THE APPLICATION OF STATISTICAL USAGE TESTING TO IMPROVE VALIDATION OF SIMULATION SYSTEMS

Robert M. Patton and Gwendolyn H. Walton
University of Central Florida
School of Electrical Engineering and Computer Science
Orlando, FL, U.S.A.

Douglas J. Parsons
U.S. Army Simulation, Training and Instrumentation Command
Orlando, FL, U.S.A.

Abstract

There are a variety of approaches to validation of simulation systems, each attempting to solve one or more specific issues. When viewed as a whole, these approaches can address a broad spectrum of issues. However, regardless of the validation approach, scalability and input domain explosion make it impossible to exhaustively test simulation systems. Improved methods are needed.

This paper describes a case study that demonstrates the application of statistical usage testing methods to provide quantitative support for test planning and user testing of a commercially available combat simulation game. First, typical approaches to validation of simulation systems are discussed with respect to some of the challenges of simulation system validation. Then the case study is used to provide a brief introduction to statistical usage testing and to illustrate the applicability of usage modeling and statistical usage testing to improve the efficiency and effectiveness of test planning, management, and analysis of test results. Results from the case study include suggestions for applying usage model analysis methods to iteratively improve usage models and support more efficient and effective achievement of specific test objectives.

Biographical Sketch:

Robert M. Patton is a doctoral student in Computer Engineering at the University of Central Florida. He received a M.S. in Computer Engineering from the University of Central Florida in 2001. His research and technical 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>.

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 in 1995. Her research and technical interests include software quality measurement and management and rigorous methods for software specification, verification, and testing. Her email is <gwalton@mail.ucf.edu>.

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>.

THE APPLICATION OF STATISTICAL USAGE TESTING TO IMPROVE VALIDATION OF SIMULATION SYSTEMS

Robert M. Patton and Gwendolyn H. Walton
University of Central Florida
School of Electrical Engineering and Computer Science
Orlando, FL, U.S.A.

Douglas J. Parsons
U.S. Army Simulation, Training and Instrumentation Command
Orlando, FL, U.S.A.

INTRODUCTION

The ability to effectively and efficiently validate a simulation system is essential to providing the user with a quality system and delivering within cost and schedule constraints. There are a variety of approaches to testing simulation systems, each attempting to solve one or more specific issues. When viewed as a whole, these approaches address a broad spectrum of issues. However, regardless of the testing approach, scalability and input domain explosion make it impossible to exhaustively test simulation systems, even when automated testing tools are used. Thus, in addition to the need for more testing, there is a need for improved methods to support "smarter" validation.

CASE STUDY BACKGROUND AND PURPOSE

Tiberian Sun Command & Conquer is an interactive combat simulation game produced by Westwood Studios (Westwood, 1999). The game allows the user to choose to play one of two opposing sides. The user may also choose to play pre-defined missions or to play a "free-for-all" combat scenario. Each side has similar structures, vehicles, and soldiers, but with slightly different capabilities. Users can choose to play the simulation on pre-defined terrain maps or create their own maps. A user can play against one to seven computer opponents at one time, or can play against other human opponents via the Internet.

The case study focused on validation of free-for-all combat, referred to as a "skirmish scenario" in the Tiberian Sun game. In a skirmish, each of the opposing sides sets up a base that consists of various structures, vehicles, and soldiers. Each side then attempts to attack their opponent until all players except one have been completely destroyed. The combat can be performed on various terrains and at various times of day.

The vehicles and soldiers are semi-automated, allowing the user to focus on the overall combat situation instead of attending to the details of the simulation. Only a minimum number of user commands are required to perform specific tasks. For example, to move a soldier from one location to another, the user selects the soldier by clicking one time on the soldier, and then selects the destination location by clicking one time on the location of choice. The soldier should then move to the destination without further input from the user. If there are obstacles between the origin and the destination, the soldier should automatically avoid the obstacles without input from the user. To attack a particular enemy target, the user selects a soldier or vehicle by clicking one time on the soldier or vehicle, and then selects the enemy target by clicking one time on the target. Based on the unit's weapons, the unit should automatically move as necessary to within the appropriate distance from the target, and then should open fire on the target.

The benefits of statistical usage testing for providing support for quantitative test planning, test generation, test automation, and quantitative analysis of test results have been demonstrated in numerous industry and government projects in a variety of domains (Agrawal and Whittaker, 1993; Sherer and Walton, 1998; Walton, Poore, and Trammell, 1995; Whittaker and Poore, 1993; Whittaker and Thomason, 1994). However there has been no published work on the application of these methods to validate simulation systems. The Tiberian Sun game provides an excellent small-scale example of the problems that are typically encountered when attempting to validate a military simulation system. Thus, this game is appropriate for use in a small case study to demonstrate and evaluate the applicability of statistical usage testing methods for combat simulation systems.

VALIDATION CHALLENGES

There are three major challenges that must be addressed to validate any simulation system, including the Tiberian Sun game used in the case study: scalability, size of the input and output domains, and the need to tailor the validation processes to meet individual project objectives.

In military simulations, many different groups of objects can 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 and 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 example, to exhaustively test Tiberian Sun's pre-defined missions on pre-defined terrains requires validation of behavioral and non-behavioral attributes for all combinations of pre-defined missions, pre-defined terrains, all possible structures, all possible vehicles and vehicle capabilities, all possible soldiers and soldier capabilities, and all possible opponents.

Due to time and resource constraints, typically all combinations can not be tested. As a result it can be difficult to make inferences from the test results. For example, suppose a tester chooses to use 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 revealed. To complicate matters, military simulations can cover a wide variety of terrains. 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 all the different types of terrain.

As a military simulation scales up, the input domain can grow exponentially. A variety of vehicles, troops, and structures can be combined in a variety of ways on many mission types and terrains. A variety of environmental conditions such as night, day, and type of weather 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 does not grow as quickly as the input domain. However the output domain can be partially unknown, or may have multiple possible responses for the same input data (Robinson, 1997). For example, when the simulation runs with human-in-the-loop, the outcome for a given set of input data may be different each time the simulation is run. Also, when military simulations are used to model future scenarios, weapons, or environments, real output data may not exist. A third example is when military simulations are used to simulate combat scenarios that are not well understood. The outcomes of these simulations are often a result of a human decision making process, and it can be difficult, if not impossible, to completely identify the output domain. The challenge presented by the output domain becomes evident even in a small-scale combat simulation such as the Tiberian Sun game when the game is played using the free-for-all "skirmish scenario" using human opponents whose goals, abilities, and temperaments may be unknown.

In the case where there is no pre-defined correct answer, simulations cannot be determined to be right or wrong, and validation is used to build user confidence in the simulation system. This motivates the need for tailoring validation processes to meet individual project objectives.

TYPICAL VALIDATION APPROACHES

There are a variety of typical approaches to validation of simulation systems, each attempting to address one or more specific issues, and each with strengths and weaknesses (Balci, 1994). These techniques have been categorized in different ways (Balci, 1994; Sargent, 1998). Most of the validation approaches discussed in the simulation literature are standard software engineering techniques. Only a few techniques specific to validation of simulation systems have been reported. These techniques were categorized by Sargent (1998) as subjective or objective testing approaches.

Subjective testing approaches rely heavily on expert opinions, or third part evaluation. These approaches can be valuable when there are some aspects of the simulation that cannot be performed objectively, cannot be specified in a detailed manner, or that require human-in-the loop. When using the subjective testing approach, the evaluation results can vary significantly depending on which "experts" do the evaluation.

Objective testing approaches rely less on expert opinions and more on statistical and automated methods. For military simulations, these approaches can be particularly useful for capturing and analyzing output of the simulation such as a weapon's rate of fire, the hit/miss ratio, or the speed of a vehicle over a particular terrain. Statistical approaches and knowledge bases can be used to analyze the data.

Subjective and objective testing methods can be combined. Examples include the use of a validation knowledge base and the use of expert systems developed using case-based reasoning to evaluate decisions as the simulation progressed (Birta and Ozmizrak, 1996; Hopkinson and Sepulveda, 1995). Advantages of these approaches are the ease in automation and possibility for subsequent analyses of the knowledge encapsulated in the experts' decisions. Disadvantages include the problem of experimental design, and the restriction of the validation exercise to

variations of known problems (Birta and Ozmizrak, 1996; Hopkinson and Sepulveda, 1995).

The ability to effectively and efficiently validate a simulation system is essential to providing the user with a quality system and delivering within cost and schedule constraints. When viewed as a whole, the typical validation approaches address a broad spectrum of issues of importance to simulation systems. However, the simulation literature clearly documents the need for improved techniques for effectively and efficiently validating the behavior of a simulation system, and the need for increased use of domain-specific knowledge in developing test plans and evaluating test results.

STATISTICAL USAGE TESTING

The use of Markov chain usage models and Markov chain analyses to provide quantitative support for test planning and test management was introduced by (Whittaker and Poore, 1993; Whittaker and Thomason, 1994). Information on developing and optimizing usage models has been provided by (Walton and Poore, 2000; Walton, et al., 1995). Statistical usage testing based on Markov chain usage models has been successfully used for a variety of software projects (Agrawal and Whittaker, 1993; Sherer and Walton, 1998; Walton, et al., 1995; Whittaker and Poore, 1993; Whittaker and Thomason, 1994). However, to date there have been no published reports describing the application of statistical usage testing to support validation of simulation systems.

Military simulations are stochastic in nature. Thus, testing based on Markov chain usage models should be well suited to support validation of simulation systems. One or more usage models are created to characterize populations of software use. For software testing purposes, the Markov chain models of interest are discrete time, finite-state, irreducible models.

Usage Model Development

Each component model's structure can be represented by a set of states and an associated set of directed arcs that define state transitions. The model structure is developed from the software specification and describes the capabilities of the software. Transition probabilities on the arcs represent the use of the system and drive the generation of scenarios of use from the model.

Usage model development and analysis is an iterative process that includes a number of challenges. Because the usage models can directly impact the effectiveness and efficiency of the testing process, care must be taken to make appropriate tradeoffs between the need to provide an appropriate representation of the user's point of view and the need for models that are reusable, flexible, and maintainable to support test objectives that may evolve during the simulation system's life cycle.

A very detailed model may require more maintenance effort as the simulation system and its usage evolves. However, a very high-level model may not support sufficient testing of all aspects of the use of the system. Clearly, the perfect amount of detail can be elusive.

A usage model must be simplistic enough to be understood by the customer, but detailed enough to support the testing team in evaluating test results. An effective usage model will promote dialogue between the customer, developers, and testers. This dialogue can enhance the clarity of the requirements and thus support reduced development and testing costs. Usage models are specifications that must be verified against system specifications and test objectives.

For the case study, usage models were developed to drive testing of the Tiberian Sun game using a single human player playing against a single computer opponent using a “skirmish” scenario. In the absence of detailed requirements or specifications for the game, the usage models were developed by the tester after reading the system documentation and playing the game for several hours to gain some insights into ways that a user could use the system. Some of the high-level models developed for the case study will be used to illustrate the usage model development and analysis processes.

The usage model of Figure 1 represents a very high level view of the use of the system. A user starts the game, chooses the “Skirmish” option, chooses a terrain map, chooses a side, and begins the combat simulation. Many other options are available to the user to “fine-tune” the simulation, but these choices are sufficient to illustrate the usage model development process.

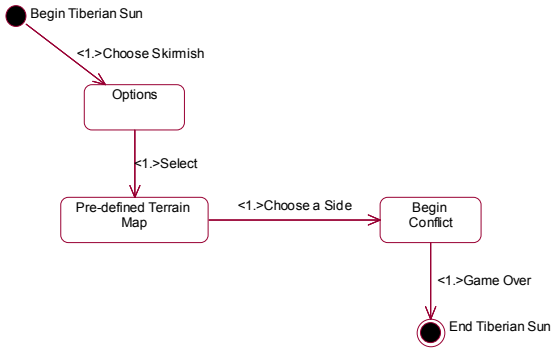

Figure 1. Example high level usage model for Tiberian Sun

The arcs are annotated as follows:

- The value within the '<' and '>' represents a probability value that the transition will be chosen by the user. In the model of Figure 1, each state has only one outgoing arc. Thus, each arc shown in Figure 1 has a probability of 1. If a state has more than one outgoing arc, then the sum of the probabilities of each arc must add to 1.
- The word or phrase following the '>' is an arc label which describes the user's input or some event which causes the transition between states.

Figure 2 extends the model of Figure 1 to include a more detailed description of the state "Begin Conflict". The sub-model within the state "Begin Conflict" describes some basic behaviors of the soldiers and vehicles within the simulation. This sub-model illustrates a lower-level abstraction of the use of the system. The outgoing arcs from each state in the model of Figure 2 have uniform probabilities, representing completely random selection of the transition to select at each state.

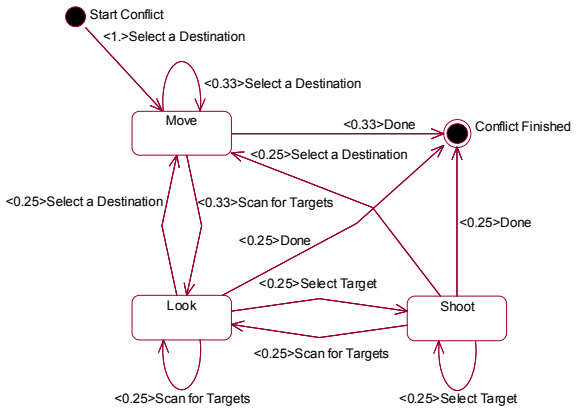

Figure 2. Example low level usage model for Tiberian Sun.

Usage Model Analysis to Support Test Planning

Markov chain mathematical analyses can provide cost-effective assistance for planning and documenting the critical aspects of the system to be tested, and can support decisions concerning how to allocate the testing resources to each type of software use.

Rather than analyze the model of Figure 1 separately from the model of Figure 2, we combine the two models, replacing the state "Begin Conflict" of Figure 1 with the model given in Figure 2. This process of expanding a high-level model by incorporating the states and arcs of lower-level models is often described as "flattening the model".

When an arc with probability 1.0 is included from the "End Tiberian Sun" state to the "Begin Tiberian Sun" to denote long-run usage, every state in the model is reachable from every other state. Thus, this model is a finite-state, irreducible Markov chain. There is a mature body of mathematical routines that can be applied to the underlying transition probability matrix of this type of Markov chains. Standard Markov chain mathematical techniques were used to calculate useful statistics to describe the flattened model. [reference Whittaker and Poore here] Partial results are given in Table 1.

State Name	Long Run Probability
Begin Tiberian Sun	0.1071
Options	0.1071
Pre-defined Terrain Map	0.1071
Start Conflict	0.1071
Conflict Finished	0.1071
Look	0.1071
Move	0.2143
Shoot	0.0357
End Tiberian Sun	0.1071

Table 1. Analysis results for Figures 1 and 2.

The long run probability for each state can be interpreted as the expected proportion of time the state appears in a large number of random test scripts generated from the model. For example, Table 1 shows that the state "Move" in the flattened model has the highest value for long run probability. This indicates that if a large number of test scripts were generated from the flattened model, the state "Move" will occur more frequently than any of the other states. The state "Shoot" will occur infrequently; it has the smallest value for long run probability.

The Markov chain analysis results and the types of sequences generated from a model are strongly influenced by the choice of values for the transition probabilities. A small perturbation to the probabilities can have a large effect on the entire model. Even in simple models, it can be difficult to predict the impacts of even small changes.

The statistics generated using uniform probabilities illustrate the testing bias that occurs as a result of the model structure (i.e., the states and arcs). Changing the arc probabilities to a non-uniform distribution will further bias the testing. Thus, it is prudent to begin the usage model development process with uniform arc probabilities on the exit arcs for each state. Examine the analytical results, then iteratively modify the

probabilities, examining the analytical results after each iteration to verify that the change accomplished the desired objectives.

For example, consider the low-level model of Figure 2. A test objective, "Collect data to verify the hit/miss ratio of the weapons of various soldiers and vehicles", would require that most of the testing time be spent in the "Shoot" state. Table 1 shows that, for test cases generated from a model with uniform probabilities, the "Shoot" state is the least likely state to occur. This implies that a very large number of test cases would have to be generated from the uniform distribution model in order to drive testing that spends a sufficient amount of time in the "Shoot" state.

To execute a large number of test cases may exceed the available testing resources. Thus, the arc probabilities must be changed to direct more testing to the "Shoot" state. One method to accomplish this is to reduce or zero the probabilities on arcs that are not important to the test objective. Figure 3 illustrates an example of such a model.

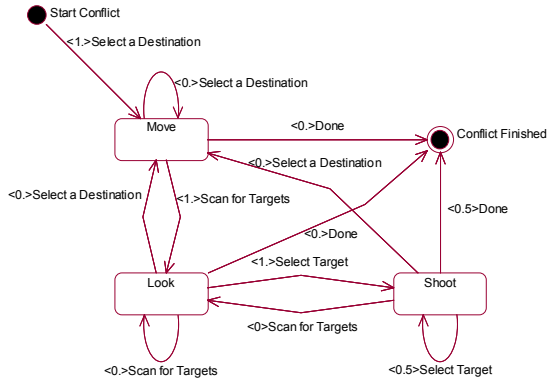

Figure 3. Modified usage model biased toward the Shoot state

When the model of Figure 3 is incorporated into the high-level model of Figure 1, a Markov chain analysis of the flattened model yields the long run probabilities of Table 2. The "Shoot" state is now the state most likely to show up in our test cases. Thus, this revised model is more appropriate for driving the generation of test sequences to support the test objective: "Collect data to verify the hit/miss ratio of the weapons of various soldiers and vehicles."

State Name	Long Run Probability
Begin Tiberian Sun	0.1000
Options	0.1000
Pre-defined Terrain Map	0.1000
Start Conflict	0.1000
Conflict Finished	0.1000
Look	0.1000
Move	0.1000
Shoot	0.2000
End Tiberian Sun	0.1000

Table 2. Analysis results for Figures 1 and 3.

Usage model structures are reusable artifacts. A single usage model structure can support a variety of sets of arc probability values to model different classes of users and classes of use of the system. This supports efficiencies in developing, verifying, and maintaining models and test plans.

Automated Generation of Test Sequences

Any number of statistically correct samples of test sequences can be automatically generated from the Markov chain models:

- For each sequence:
  - Start the test sequence at the beginning state, "Begin Tiberian Sun."
  - Automatically generate a random number between 0 and 1.
  - Select a transition from this state by considering the cumulative arc probabilities on the exit arcs of this state:
    - If the random number is less than or equal to the first arc probability, the first transition is selected.
    - Else, if the random number is less than or equal to the sum of the first and second arc probabilities, select the second transition.
    - Continue until a transition is selected.
  - The destination state of the selected transition is the next state in the sequence.
  - Generate another random number between 0 and 1 and follow the same process to select the transition from this state
  - Continue until the end state is reached, "End Tiberian Sun."
- Continue until the desired number of sequences has been generated

Figure 4 contains an example of a test sequence generated from the flattened model. The test sequence represents a scenario to be executed by the tester and describes a use of the system. More information is required in order to complete the test case. (for

example: expected results, and input data values). This information can be manually developed, or the modeling process can be enhanced to include additional annotations on the states or arcs.

Begin Tiberian Sun
1. Choose Skirmish
Options
2. Select
Pre-defined Terrain Map
3. Choose a Side
Start Conflict
4. Select a Destination
Move
5. Scan for Targets
Look
6. Select Target
Shoot
7. Select Target
Shoot
8. Select Target
Shoot
9. Done
Conflict Finished
10. Game Over
End Tiberian Sun

Figure 4. Example test case generated from the models of Figures 1 and 3.

Suppose that, during the case study to demonstrate the use of statistical usage testing techniques on the Tiberian Sun game, a new increment of the game was made available that includes a feature to allow the user to create their own terrain maps rather than use the pre-defined maps.

The previous usage models do not include this new feature. Figure 5 provides a modified version of the model of Figure 1 to reflect this new feature.

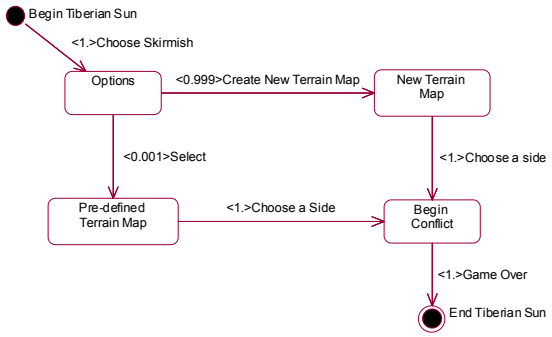

Figure 5. Modified usage model representing added features of Tiberian Sun.

Assume the test objectives for the new version of the game differ from the previous objectives. The capabilities of the game represented in the usage model by the arc from "Options" to "Pre-defined Terrain Map" and the state "Pre-defined Terrain Map" were well tested in the first increment. For the new version of the game, two test objectives are:

- Do very little, if any, re-testing of the capabilities emphasized in the previous testing
- Do focused testing on the new feature.

To support these test objectives, the arc from "Options" to "Pre-defined Terrain Map" in the high-level model is given a value very close to zero.

In the new version of the game, the user can create new terrain maps. It is important to validate that vehicles maintain appropriate mobility on each user-created terrain. Thus, a third test objective is:

- Collect data on vehicle mobility on the user-created new terrain map.

To support this test objective, the arc probabilities of the low-level model of Figure 2 are changed to focus the testing on the movement of units.

The revised model to be used to drive statistical usage testing of the new version of the game is shown in Figure 6. Note that any arcs that do not directly pertain to our test objectives have values of either zero or very close to zero.

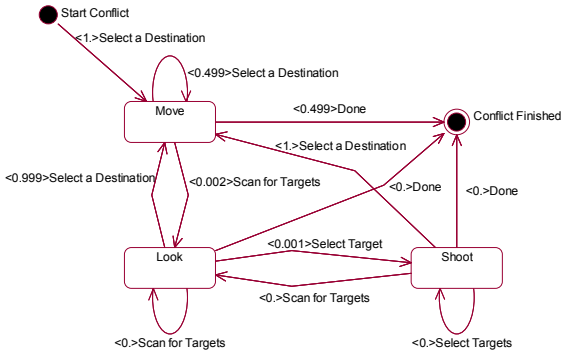

Figure 6. Modified usage model biased toward the Move state

When the model of Figure 6 is incorporated into Figure 5 and a Markov chain analysis is performed on the flattened model, Table 3 gives the long run probabilities.

State Name	Long Run Probability
Begin Tiberian Sun	0.1249
Options	0.1249
Pre-defined Terrain Map	0.0001
New Terrain Map	0.1247
Start Conflict	0.1249
Conflict Finished	0.1249
Look	0.0005
Move	0.2503
Shoot	0.0000
End Tiberian Sun	0.1249

Table 3. Analysis results for Figures 5 and 6.

The data in Table 3 indicates that the model meets the stated test objectives. The state "New Terrain Map", representing the new feature in the software, is much more likely to occur in the test cases than is the "Pre-defined Terrain Map" state. In addition, the state "Move" is expected to occur the most frequently in a set of a large number of test cases generated from this model. Figure 7 is an example test case generated from the new flattened model

```
Begin Tiberian Sun
   1. Choose Skirmish
Options
   2. Create New Terrain Map
New Terrain Map
   3. Choose a side
Start Conflict
   4. Select a Destination
Move
   5. Select a Destination
Move
   6. Select a Destination
Move
   7. Select a Destination
Move
   8. Select a Destination
Move
   9. Done
Conflict Finished
  10. Game Over
End Tiberian Sun
```

Figure 7 Example test case generated from the models of Figures 5 and 6.

Analysis of Test Sequences

To evaluate the test plan, it can be useful to know the percentage of the usage model states and arcs that will be covered if the planned test cases were to be executed without experiencing a failure. To illustrate this point, Ten test sequences were randomly generated from the new model developed for the case study. Comparing the test sequences with the usage model yielded the model coverage information given in Table 4.

The ten test cases did not result in complete coverage of the model's states and arcs. Upon examination of the data in Figure 7, it was determined that the states and arcs not covered by the test cases do not contribute toward achieving our stated test objectives.

Case	% States Covered	% Arcs Covered
1	70	46
2	70	46
3	70	54
4	70	54
5	70	54
6	70	54
7	70	54
8	70	54
9	70	54
10	70	54

Table 4. Coverage results of generated test cases

Table 4 indicates that, after test case 3, the percentage of covered states or arcs does not increase. Thus, scripts 4 through 10 re-visit states and arcs that were covered during test cases 1 through 3. This situation does not necessary imply that unnecessary testing is planned. There may be a variety of data values that can be selected each time a particular arc or state is visited but that are not explicitly described in the model. Duplicate test sequences can support repeating of a test case using different input data, thus improving the coverage of the input domain. Even if identical input values are selected on a subsequent execution of the software, the test experience may be valuable to demonstrate repeated use of the simulation system. Due to build-up of internal state data, the simulation may run very differently in the initial use from the way it runs after continued use.

Analysis of Test Results

If testing is driven by test sequences randomly generated by a usage model, and if enough of these sequences are executed to yield a test set that is statistically typical of the population of usage described by the model, quantitative analysis of test results is possible. Reliability and MTTF estimates are two of the values that can be calculated by comparing the model with the test sequences and any failures experienced during testing. [Reference Whittaker and Thomason paper here] In addition, the test sequences and the failure records can be analyzed to determine what software use has been tested, what percentage of the

testing effort was concentrated on specific uses, and what usage has not been tested. This information can be invaluable for regression test planning and can help build the user's confidence in the software, an especially critical issue for simulation systems.

Advantages

Statistical usage testing methods provide several opportunities for improving the efficiency and effectiveness of simulation system validation. A usage model is based on functional specifications, usage specifications, test objectives, constraints, and domain knowledge. Thus, all the information required to develop the model should be available before a line of code is written. This allows test planning to occur in parallel with the software development, reducing the elapsed time required for developing and delivering the product. The usage model supports modeling and quantitative analysis of software specifications, quantitative test planning and evaluation of test plans, and automatic generation of a statistically correct sample of test cases.

Test management decisions, including risk assessments, can be based on scientific data that is computed before and during the test. 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.

The usage modeling process also supports effective use of domain specific knowledge in the testing process. Domain specific knowledge can be used to help build usage models that can be used to drive testing of the software. Domain experts also support development of test objectives that provide the tester with information about potential weaknesses in the software and insight into way in which the software will be used.

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.

Costs and Cautions

There is typically a learning curve and often a need for mentoring for testers when they first begin working with statistical testing methods. The usage modeling process is labor-intensive and can require schedule adjustments during the specification phases. Developing and verifying a model that can be reused and that sufficiently covers the use of the system requires careful planning and adequate time. However, the schedule adjustments required to accommodate the modeling effort are typically offset by the value that the modeling activity brings to the specification phase.

Developing the structure of the model requires some level of domain knowledge of the system and the needs of the users. This knowledge may be difficult to obtain or understand, and may vary according to the domain expert involved in the development. However, difficulties with developing the model structure can provide a valuable early warning that the requirements and specifications are incomplete, inconsistent, or incorrect.

Usage modeling is an iterative process. It can be difficult to identify appropriate values for the usage probabilities. There may not be any available information to describe how the user will use a new system. To complicate this issue, an apparently simple change to a probability may create drastic changes in the overall usage distribution of the model as a whole. It is important to anticipate and understand the consequences of such changes. Experience and at least a basic understanding of Markov chain mathematics are needed.

Markov analyses of the test experience provides useful information, but typically cannot provide a full description. Statistical usage testing results must be interpreted correctly and within the correct context. For example, a Markov chain analysis of the model may indicate that "Forty percent of the model has been covered by the generated test cases." This is not the same as saying that "Testing is forty percent complete." Or, Markov chain analysis may indicate that the "model is entirely covered after the first twenty test cases." This should not be interpreted as "The entire system has been completely tested after twenty test cases."

Due to abstract representations of inputs and events in the usage models, Markov chain analyses cannot identify the percentage of the input and output data that is covered by a set of test cases. For example, suppose ten test cases are generated. Markov chain analysis can determine what percentage of the model is covered by those ten test cases. If multiple testers executed the ten test cases, the input selected and the output recorded by each tester may be completely different. Appropriate levels of abstraction must be considered when considering model granularity and representation of the input domain. These considerations and associated risks also provide input to analysis and interpretation of the test experience.

CONCLUSIONS

The case study demonstrated the successful application of statistical usage testing methods to provide quantitative support for test planning and user testing of a commercially available combat simulation game.

The usage modeling process helped to clarify test objectives, identify areas where the tester needed more domain knowledge, and quantify the risks associated with the limited amount of planned testing. An added bonus was that the usage models were very readable and serve as useful documentation for the test plan.
A set of random test sequences generated from usage models that were more complex than those included in this paper included some test sequences that would probably not have otherwise occurred to the tester. These test sequences do not represent usage that the tester considered "typical", but they do represent possible system behavior that should be validated.

The results of this case study demonstrated that statistical usage testing techniques are applicable to the domain of simulation systems and to military simulations, in particular. Statistical usage testing consists of well-defined, rigorous methods, which readily support automation. We expect that these methods can be used to support the test objectives for military simulation systems and to enhance validation of large, complex military simulation systems by increasing the opportunities for test automation.

REFERENCES

Agrawal, K. and Whittaker, J.A.. (1993). Experiences in Applying Statistical Testing to a Real-Time, Embedded Software System. Proceedings of the Pacific Northwest Software Quality Conference, Portland, OR, October.

Balci, O. (1994). Validation, Verification, and Testing Techniques Throughout the Life Cycle of a Simulation Study. Proc. of the 1994 Winter Simulation Conf., ACM, pp. 215-220.

Birta, L.G., and Ozmizrak, F.N. (1996). A Knowledge-Based Approach for the Validation of Simulation Models: The Foundation. ACM Trans. on Modeling and Computer Simulation, Vol. 6, No. 1, Jan. pp. 76-98.

Hopkinson, W.C., and Sepulveda, J.A. (1995). Real time validation of Man-in-the-loop simulations. Proc. of the 1995 Winter Simulation Conf., ACM, pp. 1250-1256.

Robinson, S. (1997). Simulation Model Verification and Validation: Increasing the User's Confidence. Proc. of the 1997 Winter Simulation Conf., ACM, pp. 53-59.

Sargent, R.G. (1998). Validation and Verification of Simulation Models. Proc. of the 1998 Winter Simulation Conf., ACM, pp. 121-130.

Sherer, W.S., and Walton, G.H. (1998). Practical Applications of Statistical Testing. Proc. of the 4th Joint Avionics and Weapon Support, Software, and Simulation Conf. (JAWS S3), Las Vegas, Nevada, June.

Walton, G.H., and Poore, J.H. (2000, Aug) Generating Markov Chain Transition Probabilities to Support Model-Based Software Testing. Software Practice and Experience.

Walton, G.H., Poore, J.H., and Trammell, C.J. (1995, Jan.). Statistical Testing of Software Based on a Usage Model. Software Practice and Experience.

Westwood Studios (1999). http://www.westwood.com

Whittaker, J.A., and Poore, J.H. (1993). Markov Analysis of Software Specifications. ACM Transactions on Software Engineering and Methodology, Vol. 2, No 1, pp. 93-106.

Whittaker, J.A., and Thomason, M.G. (1994). A Markov Chain Model for Statistical Software Testing. IEEE Transactions on Software Engineering, Vol. 30, No. 10, pp. 812-824.