Proceedings of the 2006 Winter Simulation Conference
L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds.

A METHODOLOGY FOR CONDUCTING COMPOSITE BEHAVIOR MODEL VERIFICATION IN A COMBAT SIMULATION

Eric S. Tollefson
Jeffrey B. Schamburg

United States Army TRADOC Analysis Center
Building 245 (Watkins Hall Annex)
Monterey, CA 93943-0695, U.S.A.

Harold M. Yamauchi

Rolands and Associates Corporation
United States Army TRADOC Analysis Center
Building 245 (Watkins Hall Annex)
Monterey, CA 93943-0695, U.S.A.

ABSTRACT

The United States Army's One Semi-Automated Forces (OneSAF) Objective System (OOS) is the next generation of Army high resolution combat models. Its development has leveraged the ever-increasing computing power available today to represent highly complex battlefield phenomena, particularly human behavior. In the fall of 2005, the Product Manager (PM) OneSAF asked us to conduct a verification of the orderable, composite behavior models within OOS. As a result, we developed and executed a unique process to verify those behaviors under tight resource constraints. Our methodology and test designs allowed us to evaluate the behaviors thoroughly with a minimum number of scenarios. Based upon our work, we were able to verify a number of composite behaviors and to provide valuable feedback to the PM OneSAF. In this paper, we provide an overview of the problem, a description of the methodology we developed, and a summary of our challenges and results.

1 INTRODUCTION

The OneSAF Objective System (OOS) is the first combat simulation, or set of simulation products, to be developed through the formalized Army acquisition process. OOS is "a next-generation Computer Generated Force (CGF) that can represent a full range of operations, systems, and control processes from individual combatant level and platform level to fully automated… [friendly force] battalion level and fully automated… [enemy force] brigade level. [It] is not a single product or system, but rather, a set of products each consisting of a set of interacting components and tools (Randolph and Sagan 2003)." One of the many unique aspects of OOS is its behavior models, the focus of this effort.

The main development phase is drawing to a close as the program prepares for product release. Prior to its release, the program must successfully pass the government acceptance testing (GAT), originally scheduled for January, 2006, but subsequently postponed until May and June, 2006. In advance of the GAT, PM OneSAF asked the Training and Doctrine Command (TRADOC) Analysis Center in Monterey, CA (TRAC-MTRY), in October, 2005, to develop and execute quantitative and qualitative test designs that verify that the orderable composite behavior models in OOS perform to their design specifications.

The primary purpose of this paper is to present the methodology we developed to accomplish this complex behavior verification task with limited time and manpower. Our work demonstrates that, with an efficient, but limited, test design, we can achieve valuable results that provide tremendous feedback to simulation developers.

In this paper, we begin with a description of the problem background that presents a general overview of the OOS model with focus on its behavior modeling functionality, additional detail concerning our problem scope, and a summary of related efforts. The main portion of the paper will lay out the methodology we developed to conduct our verification to include examples. We will then briefly describe our general results and the challenges we faced. Finally, we describe the direction of our continued work and conclude with a summary of our key findings.

2 BACKGROUND

2.1 OOS Behavior Modeling Functionality

We must first explain what is meant by a behavior model. OneSAF Objective System behavior models implement typical decision processes used within a military framework, and thus "provide command and control of equipment and unit models during simulation execution" (Henderson and Granger 2002). Therefore, they provide a means to automate standard decision processes in order to reduce or remove user input during simulation execution. The models are able to evaluate environmental and situational stimuli and cause the entities or units to react ac-

cording to Army doctrine. Thus, these models do not necessarily involve higher-order cognitive processes, as in some human behavior representation models; they normally consist of nested conditional statements to trigger tasks based upon external stimuli.

There are generally two main types of behaviors in OOS – primitive and composite. The "primitive behaviors provide simple chunks of doctrinal functionality from which more complex behavior models are built (Ibid)." The primitive behaviors include coded behavioral aspects that directly control the simulation's physical models (such as movement, weapons, sensors, etc.). The "composite behaviors represent complex behavior models and are composed of primitive behaviors and other composite behaviors (Ibid)." Composite behaviors are not code themselves, but "are defined in data files that conform to a [predefined] syntax (Ibid)." It is the composite behavior models that are the focus of this research.

A graphical user interface allows the user to develop composite behaviors from primitive and other composite behaviors. It is called the Behavior Composer, a tool "that enables users to construct composite behaviors by selecting composition elements from a toolbar, and then placing them on a drawing canvas. The Behavior Composer does not require the user to write source code or even understand the XML file format of the behavior descriptions it produces (Ibid)." While our research did not require actual behavior construction, we were able to refer to the Behavior Composer to learn more about the behavior.

The final aspect of the OOS behavior model that must be discussed is how the behavior models are used in the simulation. First, we must differentiate between orderable and reactive behaviors. Orderable behaviors are those behaviors that can be assigned to a unit or entity by the user during scenario development. A reactive behavior cannot be assigned, but can be enabled or disabled within an orderable behavior that has already been assigned. Reactive behaviors define a standard reaction to particular stimuli, for example, reacting to enemy fire. Because the occurrence of these situations cannot be predicted, they cannot be assigned within the normal mission, or execution, sequence, as orderable behaviors can. Using these two types of behaviors allows the simulation to have a set of behaviors to deal with unpredictable events and a set of behaviors that can fully define the mission, or plan, from start to finish.

2.2 Problem Statement

Although the behavior model functionality is designed to allow the user to create his own behaviors as necessary, the OOS development team created a set of 51 orderable composite behaviors representative of the most likely tasks that a unit or entity might be required to perform within a normal mission. As already discussed above, our task was to evaluate and report on the performance of these composite behavior models. Initially, our guidance was to evaluate as many composite behaviors as possible in advance of the original GAT in January, 2006. With the postponement of the GAT, we were given an extension to continue work until June, 2006. Even with the extension, our resource availability and the timeline severely constrained the scope of our work. Thus, we had to prioritize and develop a means to thoroughly test as many behaviors as possible with limited time and resources.

As implied by the wording of our task, PM OneSAF was asking us to conduct a *verification* of the composite behavior models. The Department of the Army defines verification as "the process of determining that an M&S [model and simulation] accurately represents the developer's conceptual description and specifications" (HQDA 1999). Our work was not intended to include *validation*. Validation is defined by the Army as "the process of determining the extent to which an M&S is an accurate representation of the real world from the perspective of the intended use of the M&S." As we will discuss later, making that distinction proved to be challenging when information about the behavior's 'conceptual description and specifications' was insufficient.

2.3 Background Research

While previous combat simulations have had some behavior modeling capability, we could find no established verification processes or examples specific to behavior models. Additionally, behavior model verification had not received the attention during OOS development that physical model verification had. In fact, only one other organization was working on a similar task. Our sister organization TRAC in White Sands Missile Range, NM, (TRAC-WSMR) initiated a primitive behavior model verification effort in late summer, early fall 2005, nearly the same time we had. Thus, our first step was to develop our own methodology to conduct the verification. Although there was little we could find concerning behavior model verification, there was no shortage of literature and previous research that addressed verification in general. We will only discuss those sources that were most useful to our work.

First, an OOS team traveled to Monterey in October, 2005, to install the software and train the operators on it. That team brought with them a recommended approach for the verification effort. The input was quite valuable in determining the types of information that would be most useful to the development effort and served as a skeleton upon which we constructed our unique methodology.

Our second source of information was the "VV&A Recommended Practices Guide" downloaded from the Defense Modeling and Simulation Office website (DMSO 2002). This document described the verification and validation processes, and best practices from industry, the De-

partment of Defense, and literature, particularly as they applied to combat models and simulation. From this document, we surveyed the large number of techniques available and extracted those that might be applicable to our work.

Our third reference was the "Models Development Behavior Verification Test Plan" prepared for the OneSAF program by Science Applications International Corporation (SAIC 2004). Unfortunately, while that document did give a general framework for the conduct of the verification, it provided little information concerning the methodology for selecting the test scenarios, nor exactly what the outputs should be for each of the scenarios. In fact, when we tried to run these test scenarios and collect the data, we were not even able to load the scenarios. Additionally, the list of behaviors did not correspond to the list given to us by the OOS team, because of the software changes since the document was written. Therefore, while we did use the document to provide some information, we based very little of our methodology on it.

Our fourth primary reference was the work being done by TRAC-WSMR to verify the primitive behaviors. As discussed above, the focus of their effort was on verifying the performance of the primitives, whereas our effort focused on the composite behaviors. They selected composite behaviors to execute based upon the primitives they contained, not the composite behaviors themselves. Thus, while we referred to their methodology to make sure we covered any overlapping aspects, and their results to see if there were any significant differences, we were not able to base any part of our methodology on theirs.

Finally, we consulted with the U.S. Army Materiel Systems Analysis Activity (AMSAA), which is conducting the verification of the physical models within OOS. While the focus of their effort was on an entirely different aspect of the simulation, their approach in selecting design points to test was useful to our effort.

3 METHODOLOGY

After reviewing our sources and discussing the issues as a team, we developed a methodology that would allow us to evaluate a composite behavior thoroughly, while still allowing us to address as many behaviors as possible given our resource constraints. After the initial development of our methodology, we continued to refine the process as we verified behaviors. Nonetheless, the primary steps of the methodology have remained largely unchanged. The following discussion describes our methodology by step. Figure 1 is a graphical depiction of the methodology.

3.1 Prioritize the Behaviors

After development of the methodology, our first step was to prioritize the list of composite behaviors to be evaluated and to update the list as required. A prioritized list of 51 composite behaviors was provided by the OOS training team and served as our base document. The prioritized list has undergone revisions since that initial document. Additionally, we have had to skip over behaviors whose documentation was too insufficient to proceed. Currently, the PM has directed that OOS development team reevaluate the list and reprioritize as necessary. Other changes could be made if software or algorithm failure makes verification

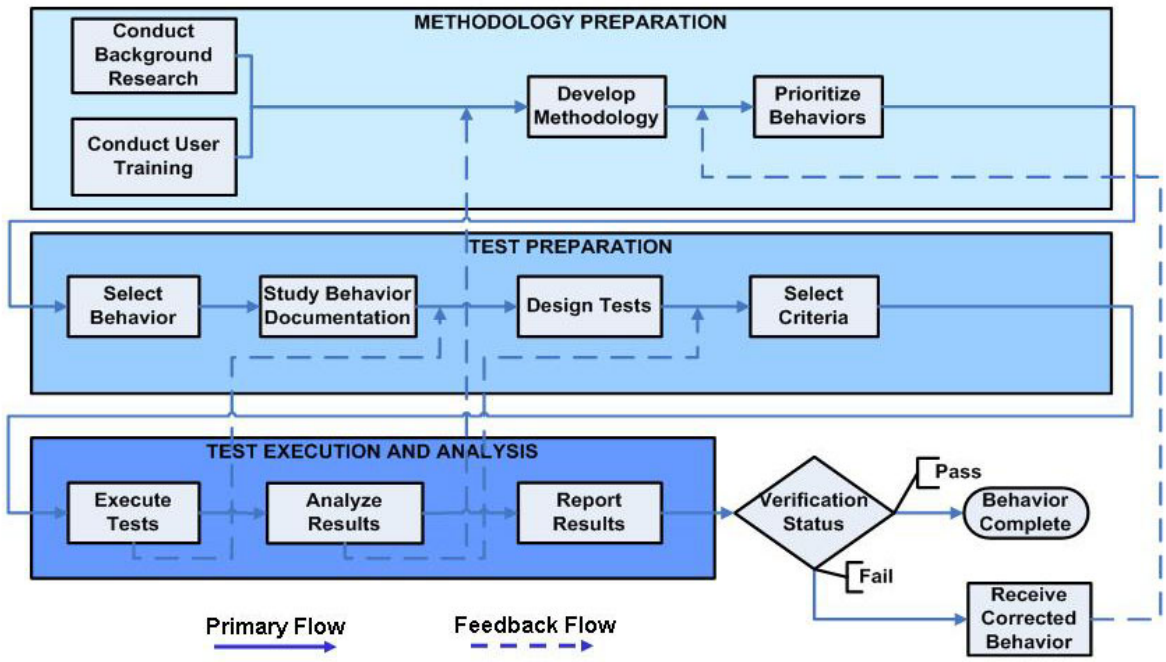

Figure 1: Composite Behavior Verification Methodology

of a specific composite behavior impossible, or if the interrelated nature of two composite behaviors suggests verifying the behaviors concurrently or consecutively.

3.2 Select Behavior

Our next step was to select a behavior from the prioritized list. Given the goal of verification, the challenge was ensuring that we had a complete conceptual description of the behavior models. For that, we looked initially to the behavior documentation.

3.3 Study Behavior Documentation and Other Sources of Information

Our primary source of information was the behavior model documentation. The OOS developers created these documents as part of their knowledge acquisition / knowledge engineering (KAKE) process. Behavior model KAKE documents attempt to capture behaviors in terms of the problem space (the description of the real world) in a way that facilitates the conversion of reality into a software model (solution space). While it is beyond the scope of this paper to describe the OOS KAKE process, we will briefly describe the key documents that were central to our research. The reader can find more information about this process in Randolph and Sagan (2003).

The primary problem-space documents are the Task Descriptions (TDs). These documents describe the Army Universal Task List (AUTL) tasks that may be represented as composite behaviors. The AUTL is a comprehensive list of tasks that the Army is required to perform in support of its mission. There is a one-to-one mapping of TDs to AUTL tasks, but not from TDs to composite behaviors. In other words, one cannot necessarily trace an implemented composite behavior in OOS directly to one TD. The TD is a problem-space document, meaning that it attempts to describe actual behaviors in a detailed manner that can then be implemented in software. Therefore, it cannot serve as a primary reference document for verification because it does not necessarily match how the behavior(s) it supports is/are actually implemented. We did refer to the TDs occasionally to see if they could shed light on gaps or misunderstandings encountered in the solution-space documentation, particularly in terms of nomenclature.

Process Step Descriptions (PSDs) further decompose and describe sub-tasks that are part of the overall AUTL tasks. A PSD may describe a sub-task of multiple AUTL tasks. There is no one-to-one mapping of PSDs to primitive behaviors. These are again problem-space documents and we only referred to them as we did with TDs.

Behavior Process Documents (BPDs) describe behaviors that needed to be represented as composite behaviors but had no associated AUTL tasks. Thus, they are used to fill in the modeling gaps left by the AUTL. As these are again problem-space documents, we only referred to them if necessary.

Use Cases describe the actual implementation of the composite behavior; therefore, there is one Use Case per composite behavior. Although titled Use Case on the actual documents, the OneSAF team also referred to these documents as Design Documents, which is more descriptive of their function. These are solution-space documents that describe how the behavior has been designed to execute, and can be considered the 'developer's conceptual description.' They have as their sources the TDs, but may or may not reflect the same logic as that in the TDs. These are the primary documents that we used in the verification process.

If the documentation failed to present a conceptual model complete enough to conduct verification, we consulted members of the OOS development team. We were typically able to consult directly with the software engineer who implemented the specific behavior we were verifying. Usually the engineer was able to answer our questions. We preferred to do this via email in order to maintain a written log of the questions we asked and the answers we received for future reference.

Our last source of information was our own team's expertise in Army operations and in using combat simulations; however, we had to be very careful not to make assumptions about how the behavior *should* perform – a validation issue. Nonetheless, we encountered quite a few instances in which this was our only recourse in order to proceed.

3.4 Design the Tests

In this step, we used our conceptual understanding of the behavior to create a test design, consisting of a set of scenarios, to evaluate the critical aspects of the behavior. Each scenario can be thought of as a single design point in the overall test. The specific methodology for choosing the number of and settings for the scenarios varied by behavior. We required different methods because of the nature of the tested behaviors. For instance, the Move Tactically behavior had 16 required and optional inputs. Those inputs aligned well with the critical aspects of the behavior that we wished to test. Tailgate Resupply, on the other hand, had only 4 required and optional inputs, but there were other aspects of the behavior that we wished to test that did not correspond to inputs. Thus, we had to take each behavior as a unique case and determine the tests uniquely, instead of using a 'cookie cutter' approach. We will use the Tailgate Resupply behavior as our example throughout this discussion. In this behavior, the unit given this task, called the supplying unit, will move to the logistics release point (LRP – location where the resupply operation will take place), supply each of the designated vehicles there, and then move to a return location (which is not necessarily

their original location). We will now discuss in detail our process for designing the test set.

3.4.1 Conditions

The basis of our test design was the set of conditions we wished to evaluate, drawn from the behavior's conceptual model. We categorized these conditions as either inputs or special cases. The following describes these in more detail.

3.4.1.1 Inputs

When a user assigns a composite behavior to a unit or entity, a dialogue window opens prompting the user to choose inputs for the required and optional parameters. Logically, each input the user enters should have an effect on the performance, or output, of the behavior. Thus, we needed to test each unique setting for each parameter to ensure that the settings created the desired effects. We also had to test behavior performance in the absence of an input for the optional parameters. In addition to the required and optional parameters, there were other parameters that were consistent across every behavior (e.g., weapons control status and rules of engagement) and some that were independent of the behavior itself (e.g., unit type and echelon). We had to consider these as well.

To return to our example, the Tailgate Resupply behavior has three required parameters: LRP location (any point on the terrain), unit to resupply (the unit that is to receive the supplies), and return location (again, any point on the terrain). It also had one optional parameter: formation (which determines the formation in which the supply vehicles will move). Thus, for these inputs, we needed to test whether the supply vehicles moved to the correct locations, resupplied the proper units, and moved in the correct formation. Additionally, the behavior is designed to work for any type of unit at any echelon (entity, team, squad, company, battalion, etc.).

3.4.1.2 Special Cases

In addition to the inputs that the user can choose, we also wanted to test the robustness of the behavior. For this, we tested cases that would involve the behavior performing at the extremes or under unusual circumstances. For some of the composite behaviors, we determined that just testing the range of inputs was sufficient; however, in most cases, we considered these additional aspects.

For our Tailgate Resupply example, special cases included testing what would happen if the supply vehicles had the wrong supplies, had an excess or shortage of required supplies, had unnecessary supplies, or had to resupply multiple units. Additionally we wanted to test different classes of supplies (e.g., ammunition, fuel, medical supplies, etc.). Such conditions might evaluate realistic situations not considered in the behavior's development.

3.4.2 Combinations

Given the large number of potential inputs and tremendous number of variations the behavior could take, we did not try to test every possible combination of input parameters. For example, the Move Tactically behavior has 16 required and optional parameters, with some having as many as 13 choices, resulting in almost a million unique combinations of parameters. We instead tried to ensure that each critical aspect was tested at least once. For instance, if an input had seven potential settings, we would have at least seven scenarios. Thus, the parameter with the largest number of potential choices tended to drive the total number of scenarios. When determining the unit type and echelon to assign the behavior to for each scenario, we ensured that we varied the unit and echelon across the scenarios, but we did not let that drive the total number of scenarios. With such a significant reduction in the number of scenarios, we had to ensure that we designed each carefully to capture any special cases we identified as well. Such considerations often added one or two scenarios to the final number.

3.4.3 Final Designs

For each of the test designs, we successfully kept the number of scenarios between six and ten. We found that range to be sufficient to test any of the behaviors we verified without taking an excessive amount of time. Our Tailgate Resupply behavior test design consisted of six scenarios.

3.5 Select Criteria

With the test design completed, we then looked to select the evaluation criteria we would use to evaluate the behavior. At a minimum, each input parameter was a criterion to be evaluated to ensure that the input selection properly affected behavior execution. Additionally, there were often other criteria that were not suggested by the inputs, but were still critical to evaluate.

Thus, for the Tailgate Resupply behavior, we were interested in ensuring that the supply vehicles moved to the proper location and in the correct formation, as dictated by the parameter inputs. We were also interested in the amount of supplies delivered and received, as well as the time it took to execute the transfer.

To evaluate the criteria, we used both qualitative and quantitative measures. Many of our measures were qualitative for two reasons. The first is that the data collection functionality of the simulation did not work properly. The second is that many of the criteria could be evaluated visually on the game board (called the Plan View Display, or PVD) during execution. Despite the fact that the data col-

lection functionality was not working, we were still able to collect data from the Status Window. The Status Window showed, for each unit or entity, nearly real-time information, such as speed, orientation, levels of supply, location, etc. Thus, we could pause the simulation at a point of interest and collect data from that window.

Error! Reference source not found. shows a portion the spreadsheet that we used to record our quantitative (data) and qualitative (visual) collection plan. In it, each condition to be verified is listed in the left column, with the next column representing the parameter setting. The next four columns alternate between the collection plan (visual then data) and corresponding results of the testing. The last two columns represent an assessment of the behavior performance (green, amber, red) for each condition and any relevant discussion. The red comment triangles in the left column represent comments entered to clarify the intent of each condition. The snapshot in **Error! Reference source not found.** shows only a subset of the overall spreadsheet without entries.

In the Tailgate Resupply behavior, we evaluated the following criteria visually: movement formation and movement to the correct locations. Quantitatively, we collected data on the transfer of the types and amounts of supplies and the specific units and entities that participated in the operation. However, there was at least one criterion that we were unable to collect at all – the time it took to transfer supplies from one vehicle to another. This was due to the fact that the Status Window had update delays that significantly impacted our ability to determine the relatively short transfer times.

Overall, the typical time we spent creating the test designs and determining the criteria was two to five days, driven, to a large degree, by the developers' responsiveness to our requests for clarification.

3.6 Execute Tests and Analyze Results

With the test design and evaluation criteria determined, we then set up the scenarios in the simulation. We attempted to keep the scenarios simple and to configure them in a way that would provide unambiguous results, instead of being too concerned about tactical validity. In many cases, each composite behavior we tested required us to learn a particular functionality that we had not used for previous behaviors. Thus, this initial portion of execution often consumed a significant amount of time, on the order of two to three days. Often, we would identify conditions that were not, in fact, testable, leading to minor modifications of the design.

Once we created the scenarios, we simply observed and collected data. Sometimes, an interesting or ambiguous result would lead us to run additional excursions with minor variations to understand what was happening. As with scenario development, we sometimes encountered situations during execution that would lead us to alter the overall test design. While we usually ran each scenario numerous times to ensure that it was set up properly, we normally used only the data from the last run for reporting purposes, unless we noticed large variations in output during our trial runs. All behaviors we have examined to date have been deterministic, although the stochastic nature of other aspects of the model still causes variations in output between runs.

Our primary concern in this verification effort was to ensure that we thoroughly recorded everything we did throughout the process, especially given the fact that our resource constraints limited the number of unique cases we could observe. We kept very detailed records in spreadsheet form that delineated our test design, the evaluation criteria, and our results (Table 1).

Table 1: Verification Collection Plan and Recording Spreadsheet

VERIFICATION PLAN & RESULTS							
		Visual Verification Plan	Visual Results	Data Verification Plan	Data Results	Status	Discussion
TASK DIALOGUE SETTINGS							
REQUIRED PARAMETERS							
LRP Location	See Scenario File						
Unit to Resupply	Section A, Armor Platoon 1						
Return Location	See Scenario File						
OPTIONAL PARAMETERS							
Formation	Vee						
OTHER							
Resupply Time	N/A						
Supplies Delivered	N/A						
Supplies Received	N/A						
Supply Accuracy	N/A						

As part of that, we often captured screenshots of particularly interesting phenomena that would be difficult to explain otherwise. Additionally, we saved all of the scenario files we used, to include any excursions we ran, so that we could include those with our reports. The average time consumed by scenario development and execution was typically five to seven days.

3.7 Report and Document Tests and Results

After the completion of each behavior, we compiled the information collected in the spreadsheets, along with the scenarios, and sent them directly to the OOS development team. In addition to reporting the results of the behavior verification itself, we also reported any documentation errors or shortcomings that we identified, as well as any general software performance issues we had encountered.

As previously discussed, the documentation was captured in two spreadsheets. The first spreadsheet contained a worksheet for each of the developed scenarios. In each, we recorded relevant information about the types and sizes of the units involved, the environmental conditions, and any special cases we examined. Additionally, the spreadsheet contained an overall evaluation of the behavior performance for that scenario (green, amber, red) as well as the collection plan and results as show in Table 1. The second spreadsheet was a working spreadsheet that contained a worksheet for each behavior evaluated. In that worksheet, we summarized the results for each scenario and provided an overall evaluation for the behavior. In addition, we discussed any documentation discrepancies we identified. In retrospect, a database would have been more appropriate and versatile; however, the spreadsheet did meet our needs.

4 RESULTS

4.1 Challenges.

One primary challenge we faced had to do with documentation. To the OOS team's credit, they did a good job of extensively documenting, especially during the initial knowledge acquisition / knowledge engineering process. Those documents provided a tremendous amount of information about the actual real-world behaviors. However, as with any major software development effort, documentation of the actual implementation often lagged behind development. In some cases, the documentation had not kept up with the newest releases of the simulation. In others, the documentation of how the behavior should perform did not have the detail we required to understand how the model should perform under varying parameter inputs and special cases. As we discussed previously, we were often able to fill in the documentation gaps through discussions with the developers.

When we were unable to obtain information we sought from the documentation or developers, we sometimes had to rely upon our own operational expertise to understand what the model should do. However, we had to take great care not to draw conclusions about behavior performance based upon our assumptions. Thus, when a behavior failed to perform in accordance with our assumptions, we had to avoid saying "Based upon our experience, behavior X should do Y; thus, because it did not do Y, it fails." When we encountered these situations we made note of what we assumed should happen and what did happen, and then labeled the behavior performance as "inconclusive" or "unable to verify."

Another challenge had to do with the software's data collection functionality. The data collection capability had not been finalized when we began the verification process. While we were able to work around that by using the Status Window, the accuracy of our results was impacted. For instance, while location was reported in the Status Window, to verify the distance between two vehicles we would have to determine the location of the two vehicles in the Status Window and calculate the distance manually. However, because the distance may vary over time due to terrain, we needed an average of values, making the process very tedious. In some cases, as in the discussion previously about not being able to measure the supply transfer times in the Tailgate Resupply behavior, we were unable to collect the data at all.

4.2 Feedback

Despite the challenges we faced, we were able to conduct thorough verifications that greatly benefited the OOS program. In correspondence sent directly to us from the PM OneSAF office, they remarked that "We can use feedback like this to make the behaviors robust. [We] especially liked the fact that TRAC Monterey did not stop when they encountered an issue but instead moved forward to use the other tools OOS offers to get to the end result." Additionally, they remarked that: "We are getting useful, specific, actionable feedback on documentation mistakes, deficiencies, and incompleteness." Thus, despite our limited manpower and short timeline, we were able to develop and execute a methodology to satisfy the OOS team and impact the program.

5 CURRENT AND FUTURE EFFORTS

As of the writing of this paper (June, 2006), we have completed the verification of 16 orderable composite behaviors. Overall, each composite behavior verification took between two to three weeks to accomplish from start to finish. We will continue our verification effort through the summer of 2006.

As the project draws to a close, we have identified two promising future research needs suggested by our efforts. The first is to develop a universal behavior modeling framework upon which all common Army simulations base their behavior models. Such a framework is necessary for comparing the results of two separate simulations in support of the same study. The second is to begin creating the behaviors that are expected to be used by future forces in the accomplishment of their mission, instead of focusing on current behavior models. After all, most of the combat modeling used within TRAC is used to support decisions relating to the acquisition of future equipment or future changes to organizational structures. The exploration of these two research areas would greatly benefit TRAC in the performance of its mission.

6 CONCLUSIONS

Our work demonstrates that with a solid methodology and careful test design, we can conduct a thorough verification of a relatively new combat modeling functionality – behavior modeling – even under tight resource constraints. The implementation of such functionality is only in its infancy. We must continue to develop and refine methods to verify the performance of these cutting-edge human behavior representations as they develop.

Additionally, our work demonstrates the critical importance of documentation. Without thorough documentation of the developer's conceptual model, we cannot truly verify a behavior; we can only surmise based on what we think should happen. Documentation must start on day one and continue to be updated as the software evolves.

While our effort has impacted the OOS program, much remains to be done, even after its successful release. Hopefully, our methodology can serve as a basis for future efforts as the program continues to evolve.

ACKNOWLEDGMENTS

This study was supported by the PM OneSAF team, who provided a tremendous amount of information, training, and resources. The views expressed herein are those of the authors and do not purport to reflect the position of PM OneSAF, the TRADOC Analysis Center, the Department of the Army, or the Department of Defense.

REFERENCES

DMSO. 2000. Defense Modeling and Simulation Office. VV&A recommended best practices guide. Available via <http://vva.dmso.mil> [accessed November 2, 2005].

HQDA. Department of the Army Headquarters. 1999. *Department of the Army Pamphlet 5-11: Verification, Validation, and Accreditation of Army Models and Simulations*. Washington, D. C.

Henderson, C., and B. Grainger. 2002. Composable behaviors in the OneSAF Objective System. In *Proceedings of the Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2002*, Orlando, Florida.

Randolph, W., and D. Sagan. 2003. OneSAF uses a repeatable knowledge acquisition process. In *Proceedings of the Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2003*, Orlando, Florida.

SAIC. 2004. Science Applications International Corporation. Models development behavior verification test plan. Orlando, Florida.

AUTHOR BIOGRAPHIES

ERIC S. TOLLEFSON, is a Military Operations Research Analyst at the TRADOC Analysis Center in Monterey, CA. He was formerly an Assistant Professor at the United States Military Academy (USMA), West Point. He has served in various infantry positions, including platoon leader, company commander, and as a staff officer. He holds a BS in Engineering Physics from USMA, 1994, and a M.S. in Operations Research from the Georgia Institute of Technology, May 2002. His e-mail address is <eric.tollefson@us.army.mil>.

HAROLD M. YAMAUCHI, is a Software Engineer at Rolands and Associates Corporation. He currently works at the TRADOC Analysis Center in Monterey, CA. He holds a B.A. in Statistics from the University of California, Berkeley and a M.S. in Operations Research from Oregon State University, 1984. His e-mail address is <hmyamauc@nps.edu>.

JEFFREY B. SCHAMBURG, is the Director of the TRADOC Analysis Center in Monterey, CA, and formerly an Assistant Professor and an Operations Research Analyst in the Department of Systems Engineering and the Operations Research Center at West Point, NY. Lieutenant Colonel Schamburg received his commission as an Infantry Officer and a B.S. in Civil Engineering from the United States Military Academy, West Point, in 1986. He received his Ph.D. in Systems and Information Engineering from the University of Virginia in 2004. His present research interests include data mining, predictive modeling, and response surface methodology with applications to military, information, and industrial systems. His e-mail address is <jeffrey-schamburg@us.army.mil>.