Embedding Simulators in a Mission Command System for Couse of Action Analysis

Daniel J. Lacks, PhD

Cole Engineering Services, Inc.

Orlando, Florida

Daniel.Lacks@cesicorp.com

ABSTRACT

This work examines the practicality and effectiveness for embedding simulators in a mission command device. The goal is to use only the operational plan in theater for simulation input, hiding all simulator details from the operator so he/she does not need to learn new tools. A prototype capability is discussed which produces a Course of Action (COA) analysis based on an operational plan produced in SitaWare and simulated by embedded headless MTWS and OneSAF simulators. After inputting the operational plan, the commander selects the number of simulation runs to execute and presses a button to start the simulation which runs faster than real time in the background. When the simulation runs complete, the commander may view the results in graphics and charts which compare the multiple runs. The future capability is projected to allow commanders to simulate any echelon and order for training and wargaming use cases.

ABOUT THE AUTHOR

Dr. Daniel Lacks is the Chief Scientist at Cole Engineering Services, Inc. (CESI). 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. Since 2001, Dr. Lacks worked as a software and systems engineer in the DoD M&S industry on Live, Virtual, and Constructive programs including Warfighter's Simulation (WARSIM) and the One Semi-Automated Forces (OneSAF) Integration and Interoperability Support (I2S) program. Dr. Lacks lead the CESI Internal Research and Development (IRAD) initiative to integrate the Marine Air Ground Task Force (MAGTF) Tactical Warfare Simulator (MTWS) and OneSAF in the cloud, demoed at ITEC in 2015.

Background

Problem Statement

Upon examining the multitude of creative and effective modeling and simulation (M&S) solutions across the community, a rule of thumb emerges. When systems become more sophisticated, they inherently become more complex. This phenomenon is not unique to M&S. As (Arthur, 1993) points out, a powerful jet engine built in the 1990s was 30 to 50 times more powerful than a jet engine built in 1930s, but it took a simplistic system designed before the age of computers and "encrusted" it into an array of systems and subsystems with approximately 22,000 parts. The M&S systems of 2018 are far more feature rich than systems built in 1958, but the history is slightly different. Back then, simulation results took too long to get. They produced ambiguous results using analog computers, needed too many skilled people, and were not cost effective (Sagar, 2000). Fortunately, simulations today leverage digital computers and may be cost effective, but they are still sophisticated and complex. Complexities may include, similarly, needing too many skilled people to operate a simulator, developing user interfaces which are hard to use or require training to operate, and employing cybersecurity countermeasures on computers and networks.

While our community is *pulling* towards open systems architectures, simulation as a service, cloud hosting, and other advanced concepts, there is an equivalent *tugging* from the user communities to make systems easier to operate, deploy, maintain, and secure. The industry is responding to this, for example, in the virtual and live communities with solutions like augmented and virtual reality to immerse trainees into familiar environments that do not require them to learn the mechanics of simulator software. At ITEC 2017, there was a drive to having "casual gaming, anytime anywhere" (Muller, 2017), "decreasing the lead time" (Schmidt, 2017), and "creating a low cost simplified constructive simulation solution" (Jinnestrand, 2017). These statements imply some user communities are looking for simpler solutions rather than more sophisticated ones.

The constructive simulation domain training audiences today are already immersed in an effective environment which includes their tactical MC systems, communications devices (phones and radios), and other common planning tools such as whiteboards, pen, and paper which replicate their Tactical Operations Centers (TOCs). Those MC systems may be networked to a complicated federation of simulation systems, hidden in another room or perhaps another continent. The simulators are driven by any number of trained personnel to include operators, software developers, database administrators, Information Technology (IT) staff, systems engineers, and others. Quite often, this staff is augmented by automated and semi-automated intelligence to drive large scenarios, which also complicates the effectiveness of conducting a simulation exercise.

There are many reasons to embed simulation into mission command systems rather than merely interoperate, as is common today with solutions like the Mission Command Adapter-Web Service (MCA-WS). In this work, we focused on the course of action (COA) development and analysis use case while hiding the simulation from the user. Once the simulation is embedded in the mission command system, additional use cases become possible (Surdu, 2002). The embedded simulation can be used for iterative refinement of a selected COA into a plan. This includes enabling multiple echelons to collaboratively develop the plan in parallel, rather than the current, largely serial manner. Upon selection and refinement of a COA into a plan, the plan could then be played back through the mission command systems to facilitate mission rehearsal. This same mission rehearsal capability would also facilitate embedded operator and collective training with significantly less reliance on simulation centers and simulation operators. Once the operation begins, the plan can run in parallel (and maybe a little ahead) of the current operation to identify when the plan is going awry and alert the commander and his staff to conduct proactive re-planning and/or development of branches and sequels.

The solution presented in this work focuses on using a constructive simulation to facilitate COA analysis while hiding all the simulation complexity from the operator. The mission command system operator does not need to know how the simulation works – or even what constructive simulation is helping to analyze the COA. The MC operator simply enters one or more COAs into his or her MC tool (which he or she ostensibly or actually does anyway), presses a “simulate” button, and is presented with empirical results of simulating those COAs against one or more enemy COAs to help guide decision making. This low overhead approach, by design, would produce less granular results than the sophisticated simulation exercise. During COA analysis the simulation needs to be accurate enough to help determine which COA best meets the commander’s intent and should be refined. It does not need to provided highly detailed results or enable the operator to have exquisite control of the actions of every entity and unit within the simulation. This careful balance is useful to the professional, experienced commander and staff operating the MC system in Disconnected, Intermittent, and Low Bandwidth (DIL) environments at the point of need. We examine such an approach using the SitaWare Headquarters (HQ) MC system, coupled with the Marine Air Ground Task Force (MAGTF) Tactical Warfighting Simulation (MTWS) and One Semi-Automated Forces (OneSAF) to examine embedding simulators into the MC stack.

SitaWare HQ

SitaWare provides an open architecture Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance (C4ISR) platform for more than 15 nations (C4ISR Software, 2017). The platform facilitates making complex decisions on joint and coalition battle networks and is accessed from a web browser. The open architecture allows developers to add features to SitaWare HQ that are not native to the baseline. These features may include new web interfaces, backend communications, or accessing the database – each are leveraged for this study. One of SitaWare's strengths is its operational simplicity which is why we reused the system rather than building another GUI and suite of tools to embed in the MC server stack. Also, since SitaWare HQ is used in the Joint arena, it provides more representative capability (air, naval, land, joint, coalition, government agencies, and NGOs) to describe scenarios than MTWS or OneSAF alone. This may be leveraged by future simulators or by expanding OneSAF or MTWS capabilities.

Reusing SitaWare provides additional distinct time and cost savings benefits to creating a low overhead embedded training and wargaming solution. SitaWare provides the mechanics and NATO standard graphics to construct scenarios and overlays in, what is referred to as, the SitaWare Plan. The plan provides real-time collaborative military planning capabilities which synchronizes changes across users. Those changes are saved into a database which is accessible through the Application Programming Interface (API) or by accessing the database directly. SitaWare provides the ability to define task organization and force management, supplies, platforms, and basic loads. This information may be imported into the simulator from SitaWare and/or exported from the simulator to SitaWare. SitaWare also provides reporting capabilities, we reuse some of those constructs to create our own COA reports. There are other functional and non-functional features, such as security, communications, diary, performance, and more, which come with SitaWare. While they are not explicitly used by our experiment, they are present for future capability without us having to implement these features if we had built a new tool.

OneSAF

OneSAF is a toolkit which facilitates building a Live, Virtual, or Constructive (LVC) simulation, typically for brigade and below entity-level staff training, experimentation, acquisition, research, testing, and other military and domestic activities (PM ITE, 2017). OneSAF can operate as a constructive simulator out-of-the-box, though it is designed and intended to be customized to meet each users need through software and/or data development. OneSAF also supports limited aggregate and Unit Level Entity (ULE) representations and behaviors. In the military domain, OneSAF supports ground combat, intelligence, air operations, fire support, maritime operations, combat engineering, combat service support, CBRN combat models, cyber, and more. It interfaces with simulators using High Level Architecture (HLA), Distributed Interactive Simulation (DIS), and other protocols. OneSAF adheres to community standards such as Military Scenario Definition Language (MSDL) and Coalition-Battle Management Language (C-BML). OneSAF features digitized terrain, weather, sea state, and runs a simulation engine which keeps track of the time of day in real-time or faster than real-time. It has various assistive capabilities such as After Action Review (AAR), scenario generation, and cloud optimizations. The MCA-WS stimulates a variety of tactical systems, however it is not needed for this experiment since we built a new interface to automate driving the simulator.

MTWS

MTWS is a constructive simulation for aggregate-level battalion to multi-Corps level Joint Task Force (JTF) staff training, however it supports entity-level internal representations. In the military domain, MTWS supports ground combat, intelligence, air operations, fire support, maritime & amphibious operations, combat engineering, combat service support, and CBRN combat models. It supports simulation interoperability using HLA and DIS. MTWS also features digitized terrain, weather, sea state, and runs a simulation engine which keeps track of the time of day in real-time or faster than real-time. It has various assistive capabilities such as After Action Review (AAR) and scenario generation. MTWS stimulates a variety of MC systems, however this capability is also not needed for this experiment since we built a new interface to automate driving the simulator.

Design

In order to create this embedded simulation capability, we established ground rules and expectations. The primary rule is that the operator is only allowed to interact with SitaWare. The simulation experience needs be completely headless and hidden from the operator. This rule decomposes into many automation tasks to include populating default values for various order inputs, deconstructing fragmentary orders (FRAGOs), generating Air Tasking Orders (ATOs), executing behaviors, governing supply levels, factoring logistics (known as holding types in SitaWare), target/munition selection, and all of the complex logic that normally requires rooms full of operators and Subject Matter Experts (SMEs) to employ. It also defines an expectation that simulation results will return quickly, not in real-time, but faster than real-time so that the commander does not have to wait three days to execute six 12-hour COA variations.

Solving both of these challenges provides a backdrop to the complexity involved with embedding simulation in the MC stack. The challenges introduce the fuzz in the logic that produces less granular results than the sophisticated simulation exercise, but also greatly simplify the commander's intent to view forecasts between several sips of coffee. This approach is, from our experience, unique. While we are familiar with the differences in results between using an aggregate- or entity-level simulation, those results are driven by humans making explicit decisions to drive the simulation from the commander's guidance or intent. Removing the human from the loop drove us to produce several COAs with explicit constraints so that the commander may review these COAs, perhaps challenge some of them, and then explore them with additional iterative runs. This is within the scope of the experimentation performed and the level of effort we wished to expend on the commander to review the results. After all, we are producing forecasts, not crystal balls.

Creating constraints assists us with automating decisions and populating values that normally would be populated by humans in the loop operating the simulators. Multiple constraints are created across different runs to allow the commander to see different points of view generated in COAs for a particular plan. These constraint factors include casualties, ammo, fuel, equipment, and overall. The most likely and most dangerous outcomes are flagged given the constraints specified compared across each of the COAs.

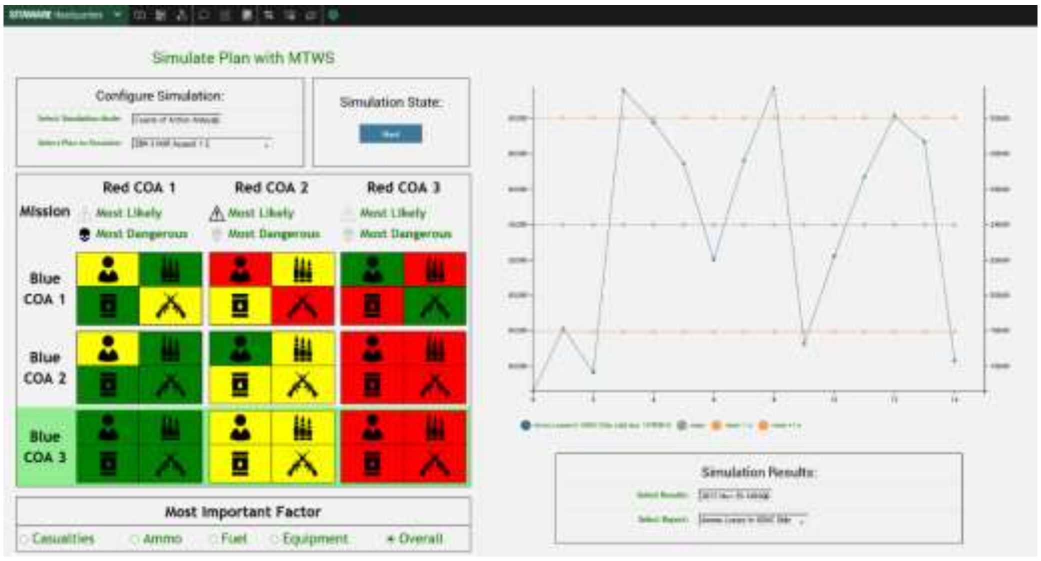

Figure 1 - Running Simulations and Displaying Results

The basic execution design is such:

- The operator creates multiple plans in SitaWare, three BLUE COAs and three RED COAs.
- The operator describes how many times to iterate over each plan, then presses the "simulate" button.
- The simulator runs headless in the background. The user may view results as they are obtained.
- The operator filters on various constraints when the runs are complete. For example, the operator clicks an option to show which COA uses the least amount of fuel. That displays the results of the other constraints, perhaps the result using the least amount of fuel had the highest attrition.
- When the simulation is complete, there are nine different missions to compare against when three COAs are specified for RED and BLUE. As shown in Figure 1, good outcomes for each factor are shown in green, medium in yellow, and poor in red.

Our approach to embedding OneSAF and MTWS in the MC server stack first started with OneSAF, then moved onto MTWS. In a previous ITEC presentation (Lacks, 2015) and on the show floor, we showed that MTWS and OneSAF can interoperate in a cloud. However, this experiment to embed MTWS and OneSAF into the MC stack does not go as far to interoperate the three SitaWare, OneSAF, and MTWS systems simultaneously simply due to time and budget constraints. While there will be technical challenges to interoperating all three systems simultaneously, the technical aspect was not a deterrent to consideration. Perceived challenges are typical for integration such as matching up holding types to simulator enumerations. Our approach is designed to not only accommodate a future combination of MTWS and OneSAF with SitaWare, but other simulators of any make or model. This is a concept for future exploration and represents the point where the scope of our experimentation ends.

The future compatibility with MTWS, OneSAF, or any simulator is achieved by the design of our open source SitaWare interfacing REpresentational State Transfer (REST) web service called Sim Controller, implemented in Node.js. Sim Controller interfaces with the SitaWare Plugin to start, stop, and status the simulators running as well as orchestrate the various simulator runs that are specified by the operator. We then developed capabilities in the simulators to assist with translating SitaWare events and data to the native simulators, in our case MTWS and OneSAF. These capabilities facilitate the negotiation of holding types in SitaWare with platform, equipment, and supply enumerations in the simulators. They exchange plan information with the simulator and assists with constructing a scenario for the simulator to run. As the scenario executions complete, the capabilities assist the simulator saving results and by placing them in the SitaWare database so they may later be used to generate reports and graphs.

Results

The overall results of the experimentation are successful, but there will definitely be challenges in hardening the capability to cover a broad spectrum of missions. The simulator capability implementation varied between OneSAF and MTWS. Since OneSAF is an open source system designed for extension and composability, we created a SitaWare Interface OneSAF system composition (in Java, the primary native OneSAF language). We treated MTWS as a closed system and developed the MC Embedded capability (also in Java) which leverages the Automated System Control (ASC) capability. ASC provides the ability to run basic MTWS commands and record results overnight and headless. The MC Embedded tool generates an MTWS batch file, or scenario, starts MTWS in ASC mode, points MTWS to load the batch file for execution, and then shuts MTWS down.

OneSAF generally required fewer inferences to be made when automating populating orders by the web service since there is already automation built into OneSAF, despite that OneSAF is an entity-level simulator with a higher fidelity of detail. Thus, MTWS generally required more inferences to be made when automating the construction of missions and orders. One of the most complex orders to create automatically is the ATO. Our demonstration scenario accommodated defaulting values which made sense for the particular scenario, but generalizing these inputs for any future scenario will definitely pose a challenge and requires more investigation on how to infer a commander's intent based on standard symbology placed in the SitaWare Plan.

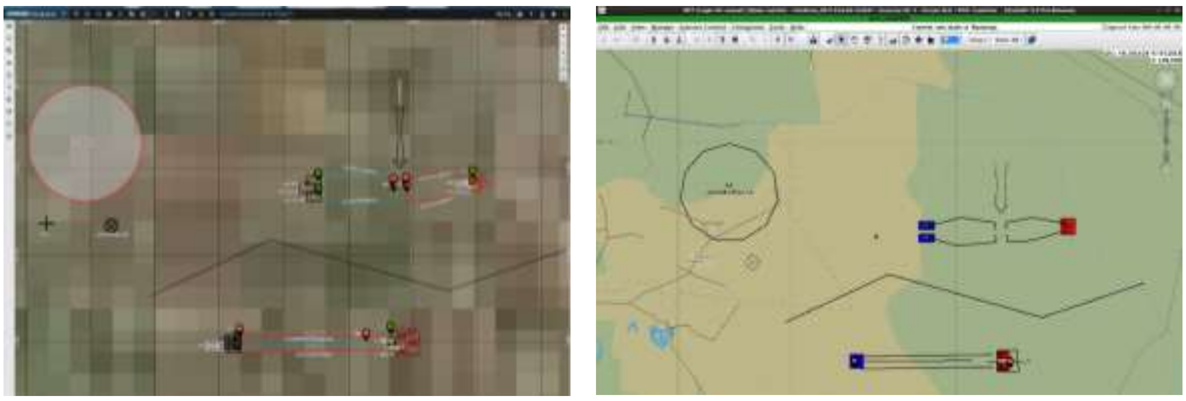

Figure 2 – While OneSAF is hidden from view, we can see the automated scenario force laydown generated by the web service is identical between the SitaWare Plan (left) and OneSAF (right).

The results generated by the simulators are, in our estimation, equivalent regardless is a human or a tool generated the inputs – at least for the prototype scenarios examined. OneSAF and MTWS provide enough built in randomness to provide different, but believable, results when running the same scenario multiple times. The biggest challenge to obtaining results was to obtain them fast enough. This led us to investigate and possibly improve performance as a positive outcome for conducting these experiments.

For MTWS, the original measured maximum speedup was between approximately 6:1 and 10:1. Improvements were made to operate MTWS up to 60:1. Thus, if a 2-hour mission starts at 1200 exercise time and 1200 real time, and the clock is running at 60:1, then 1400 exercise time arrives at 1202 real time. Events are not skipped, the clock just ticks faster.

But, the simulation event density impacts performance. If there are a lot of events to process, the simulator may not keep up with the pace of 60:1 due to hardware limitations, but no events are lost nor is the fidelity compromised.

For OneSAF, our team identified and developed significant performance improvements necessary in the simulation engine. We discovered that models were sending out duplicative changes, roughly 35-40% within the OneSAF distribution. Additionally, we corrected inefficient thread synchronization and over reliance on the use of introspection in performance-critical areas of the simulation. Once these issues were corrected, in a smaller real-world entity-level scenario with roughly 200 entities, a 250:1-480:1 speedup was observed (hardware dependent), which could easily translate to 1000:1 speedup using ULEs. These changes additionally reduced OneSAF's time to join an HLA federation by 350%, vastly improving OneSAF's scale in a distribution. In a larger real-world scenario with approximately 4000 entities, a speedup of 6:1 was observed when asked to run as fast as possible - note that the hardware was not identical nor as optimal to the first experiment by design since we were looking to push the limits of the performance improvement to find its range to quantify expectations. While entity count may be a factor in determining the amount of speedup achievable, our observations pointed us to believe that the density of the simulation events were more of an indicating factor. Thus, the simulation runs faster when it has less to do, but also runs faster than previously when it processes events due to the improvements implemented. The improvements are currently being tested by the OneSAF program and are expected to be released to the community in OneSAF v8.8 in 2018.

Future Work

The results of this experiment point to a very useful capability to commanders and staffs that could be extended and made available to operators in a short period of additional development. Additional enhancements planned for future development include more work to enable the simulation to better understand the plan and properly populate the simulation, the ability to automate assemble you COA combinations for simulation, better analysis and decision support tools, improved simulation performance, the ability for SitaWare to select the best simulation based on the plan, create some static constrain checking before simulation, and implement the other use cases.

Conclusion

This work examines the practicality and effectiveness for embedding simulators within a mission command system in a way that is seamless and transparent to the operator. The goal is achieved in the prototype to use only the planning tools native to the mission command system for simulation input, hiding all simulator details from the operator so he/she does not need to learn new tools. The prototype capability facilitates a COA analysis and selection. Operators produced COAs for both Red and Blue in SitaWare and simulated the combinations of those plans using MTWS and OneSAF. The future capability is projected to allow commanders to simulate any echelon and order for training and wargaming use cases, but more work is needed to automate populating a variety of orders which were not covered within the scope of the prototype and to continue to improve performance.

References

Bryan, W. Arthur. (1993, March). Why do things become more complex? *Scientific American*, Page 92. Retrieved from URL http://www.uvm.edu/pdodds/files/papers/others/1993/arthur1993b.pdf.

C4ISR Software. (2017, December 14). Retrieved from https://www.systematicinc.com/.

Jinnestrand, Ulf (2017). *A Concept Development of a Lightweight Combined Operations Training Capability.* Presented at ITEC 2017, Rotterdam, The Netherlands.

Lacks, Dr. Daniel J. (2015). *MTWS in the Cloud.* Presented at ITEC 2015, Prague, Czech Republic.

Muller, Tijmen (2017). *Mobile Serious Gaming for Military – Two Cases*. Presented at ITEC 2017, Rotterdam, The Netherlands.

PM ITE (2017). One Semi-Automated Forces (OneSAF). Retrieved from URL http://www.peostri.army.mil/onesaf.

Schmidt, LTC (DEU A) Woflhard (2017). *Preparing nations for training in MN/NATO events – Best Practices.* Presented at ITEC 2017, Rotterdam, The Netherlands.

Shinde, Sagar. (2000, April 7). Introduction to Modeling and Simulation: Historical Perspective. Retrieved from http://www.uh.edu/~lcr3600/simulation/historical.html.

Surdu, John, John Hill. (2002, April). *Simulation in Command Posts of the Future.* Advanced Simulation Technology Conference, San Diego, CA, 14-18 April 2002, pp. 77-86.