevated permissions):

Table 1. ULE Behaviors, Interventions, and Magic Behaviors.

ULE Behaviors	ULE Interventions	ULE Magic Behaviors

• Move Tactically
• Detonate Explosives
• Explosive Emplacement
• Construct Fighting Position
• Minefield Emplacement
• Obstacle Emplacement
• Recon Ground
• Classify Bridge
• Clear and Mark Lane
	• Why Not Moving?
	• Why Not Sensing?
	• Follow Route
	• Fire Direct
	• Set Weapon Control Status
	• Change Orientation
	• While Moving:
	  ○ Speed Up
	  ○ Slow Down
	  ○ Halt
		• Heal
		• Mount/Dismount
		• Hitch/Unhitch
		• Set Weapon Control Status

• Patrol • Transfer Supplies		

Tactical Device Stimulation
Mission Command integration via the Mission Command Adapter-Web Service (MCA-WS) supports reporting unit location, situations observed, task organization, basic load and cargo supply types, and supply levels.

RESULTS

The ULE capability was first integrated into OneSAF version 8.0 as an introductory limited use capability. The OneSAF 8.5 release provides an initial ULE capability to primarily represent wrapper forces. ULEs were first used by users to facilitate an integrated evaluation of emerging concepts and capabilities. The user's scenario required a significant amount of OPFOR entities that pushed the force size beyond previous events and beyond what OneSAF recommends. They qualify as ULE wrapper forces to support mission command situational awareness for the surrounding OPFOR and to reach the force size needed. This section discusses the results of the ULE capability as used by the users and on the main OneSAF program's testing in general.

Usability

The usability of the ULE capability was generally well received by both the user event and the OneSAF test team. Unfortunately, there were a number of other problems not associated with ULE during the user event that prevented the operators from fully evaluating usability. Users are very pleased with the ULE abilities for moving large groups of entities. This is true both for normal users as well as senior controllers and scenario generators. ULE is typically used for surrounding wrapper forces, but because it is easier to control large groups, many users turned their units into ULEs for the initial movement phase of an exercise. They would then convert them back to entities prior to expected contact once they arrived at the assembly areas or when an encounter occurred. This has the potential to stress the system with a lot of "on the fly" ULE aggregation and disaggregation. Normal users began requesting abilities to aggregate and disaggregate, currently this requires elevated permissions.

It can be difficult to know the limitations of ULEs. Strict filters on compatible behaviors help the user to assign direct orders to ULEs. In OneSAF, most of the automated behaviors are designed to execute at the platoon and company level with each individual or vehicle modeled separately. These behaviors do not work with an aggregate representation. Aggregate versions of behaviors can be executed, for example to ambush or assault a building, but they execute unexpectedly when ordered on a ULE because the unit icon acts like a single entity in some cases. Since ULEs are integrated in the system and the GUI generally tries to prevent user error, some users are confused why they cannot perform expected tasks at the company or platoon level when their unit is a ULE. This issue is generally resolved after training.

A bigger, and more difficult issue to train are possible ULE capabilities. Since ULEs can represent any echelon and OneSAF orders can also be applied at any echelon, it is easy to get into a situation where a user has a traditional unit but with ULE subordinates (both direct or even an echelon or two down). For example, there may be an infantry platoon with ULE squads. The user might expect to be able to give that unit an Assault Building behavior. The system allows this because it does not filter behaviors available to units with subordinate ULEs. This issue is worsened because the system does not have a way to help the user understand this problem and it is difficult to annotate possibilities in combinatorial cases. We found that it is very important during scenario generation to be very deliberate about where ULEs are used by selected echelon. The senior controllers and scenario generators reduce this potential usability problem by ensuring units are aggregated at the appropriate level based on a user's role.

Computation

The ULE capability increases the effective force size in a large scale OneSAF system. The question therefore is what is the actual effective increase? OneSAF maintains a fairly coarse performance test known as the "Line on Line" test which puts a line of identical tanks against each other over flat terrain. This provides a simple and predictable analysis of how quickly the simulation engine and models can process the most common events. The OneSAF developers created a version of the line on line test using company and platoon ULEs. This gives a general idea of the performance increase obtained from the use of a ULE over the equivalent traditional unit of entities line on line test. This test

showed that running equal numbers of tanks at platoon level ULE (four tanks per ULE) was 5.4 times computationally faster than running medium resource entities.

The performance improvement is noticeable when running large scale exercises. On the OneSAF program, the normal large scale test prior to introducing ULEs was 12k entities using three computers (SimCores) running the entity models. Outside the OneSAF program, some users ran up to 40k entities using significantly more SimCores and/or clustering OneSAF into multiple HLA federates. Since adding the ULE capability, the large scale test run by the OneSAF program has grown to 45k entity equivalents on the same SimCores and has been run up to 75k entities with additional SimCores with a limited success maximum of over 100k entities. In the standard 45k entity count scenario, around 10k entities are medium resolution and the other 35k are represented as ULEs and different echelons (mostly company and platoon). In all of these cases, the entities are a mix of medium resolution entities and ULEs and our standard of measurement includes running in real time with or without ULEs.

As the ULE capability was integrated into OneSAF, it allowed significantly more entities to be used on a regular basis than had been used before which challenged real time operability. This created a lot of stress on the system in areas that were not expected. The SimCores were previously seen as the bottleneck to expanding the entity count. But once the system could handle the large numbers of entities (greater than 40k), the user interfaces became very slow dealing with the entity count. By design, the ULEs were built to take advantage of the OneSAF infrastructure by making the infrastructure see the ULE and the ghost entities as normal entities in most respects. This resulted in larger unit and entity lists, more things to display and track, and more information to store. This also caused a need to increase the memory profile on the user interfaces to handle the entity count. A number of other system-wide optimizations were put in place. Most of these issues were corrected by the time of the user event, but the user event pushed the entity count even higher than had previously been attempted and still exposed similar problems to a lesser extent.

The ULE capability reduced the overall system load with the majority of the OPFOR entity represented as ULE during the user event. One problem encountered was that users frequently changed their minds about which entities needed to be ULE and switched them to medium resolution entities during execution. This caused churn in the system as medium resolution entities consume more resources than ULEs.

Overall, the ULE capability has met its intent. Though there were issues with the ULE capability and OneSAF running higher entity counts in general, it did accomplish the goal of providing wrapper forces for stimulating and testing the mission command systems. It also reduced operator workload when performing some maneuver functions. The usability, though not fully successfully tested given some other non-ULE issues, did show well enough to run the simulation.

LESSONS LEARNED

The objectives of the ULE effort are to reduce computational load and to reduce operator load during event execution. Not surprisingly, once the ULE capability was in the hands of users, it was applied in ways not envisioned. The user community identified an advantage to the ULE in scenario creation. When creating scenarios involving large units (divisions or multiple brigades), the users performed the initial force laydown with all ULE representation. This simplifies unit placement and starts the scenario with minimal computational load. During event execution, the users convert ULEs to traditional compositions (and back again) as the situation warrants. This technique helps govern performance and allows users to increase the force size without increasing the hardware footprint.

We had envisioned users converting from ULE to traditional units rarely during an event. New challenges arose as the ULE represented units' sizes and types were increased. Conversion between traditional and ULE representations frequently, or with large units, exposed unfortunate negative effect on system performance. Creating a ULE requires the system to calculate a ULE composition from scratch, originally every time. This includes iterating across all of the unit's entities to select the representative entity and other such high-cost activities. To enable the seamless transition from ULE to traditional unit, the OneSAF stores both the ULE and the traditional entity-based version of each ULE unit. While the ULE implementation is active the traditional unit is not and vice versa; this prevents recalculating the ULE compositions trading memory for processor load and allowed us to optimize conversions.

The ULE implementation previously suffered poor performance when having to report ghost entity locations. The system calculated the location (offset from centroid converted to global coordinates) every time location was

requested. Since requests could arrive asynchronously, the system would often recalculate a location for the same ghost multiple times per system "tick". The issue was subsequently addressed by calculating location information for each ghost at most once per tick and caching the result to provide for additional requests.

The original ULE concept envisioned implementation primarily at the platoon level and below. When escalating to larger unit representations, we learned that ULEs perform unrealistically in restrictive terrain, especially where there should be limited crossing points (i.e., bridges). Since ULE mobility is calculated for a single representative entity, the entire unit represented by the ULE can cross a single lane bridge in the time of a single entity. In the real world, the subordinate entities would take a non-negligible amount of time failing out and into formation to cross the bridge. A workaround is recommended aggregate ULEs at lower echelons when possible; this is still a single operation and does not require selecting every subordinate unit to convert them individually.

FUTURE WORK

The work on ULE has proven the viability of the concept and provided a useful capability. However, there is work that could be done to improve the usefulness of the ULEs beyond wrapper forces. Indirect fire capabilities are not yet implemented. ULEs currently take appropriate damage from incoming indirect fire but do not provide indirect fire engagements. The primary challenges are the Fire Direction Center logic and the specialized formations as they relate to sheaf patterns. Traditional tube artillery units in particular lend themselves to ULE implementation once the indirect fire logic issues are resolved. These units typically contain large numbers of entities, usually only a limited number are relevant to the scenario. It is foreseen that ULE indirect fire implementation will simplify operations for users that want to conduct fire missions but where fire missions are not the primary focus of the scenario, exercise, or experiment.

It was considered at one point whether the switching between traditional and ULE representation should be automated, the current implementation gives the user full control without any automated surprises. We now believe that there is room for configurable automated conversion between ULE and traditional units triggered primarily by geographic conditions or when performing particular types of behaviors or interventions. ULE automated conversion may be applied by containment within entity level play boxes or terrain feature boundaries such as urban areas. As it is not a desired effect to continually switch between unit representations while maneuvering, a potential solution is to create a buffer zone such that the unit converts to traditional representation at one edge of the zone and converts to ULE at the other edge. Research is needed to determine if the buffer size needs to be scaled based on the represented unit echelon. It is possible to conceive incrementally transitioning a unit from ULE to traditional representation and vice versa considering the partial de-aggregation technique discussed below.

The current ULE implementation allows a user to de-aggregate a ULE by echelon. Thus, a ULE representing a battalion could be converted to a unit of company level ULEs (and other sized ULEs for the various headquarters and supporting elements that may not be organized at the company level). While this remains a very useful capability, we have learned that it would be equally useful to selectively de-aggregate within a unit. For example, the user should be able to only de-aggregate selected companies into traditional units and entities while the remainder of the battalion stays represented as a ULE. This approach allows the user to order the entire battalion unit to move together, while also allowing the user to task the companies separately. The implementation requires consultation with users to determine optimal solutions.

A related future capability is to extract selected entities (i.e., UAS) from a ULE. OneSAF supports units consisting of a mix of ULE and traditional entities. Additionally, we prefer to implement logic that allows the puppet master to control a traditional entity and the human user to control the entire unit (ULE and extracted entity) without having to issue separate commands. In general we recommend the use of true entity level representation for urban operations. However, as the military wrestles with concepts for operating in extremely large, dense urban environments (megacities), there may be value in implementing urban combat behaviors in ULE. The location calculation and visual representation point for a ULE is the centroid of the unit. For large units and extended formations, this is often suboptimal. This logic can be improved to include a data or situationally driven location calculation point to allow you to show the lead of a unit when moving in formation.

CONCLUSION

The ULE approach allows performance savings for modeling aggregate representations of units rather than modeling individual entities. It has been shown that the ULE approach has merit for modeling larger number of entities at varying and appropriate levels of fidelity based on the dynamics and physical characteristics of the battlefield, while reducing the computational resources required. However, implementation of the approach is complex; the how's and when's to transition between high fidelity and aggregation is a function of scenario situation and requires more exploration. The implementation of the ULE approach is an art at this time, but through additional exploration can mature to more of a science and become a valuable tool for the modeling, simulation and training community. While there are functionality costs to providing such a capability, the operators at any point can switch between traditional units and ULEs in order to avoid capability lock-in to a lower fidelity model. To that extent, models were carefully considered to balance performance, functionality, and user expectations when developing aggregate representations, compositions, supplies, interoperability, sensing, mobility, engagements, vulnerability, behaviors and interventions, and tactical device stimulation capabilities (Lacks, 2014). The initial ULE use case was envisioned to support wrap-around forces, but over time as capabilities improve and expand to include models and behaviors, the ULE capability may be able to augment entity simulation with a low fidelity aggregate simulation capability.

ACKNOWLEDGEMENTS

The authors acknowledge Rocky Hanlin for providing excellent representative feedback of the ULE users. We also acknowledge Tom Riley, Senior Software Engineer on the OneSAF I2S program, for his continuous pursuit of ULE perfection over its full lifetime to date, tireless work ethic, and consultation writing this paper.

REFERENCES

Department of Defense (1999). *Department of Defense Interface Standard: Common Warfighting Symbology*. Retrieved May 24, 2016, from http://www.cs.cmu.edu/~softagents/papers/MILSTD_2525B-Common_WFX_Symbology-30Jan99.pdf.gz

Joyner, David (2007). An Introduction to Systems of DEs. *United States Naval Academy*. Retrieved June 20, 2016, from http://www.usna.edu/Users/math/wdj/_files/documents/teach/sm212/DiffyQ/de-lanchesters-eqns.pdf.

Lacks, Daniel J, PhD (2014). OneSAF Unit Level Entities (ULE). *Cole Engineering Services, Inc. presentation to TEMO*. Available upon request.

PEO-STRI (2016). *One Semi-automated Forces (OneSAF)*. Retrieved June 17, 2016, from http://asc.army.mil/web/portfolio-item/peo-stri-one-semi-automated-forces-pdm-onesaf/.

Strickland, Jeffrey (2011). *Using Math to Defeat the Enemy: Combat Modeling for Simulation*. Raleigh, NC: lulu.com.

US Army (2008). *History of Military gaming.* Retrieved June 20, 2016, from https://www.army.mil/article/11936/history-of-military-gaming/.

Wikipedia (2016). List of Avalon Hill games. Retrieved June 20, 2016, from https://en.wikipedia.org/wiki/List_of_Avalon_Hill_games.