Proceedings of the 2001 Winter Simulation Conference
B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, eds.

USAGE TESTING OF MILITARY SIMULATION SYSTEMS

Gwendolyn H. Walton
Robert M. Patton

School of Electrical Engineering and Computer Science
University of Central Florida
Engr Bldg Room 407, P.O. Box 162450
Orlando, FL 32816-2450, U.S.A.

Douglas J. Parsons

U.S. Army Simulation, Training
and Instrumentation Command
AMSTI-ET
12350 Research Parkway
Orlando, FL 32826-3276, U.S.A.

ABSTRACT

Scalability and input domain explosion make it impossible to exhaustively test simulation systems. Improved methods such as statistical usage testing are needed to provide quantitative support for test planning and test management. This paper describes the challenges and the state of the practice of testing simulation systems. A brief introduction to statistical usage testing is provided. An approach to developing an abstract usage model structure appropriate for testing military simulation systems is suggested and illustrated. This approach supports the creation and analysis of test scenarios that are flexible enough to handle a wide range of uses in military simulations.

1 INTRODUCTION

A variety of authors have written in the publicly available literature concerning the challenges in testing simulation systems. (For example: Balchi 1995, Birta and Ozmizrak 1996, Gonzalez and Dankel 1993, Hartley 1997, Hopkinson and Sepulveda 1995, Jacobson and Yucesan 1992, Page and Smith 1998, Shannon 1992, and Smith 1998.) Three major challenges are scalability, size of the input and output domains, and the need to tailor the verification and validation processes to meet individual needs. As discussed in Section 2, these and other challenges make it impossible to exhaustively test simulation systems. Improved methods are needed to provide quantitative support for test planning and test management.

The DMSO Recommended Practice Guide (2001) defines validation of simulations as "the process of determining the degree to which a model or simulation is an accurate representation of the real world from the perspective of the intended uses of the model or simulation." According to Hartley (1997), the proper motive for validation and verification is the intended use of the model. Robinson (1997) states that a model can only be shown valid if it is demonstrated to be accurate for the intended purpose. Sargent (1998) echoes the DMSO definition by describing validity as being "concerned with determining that the model's output behavior has the accuracy required for the model's intended purpose over the domain of its intended applicability." According to DMSO (2001), validation scenarios should "sufficiently characterize the probability distributions representing areas where the simulation's behavior is stochastic", and validation scenarios should "sufficiently characterize the behavior of the simulation that the application will use most and, therefore, will introduce the greatest probability of errors occurring."

The DMSO Recommended Practice Guide (2001) further states that: "Good scenarios will produce testing results that sample a simulation's behavior sufficiently enough to enable accurate assessments of a simulation's validity for some application. A simulation with poorly understood behavior requires scenario designs that generate enough data to sufficiently characterize those regions where a simulation behaves well and poorly."

The benefits of statistical usage testing for supporting test planning, test generation, test automation, and software certification have been demonstrated in numerous industry and government software development projects in a variety of domains. (For example: Agrawal and Whittaker 1993, Sherer and Walton 1998, Walton et al. 1995). These successes and the above statements from the DMSO Recommended Practice Guide on Validation and Verification justify the investigation of the applicability to statistical usage testing to military simulation systems.

2 TESTING CHALLENGES

In military simulations it is possible that many different groups of objects exist at one time, with each group possibly a different size. There is a scalability issue with regard to the number of different objects that can coexist in the

simulation at one time. There is also a scalability issue with regard to the number of objects of the same type that can coexist. Each of these scalability issues can create a combinatorial explosion of objects to be tested. For large systems, due to time and resource constraints, not all combinations can be tested. For instance, suppose a tester chooses to have four different kinds of objects with two objects of each kind. The simulation system may be able handle this configuration successfully, but may not be able to handle the addition of a fifth different kind of object. If the tester only tests four kinds of objects, the defect that causes the system to fail at the fifth kind of object will not be surfaced. To complicate matters, military simulations can cover a wide variety of terrain. As a result, to fully test the simulation system, all the combinations of different types of objects and number of objects must be combined with the each of the different types of terrain.

A related problem to scalability is the problem of the size of the input and output domains. As a military simulation scales up, the input domain can grow exponentially. For example, the simulation can include any of a variety of vehicles, troops, and structures that can be combined in a variety of ways. In addition, there are many mission types and terrains on which the vehicles, troops, and structures may be used. Furthermore, there are a wide variety of environmental conditions such as night, day, and type of weather that may be added to the simulation. Each object has a variety of behaviors that it can perform. Even for a very small-scale simulation, the combination of these input factors is quite large. When the simulation is scaled upward, the input domain will scale at a much faster rate.

The output domain may not grow as quickly as the input domain. However, information about the output domain may be incomplete, and there can be multiple possible responses for the same input data. For example, when a simulation runs with human-in-the-loop, the outcome for a given set of input data can be different each time the simulation is run. In addition, when a simulation includes models of future environments, scenarios, or weapons, field testing is not possible and the expected output may not be completely specified. Furthermore, the details of some combat scenarios may not be well understood, and their outcomes can be determined by or influenced by a human decision making process that is not well understood. Consequently it is often difficult, if not impossible, to completely identify the output domain.

When it is not possible to determine whether the outcome of a simulation is correct, the verification and validation process is used to build the user's confidence in the simulation system. This motivates verification and validation plans tailored to the user's needs and emphasizing the issues most important to the user.

3 CURRENT TESTING PRACTICE

A variety of verification and validation techniques have been developed and employed on simulation systems. These techniques have been categorized in different ways by (Balci 1994 and Sargent 1998) Most of the validation and verification techniques discussed in the simulation literature are standard software engineering techniques. These will not be discussed here. However, a few techniques specific to simulation systems have been reported. These simulation specific techniques are generically categorized by (Sargent 1998) as either subjective or objective. Important issues of each category are summarized below.

3.1 Subjective Testing

Subjective testing approaches rely heavily on expert opinions, or third party evaluation. The simulation is run; the expert evaluates the simulation experience and results based on some specific criteria such as Turing tests or face validity; and comparison to other valid models may be performed. (Sargent 1998)

The advantage of the subjective approach is that there are some aspects of a simulation that cannot be performed objectively, or cannot be specified in a detailed manner such as the audio or visual aspects of a simulation. For instance, suppose a military simulation employed the sound of gunfire as part of the simulation. How do you objectively evaluate that a rifle does not sound like a small handgun? How do you specify in a detailed manner or objectively evaluate that a machine gun does not sound like a drum roll?

One disadvantage of the subjective approach is that some aspects of the system may be inadvertently overlooked. Another disadvantage is that the evaluation of the simulation may change depending on which "expert" is asked to do the evaluation. Another consideration is the number and type of experts used. Some simulations may be so large and complex that a variety of experts in different fields may be needed. Consequently, the results of their review may vary, making it difficult to determine the overall validity of the simulation. Despite these drawbacks, the use of expert opinion or third party evaluation provide an invaluable approach to validating and verifying aspects of a simulation to which there is no alternative approach, especially for human-in-the-loop systems such as military simulations.

3.2 Objective Testing

Objective testing approaches rely less on expert opinions and more on statistical and automated methods. These approaches typically require that the system be observable, meaning that data of particular variables can be recorded as the simulation is running. This data that is recorded can then be statistically compared to a set of data that has been re-

corded from the real system, or can be analyzed in some other way by domain experts. For military simulations, this approach can be particularly useful for comparing such things as a weapon's rate of fire, the hit/miss ratio, the speed of a vehicle over a particular terrain, or environmental conditions. Statistical techniques provide insight into the accuracy of input-output data sets of the simulation. There exist a variety of statistical approaches that can be successfully used, including confidence intervals, hypothesis tests, data plots, and multivariate statistical approaches.

Another approach is the use of a validation knowledge base (VKB) (Birta and Ozmizrak 1996). This approach incorporates expert knowledge along with any observable data from the system to determine validity of a simulation. One advantage of this technique is that it attempts to formally collect all aspects of the expected behavior as determined by experts and then compare the expected behavior with the observed data. Another advantage is that this approach can be easily automated so that test results can very quickly be analyzed.

However, there are several disadvantages to using a validation knowledge base. First, there is the experiment design problem described by Birta and Ozmizrak (1996). Once the knowledge based expert system has been developed, how does one choose experiments such that the knowledge base is covered efficiently and effectively? According to Birat and Ozmizrak, validation "requires generating all possible behavior instances which, of course, is not possible in practice. Therefore, from a practical point of view, it is necessary to regard validation as the process of showing that any particular subset of a dynamic object's behavior is consistent with its VKB."

The subset of behavior should cover as much of the VKB as possible. The subset should also minimize the number of tests that need to be done since each test will be quite lengthy and consume substantial resources. This is a difficult problem. Birta and Ozmizrak suggest that a solution can be achieved by properly setting up an experiment design problem and, with additional restrictions, as a constraint satisfaction problem. More than one optimal subset may need to be determined depending on testing objectives, the fault-finding abilities of the subset, and the cost-effectiveness of the subset. Another disadvantage is that the VKB inherits all the challenges of the development and validation of an expert system.

Expert systems have also been applied to the validation of military simulation training systems. According to Hopkinson and Sepulveda (1995), expert systems provide an excellent method for real-time evaluation of a trainee's performance. Their expert system was developed using case-based reasoning to evaluate the trainee's decisions as the simulation progressed. The approach to developing the expert system was to identify the invalid techniques, which led to favorable outcomes. Hopkinson and Sepulveda (1995) refer to this as a "sandbox" approach, which encourages creativity on the part of the trainee but without letting the trainee go outside specific boundaries. An advantage of this approach is that real-time evaluation of a trainee can be performed automatically. Also, by focusing on invalid uses and techniques, the number of test cases needed to effectively and efficiently cover the known input domain can be significantly reduced. A disadvantage of this system is that case-based reasoning systems are restricted to variations of known problems. This can potentially cause the validation of such a simulation to be unnecessarily biased toward or against particular problems. In addition, this approach does not assist with the test planning process or with test scenario generation.

These techniques each attempt to address a specific aspect of testing simulation systems. They each have advantages and disadvantages. When viewed as a whole, they cover a very broad spectrum of the problems with testing simulation systems. However, from the discussion of techniques in the literature, it is evident that the problem of effectively and efficiently validating the behavior of a simulation system and the need to use domain-specific knowledge persists.

3.3 Improvements Needed

Increasing the user's confidence is one of the primary goals of validation and verification of simulation systems. Thus, an improved testing approach would be to identify fewer test cases that cover the requirements but are more general in nature with respect to the requirements, providing a better approximation to the way in which the user will use the system. By doing this, the tester may be more likely to identify problems that the user will find. If these problems can be caught and fixed before the user sees them, then the confidence of the user in the system can be increased.

Current simulation testing methods often make little use of automation. As a result, very little testing can be accomplished in a given amount of time because each test case can be tedious to setup, and it can be time consuming to execute each test case and record the results. A considerable amount of time is also required to analyze and understand the results and to determine if an error found in the results is caused by an improper conceptual model or improper implementation of that model. Thus, any automation of any step in the testing process could provide invaluable benefits in terms of time and cost savings.

4 STATISTICAL USAGE TESTING

From the user's perspective, a software system is essentially a translator. The user provides input, and the system translates this input into some set of output. To the user, how the system performs this translation does not matter, as long as the desired output is produced. Testing a software system according to inputs and outputs is called black

box testing, or functional testing. Essentially, the black box testing process provides the system with a specified set of inputs and compares the output with the desired output to determine accuracy and precision.

One drawback to black box testing is that the number of possible inputs can grow exponentially as the complexity of the system grows. Testing all possible inputs very quickly becomes impractical. However, because there are certain inputs in the input domain that the user may never use, it is possible to help reduce the number of inputs that need testing by considering the way in which the user will use the system. By doing this, the tester can perform fewer tests while simultaneously gaining an understanding of the intended use of the system and its expected reliability. This type of testing is known as usage testing.

The justification for usage testing is well grounded in software engineering theory and practice. Adams (1984) showed that the vast majority of software failures observed during operation of software are caused by a small proportion of software faults, and a great proportion of latent software faults will very rarely, if ever, result in observed failures of the software in practice. Adams confirmed this failure/fault distribution across nine different major commercial systems developed by IBM. Mills (1992), Musa (1993), and others (for example, Walton et al. 1995) built on Adams' findings and observed that targeted testing towards high usage software can be very successful in reducing the number of operational failures. Fenton and Ohlsson (2000) verified Adams' findings on two major consecutive releases of a large legacy project developing switching systems.

An efficient method for developing a plan for usage testing is to create a model of the intended use or behavior of the system. Such a model can be used to automatically generate a series of test cases.

Military simulations are stochastic in nature. Markov chain usage models are also stochastic in nature, and the test cases that are generated from a Markov chain usage model will reflect this. The testing process for military simulations should take advantage of this commonality.

Markov chain usage testing is a well-defined, rigorous method, which readily lends itself to automation. If the usage model is implemented as a Markov chain, the model and the test cases can be analyzed using Markov chain mathematics to determine such things as reliability, mean time to failure, and confidence values concerning how much testing of the system's uses is included in the series of test cases (Whittaker and Poore 1993, Whittaker and Thomason 1994).

Markov chains are comprised of states and arcs, which graphically describe the expected operational use of the software system. In the software engineering literature, the states are often states of use of the software and the arcs define an ordering that determines the event space, or sequences, of the experiment.

Figure 1 shows a Markov chain usage model that describes a greatly simplified operational profile for a US Army Computer Generated Forces (CFG) simulation system. Table 1 lists a subset of state designations used in the model of Figure 1.

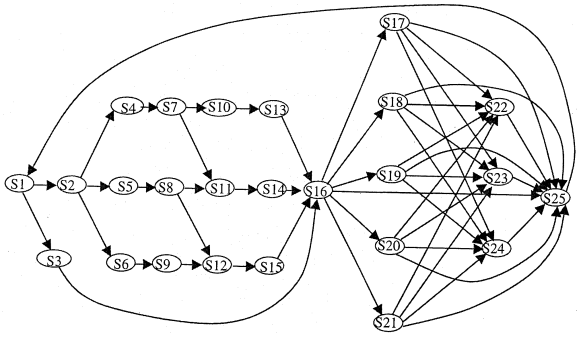

Figure 1: CGF Simulation System Markov Chain Model.

Table 1: State Definitions for Figure 1

State	Definition
1	System ready for instruction
2	Select "New Scenario"
3	Select "Load Saved Scenario"
4	Open Unit Editor and create vehicle
5	Open Unit Editor and create platoon
6	Open Unit Editor and create company
7	Assign vehicle task
…	…
25	End Scenario

Markov chain usage models can support the quantitative analysis of test plans and test results. For example, Table 2 shows a sample of the Markov chain analysis results for Figure 1 when state-to-state transition probabilities were based on the author's experience working with the CGF simulation users.

Table 2: Markov Chain Analysis Results for Figure 1

Expected Script Length	8.80
Long run probability for state 'S2'	0.1020
Long run probability for state 'S3'	0.0110
Long run probability for state 'S4'	0.0300

Markov chain models can be used to automatically generate statistically correct samples of test cases. Simply by changing the arc probability values, the tester can significantly alter the kind of test cases that can be generated. This greatly reduces the time spent on creating test cases and allows more time to be spent on the analysis of test results.

Testing based on Markov chain usage models yields information that can be used to identify and document the aspects of the software that have been tested, how often they have been tested, and which aspects have been ignored. For regression testing, this information can be invaluable. It can be tedious and difficult to identify which parts of a knowl-

edge base have been tested. With Markov chain usage models, the tester can very clearly identify what areas of the models and what percentage of the model has been covered.

Domain specific knowledge can be used to help build usage models, which are then used to test the software. Domain knowledge can be quite valuable in that it provides the tester with knowledge of potential weaknesses in the software and insight into the way in which the software will be used. Furthermore, usage models can be based on procedures that explicitly define behavior or use of the system such as control systems, or they can be based on some abstraction of the behavior of the system. Thus, usage modeling can be used to support both functional verification and system validation.

With the application of usage modeling and Markov chain analyses, the test plan can focus separately on different use cases of the user's domains. Markov chain usage models can also be used to model different types of missions where the behavior will change based on the mission, type of terrain, and time of day. For example, a usage model can reflect whether a mission is a reconnaissance, search and destroy, or search and rescue mission.

By carefully applying Markov chain based usage testing, we expect that test scenarios can automatically be generated that achieve the goals and guidelines set forth by the DMSO. This practice should ultimately improve the state of the practice of testing military simulation systems. However, an analysis of the model of Figure 1 indicates that, using this type of model syntax, the number of states could quickly grow out of control. Useful abstractions and usage model syntax are needed. A means to address this issue is discussed in the following section.

5 USAGE MODEL GUIDELINES AND SYNTAX

Usage models directly affect the level of effectiveness and efficiency of the testing process. Consequently, it is imperative that usage models be carefully developed according to the user's point of view and that usage models provide flexibility to the tester.

A number of challenges exist when developing usage models. If there is too much detail, the model may not have the flexibility to be reused. However, if there is not enough detail, the model may not sufficiently test all aspects of the use of the system. Clearly, the perfect amount of detail can be elusive.

Usage models for military simulations must sufficiently model the scenarios that are of interest to the user. According to the HLA Federation Development and Execution Process document (2001), the primary input to the activity of developing scenarios is "the operational context constraints specified in the objectives statement." In addition, this HLA document states: "A federation scenario includes the types and numbers of major entities that must be represented by the federation, a functional description of the capabilities, behavior, and relationships between these major entities over time, and a specification of relevant environmental conditions that impact or are impacted by entities in the federation. Initial conditions (e.g., force lay downs), termination conditions, and specific geographic regions should also be provided."

An effective usage model must promote dialogue between the customer, developers, and testers. This dialogue should enhance the clarity of the requirements, which ultimately will reduce development and testing costs. If the usage models are to be used to verify requirements with the customer, then the usage models should be simplistic enough to be understood by the customer, but capable of being used by the testing team. Finally, the usage models need to be easily verified, reused, and maintained. Usage models that cannot be easily verified run the risk of not accurately portraying the use of the software.

Walton et al. (1995) provided the following guidelines for creating effective usage models:

1. Identify the test objectives.
2. Identify the test constraints.
3. Determine test environment and automation issues.
4. Define the boundaries of each test.
5. Define "user" and "use" for the purpose of each test.
6. Determine appropriate usage model strata.
7. Develop and document usage model structure for each model.
8. Determine transition probabilities for each model.
9. Verify the model.
10. Iterate as needed.

For usage models of military simulation systems, we recommend the following additional guidelines for use when performing steps 6 and 7.

- Specify a specific geographic location at the beginning. Do not change this location at any point in the usage model.
- Identify force lay downs at the beginning.
- Weather conditions can change at various points in the usage model.
- Formation can change at various points in the usage model.
- Number and type of targets can change at various points in the usage model.
- Change the model structure as needed to add realism, randomness, and variety of scenarios.
- Use mission types, mission objectives, and location of mission to determine usage model strata.
- Use tactics and procedures to determine and verify the model structure.

- Change the model probabilities to reflect test objectives and constraints.
- Use uniform arc probabilities to increase model entropy.

When performing steps 6 and 7, it must be understood that, when executing test scripts derived from usage models, the amount of detail given in the model is a critical issue. Thus, a significant amount of thought is needed for determining the model strata and transition probabilities. As reported in the literature, there are a variety of syntactic ways to develop usage model structures. Usage models have been developed for user interfaces (For example, Sherer and Walton 1998), and for control systems in which the structure of the behavior and inputs to the system were already clearly documented (For example, Agrawal and Whittaker 1993). For these examples, the states represent internal states of the software and the transitions between states represent different input to the system.

While these approaches are useful, they do not always appropriately handle military simulations. Consequently, we suggest a new syntactic approach:

- Identify the states of a usage model with the basic behavioral units of a simulation system.
- Transitions to these states are given labels that describe how those basic behavioral units should be performed.

When developing models for simulation systems, we recommend that the states represent the functions or behaviors of interest in the software being tested and the arcs represent the different sequences or combinations of those functions. The states and arcs comprise the structure of the model.

This approach to developing the usage model structure helps to reduce the number of necessary states, and helps create test scenarios that are flexible enough to handle the wide range of uses in military simulations. This proposed syntactic approach is demonstrated in the following section.

6 EXAMPLE USAGE MODEL

According to Smith (1998), a generic model that is used in the development of military simulations is that of "Move-Look-Shoot." This sequence of events attempts to model the general behavior of men and machines during combat. Basically, a unit moves from its point of origin to some specific location, looks for targets, and then shoots at targets. This sequence can be repeated multiple times, or performed in a variety of sequences.

The usage model shown in Figure 2 was developed based on this generic model. The structure of the model for this example is intended to be simplistic. As appropriate, the model can be made more complex and realistic using domain specific knowledge for a particular simulation system. The model consists of five states: Start, Move, Look, Shoot, and End. (For continuous application of the model, there is an implied arc from "End" to "Start".)

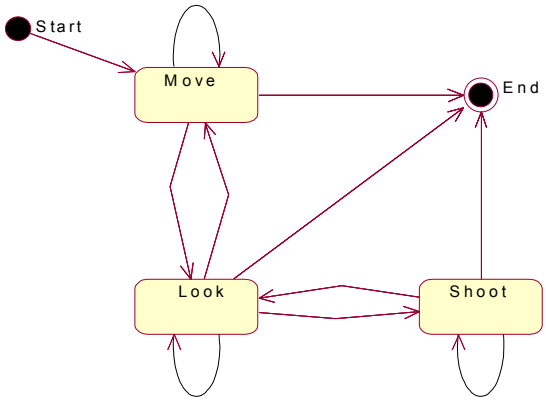

Figure 2: Usage Model of Move-Look-Shoot Behavior

Each of the Move, Look, and Shoot states could be composite states in which there is a more detailed model contained within the state. For instance, there may be different movement techniques based on the mission type. Also, the entire five-state model of Figure 2 could be contained within a higher abstraction layer.

In Figure 3 below, we have extended the usage model of Figure 2 according to the guidelines previously specified.

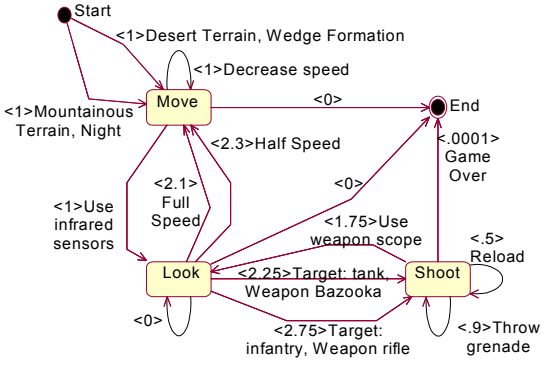

Figure 3: Move-Look-Shoot with Uniform Arc Values

Markov chain analysis techniques (as described in Walton and Poore 2000, Walton et al. 1995, Whittaker and Poore, 1993, and Whittaker and Thomason 1994) are applied to the model. We examine the long run probability values for the three major states (Move, Look, and Shoot), and also the expected test scenario length. As indicated in Table 3 below, the state 'Look' has the highest long run probability. This means that 'Look' is more likely to show up in the generated test scripts than the other states, and, for use of the software as described by this usage model, the software will spend most of its time in the 'Look' state.

Table 3: Markov Chain Analysis Results

Expected Script Length	95207.00
Long run probability for state 'Move'	0.3235
Long run probability for state 'Look'	0.3456
Long run probability for state 'Shoot'	0.3309

The model of Figure 3 focused on fairly generic behavior model that is applicable to a variety of vehicles and soldiers. However, there are some units in military simulations whose behavior or use cannot be modeled in such a way. An example of such a unit is an air defense unit. These units are typically either non-moving units or units which move infrequently. They are composed on some type of radar and weapon. Furthermore, air defense units only target aircraft and typically do not have any ground defense of their own. To handle units such as this, the usage model of Figure 3 has been revised by adding another level of detail to the 'Look' state. The revisions are illustrated in Figure 4.

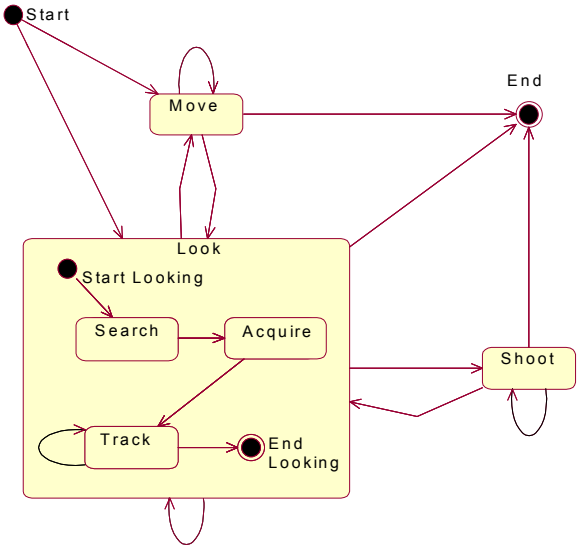

Figure 4: Simple Usage Model for Air Defense Units

The structure of the simple usage model of Figure 4 assumes that the Fire command is not automated. Air defense units begin by searching for new targets. Once a target is within range of the radar, the air defense unit acquires the target and begins tracking it. Once a target is being tracked, a fire command is issued, and the air defense unit begins firing at the target.

Transitions between these states are dependent strictly upon three types of input. The first type of input is the target aircraft. The target aircraft must be flying within range of the radar and without an unobstructed line of sight. In addition, for certain types of air defense units, the aircraft must be flying a specific altitude. (as described by the Air Defense Artillery web site 2001) The second type of input is the terrain. Depending on the terrain, an aircraft may temporarily be invisible to the radar despite being within range of the radar. For example, in a mountainous terrain, an aircraft may become visible to the radar, but then become invisible if the aircraft moves below the mountaintops such that the mountains obstruct the line of sight of the radar. The third type of input is the fire command. A fire command determines when to fire and at which target to fire. On some units, this may be automated.

Arc labels and arc probabilities can be added to the model to provide more descriptions about the input necessary to transition between states. Depending on the probabilities selected, the average test case length will increase or decrease. Furthermore, changes in probabilities will change the long-run probability of occurrence of each state in test cases generated from the model. A careful analysis of Markov chain mathematical results with respect to test objectives will assist the tester in selecting arc probabilities.

More detailed usage models are easily developed and documented by the addition of lower-level models and additional arcs to describe data partitions of the input domain at the desired level of detail. Generated test scripts from this sort of abstract model were similar to test cases developed by hand for the functional testing of the Army OneSAF Testbed Baseline program.

7 CONCLUSIONS

Usage testing methods based on Markov chain models can support effective and efficient validation of the behavior of military simulation systems. An effective usage model is built from domain-specific knowledge and incorporates test management objectives and constraints. The modeling process promotes dialogue between the customers, developers, and testers. This dialogue should enhance the clarity of the requirements, which ultimately will reduce development and testing costs and risks. Markov chain model-based usage testing can support increased test automation and quantitative analyses of test plans and testing results.

REFERENCES

Adams, E.N. 1984. Optimizing preventive service of software products", *IBM Journal for Research and Development*, Vol 28, No. 1, January 1984, 3-14.

Agrawal, K. and J. A. Whittaker.1993. Experiences in applying statistical testing to a real-time, embedded software system, *Proceedings of the Pacific Northwest Software Quality Conference*, Portland, OR, October 1993.

Air Defense Artillery. 2001. http://www.airdefense artillery.com, accessed March 30, 2001.

Balci, O. 1995. Principles and techniques of simulation validation, verification, and testing. In *Proceedings of the 1995 Winter Simulation Conference*, ed. C. Alexo-

Alexopoulos, K. Kang, W.R. Lilegdon, and D. Goldsman, ACM, 1995, 147-154.

Balci, O. 1994. Validation, verification, and testing techniques throughout the life cycle of a simulation study. In *Proceedings of the 1994 Winter Simulation Conference*, ed. J.D. Tew, S. Manivannan, D.A. Sadowski, and A.F. Seila, ACM, 215-220.

Birta, L.G. and F.N. Ozmizrak.1996. A knowledge-based approach for the validation of simulation models: the foundation. *ACM Transactions on Modeling and Computer Simulation*, Vol. 6, No. 1, Jan. 1996, 76-98.

Currit, P.A., M. Dyer, and H.D. Mills. 1986. Certifying the reliability of software. *IEEE Transactions on Software Engineering*, Vol 12, No 1, January 1986, 3-11.

DMSO. 2001. http://www.dmso.mil, accessed March 30 2001.

Fenton, N.E. and N. Ohlsson. 2000. Quantitative analysis of faults and failures in a complex software systems. *IEEE Transactions on Software Engineering*, Vol 26, No. 8, August 2000, 797-814.

Gonzalez, A.J. and D.D. Dankel. 1993. *The Engineering of Knowledge-Based Systems*. Prentice Hall, Englewood Cliffs, N.J., 1993.

Hartley III, D.S. 1997. Verification and validation in military simulations. In *Proceedings of the 1997 Winter Simulation Conference*, ed. S. Andradottir, K.J. Healy, D.H. Withers, and B.L. Nelson, ACM, 925-932.

HLA Federation Development and Execution Process. 2001. http://www.dmso.mil/index.php? page=410, accessed March 30, 2001.

Hopkinson, W.C. and J.A. Sepulveda. 1995. Real time validation of man-in-the-loop simulations. In *Proceedings of the 1995 Winter Simulation Conference*, , ed. C. Alexopoulos, K. Kang, W.R. Lilegdon, and D. Goldsman, ACM, 1250-1256.

Jacobson, S.H. and E. Yucesan. 1992. Building correct simulation models is difficult. In *Proceedings of the 1992 Winter Simulation Conference*, ed. J.J. Swain, D. Goldsman, R.C. Crain, and J.R. Wilson, ACM, 783-790.

Mills, H.D. 1992. Certifying the correctness of software. In *Proceedings of the 1992 International Hawaii International Conference on Systems Sciences, Vol II, Software Technology*, ACM, 373-381.

Musa, J. D. 1993. Operational profiles in software reliability engineering. *IEEE Software*, March 1993, 14-32.

Page, E.H. and R. Smith. 1998. Introduction to military training simulation: a guide for discrete event simulationists. In *Proceedings of the 1998 Winter Simulation Conference*, ed. D.J. Medeiros, E.F. Watson, J.S. Carson, and M.S. Manivannan, ACM, 53-60.

Robinson, S. 1997. Simulation model verification and validation: increasing the user's confidence. In *Proceedings of the 1997 Winter Simulation Conference,* ed. S. Andradottir, K.J. Healy, D.H. Withers, and B.L. Nelson, ACM, 53-59.

Sargent, R.G. 1998. Validation and verification of simulation models. In P*roceedings of the 1998 Winter Simulation Conference*, ed. D.J. Medeiros, E.F. Watson, J.S. Carson, and M.S. Manivannan, ACM, 121-130.

Shannon, R.E. 1992. Introduction to simulation. In *Proceedings of the 1992 Winter Simulation Conference*, ed. J.J. Swain, D. Goldsman, R.C. Crain, and J.R. Wilson, ACM, 65-73.

Sherer, S.W. and G.H. Walton, 1998. Practical applications of statistical testing. In *Proceedings of the 4th Joint Avionics and Weapon Support, Software, and Simulation Conf.* (JAWS S^3), Las Vegas, Nevada, June 1998.

Smith, S.W. 1998. Essential techniques for military modeling and simulation. In *Proceedings of the 1998 Winter Simulation Conference*, ed. D.J. Medeiros, E.F. Watson, J.S. Carson, and M.S. Manivannan, ACM, 805-812.

Walton, G.H. and J.H. Poore. 2000. Generating Markov chain transition probabilities to support model-based software testing. *Software Practice and Experience*, Vol 30, August 2000, 1095-1106.

Walton, G.H., J.H. Poore, and C.J. Trammell. 1995. Statistical testing of software based on a usage model. *Software Practice and Experience*, Vol 25, No 1, January 1995, 97-108.

Whittaker, J.A. and J.H. Poore. 1993. Markov analysis of software specifications. *ACM Transactions on Software Engineering and Methodology*, Vol. 2, No 1, January 1993, 93-106.

Whittaker, J.A. and M.G. Thomason. 1994. A Markov chain model for statistical software testing. *IEEE Transactions on Software Engineering*, Vol. 30, No. 10, October 1994, 812-824.

AUTHOR BIOGRAPHIES

GWENDOLYN H. WALTON is an Associate Professor of Computer Engineering at the University of Central Florida. She received a Ph.D. in Computer Science from the University of Tennessee. Her research interests include software quality measurement and management and rigorous methods for software specification, verification, and testing. Her email is <gwalton@mail.ucf.edu>.

ROBERT M. PATTON is a doctoral student in Computer Engineering at the University of Central Florida. He received a B.S. and M.S. in Computer Engineering from the University of Central Florida. His research interests include validation and verification of simulation systems, automated testing and analysis of software intensive systems, and artificial intelligence systems. His email is <rmpatton@yahoo.com>.

DOUGLAS J. PARSONS is a Sr. Systems Engineer at the U.S. Army Simulation, Training, and Instrumentation Command. He received a M.S. in Systems Management (Operations Research) from Florida Institute of Technology, and is currently working toward a M.S. in Industrial Engineering (Interactive Simulation and Training) from the University of Central Florida. His technical interests include software systems test design, software reliability, and cognitive modeling. His email address is <doug_parsons@stricom.army.mil>.