System Testing Using Use Cases for an ER Simulation Model

Guodong Shao
Manufacturing Simulation and Modeling Group
National Institute of Standards and Technology
Gaithersburg, MD 20899-8260, U.S.A.
E-mail: gshao@nist.gov

KEYWORDS
Modeling and Simulation, System Testing, Use Case.

ABSTRACT

Modeling and simulation (M&S) techniques are increasingly being used to solve problems and aid decision making in many different fields. It is particularly useful for Department of Homeland Security (DHS) applications because of its feature of non-destructive and non-invasive method of observing a system. Results of simulations are expected to provide reliable information for decision makers, but potential errors may be introduced in the M&S development lifecycle. It is critical to make sure to build the right model and that the model is built right. System testing is an effective methodology that can help to ensure the functionality of a software system. It can also be applied to M&S applications. Use cases are usually used to specify requirements of a simulation system. The collection of use cases can cover the complete functionality of the simulation system and provide information necessary to generate test cases for system testing. Since use cases are associated with the front end of the M&S development lifecycle, testing can get started much earlier in the lifecycle, allowing simulation developers to identify and fix defects that would be very costly if found in the later stages. This paper identified the needs of system testing using specifications for M&S applications for DHS applications and providing a novel approach of Verification, Validation and Testing (VV&T) for DHS M&S community. As an example application, a hospital emergency room (ER) simulation model was introduced. Use cases for the ER model were developed. Functional system test requirements and testing criteria of the ER model were discussed. Based on the coverage criteria, activity diagrams associated with the use case are created to capture scenarios and allow the specification of use case to be tested.

INTRODUCTION

Modeling and simulation (M&S) techniques are more and more being used to model real world problems in many different applications. M&S is an effective means to shorten real system development time by answering many what-if questions first. IEEE *standard glossary of* modeling and simulation terminology (IEEE 1989) has the definitions for model and simulation as “A model is an approximation, representation, or idealization of selected aspects of the structure, behavior, operation, or other characteristics of a real world process, concept, or system.” “Simulation is a model that behaves or operates like a given system when provided a set of controlled inputs.” M&S is the process of constructing a model of a system that contains a problem and conducting experiments with the model on a computer for a specific purpose of solving the problem and aiding in decision-making. The developers and users of the simulation models, the decision makers using the results of these models, and individuals affected by decisions based on such models are all concerned with whether a model and the simulation results are correct (Sargent 2007).

M&S is particularly valuable for DHS application, because M&S provides a non-destructive and non-invasive method of observing a system and also provides a way to test multiple inputs and evaluate various outputs (Donald and Brown 2005). Simulations allow users to reconstruct a comprehensive representation of real-world features during disaster response (Lisa 2006). The limitations of live exercises can be overcome through the use of simulation models that allow emergency response personnel across multiple levels in multiple agencies to be exposed to the same scenario. Simulation models can help the decision makers determine staff and resource levels in hypothetical terrorist attack scenarios (Shao and Lee 2007). These M&S applications often introduce new risks associated with potential errors in creating the model (programming errors) and inadequate fidelity (errors in accuracy when compared to real-world results). There are no established procedures for determining whether the results obtained from an M&S application are correct or satisfy real world needs. To ensure that a valid model and a credible simulation that produce correct results exist, verification, validation and testing of the model and the resulting simulation must be employed throughout the life cycle of an M&S application. (Cook and Skinner 2005). (Balci 2007) defines the model VV&T as follows: “Model validation is substantiating that the model, within its domain of applicability, behaves with satisfactory accuracy consistent with the study objectives. Model validation deals with building the right model. It is conducted by running the model under the same input condition that drive the system and by comparing model behavior with the system behavior. The comparison of model and system behaviors should not be made one output variable at a time. Model verification is substantiating that the model is transformed from one form into another, as intended, with sufficient accuracy. Model verification deals with building the model right. The accuracy of transforming a problem formulation into a model specification or the accuracy of converting a model representation in a micro flowchart into an executable computer program is evaluated in model verification. Model testing is demonstrating that inaccuracies exist or revealing the existence of errors in the model. In model testing, we subject the model to test data or test cases to see if it functions properly. “Test failed” implies the failure of the model, not the test. Testing is conducted to perform validation and verification. Some tests are devised to evaluate the behavioral accuracy (i.e., validity) of the model, and some tests are intended to judge the accuracy of model transformation from one forming another (verification).”

IEEE Standard Computer Dictionary (IEEE 1990) defines system testing as “System testing is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic.” System testing is concerned with testing an entire system based on its specifications. It is independent of the process used to create any application. The tester evaluates the application from a user perspective. Internal design details are irrelevant and do not affect how tests are defined. The application’s behavior, whether presented as use cases or other forms of requirements, drives the development of test cases (Tamres 2002). Effective system testing requires a concrete and testable system-level specification. A system specified with use cases provides much of the information necessary for system testing…the collection of use cases is the complete functionality of the system (Booch et al. 1999) (Kaner 2002). Unified Modeling Language (UML) (Fowler 2005) (Gomma 2003) use cases are usually used to define the M&S system requirement, specification and design.

Normally, testing addresses only verification by checking if the implementation meets the specifications. System testing using use case models also assists model validation. A complete analysis of the use case models not only evaluates whether the generated tests cover the requirements, but also evaluates whether the use case description meet the intended use needs (Hasling et al. 2008).

Traditionally, test case design techniques include analyzing the functional specifications, the software paths and the boundary values. These techniques are still valid, but use case testing provides a new perspective and identifies test cases in its unique way (Collard 1999). Early in the lifecycle of a software system there is no code to execute but there are models – requirement models, analysis models, architecture models, and others (McGregor 2007). Briand and Labiche presented the testing object-oriented systems with the UML functional system test methodology. They derive test requirements from use case description, interaction diagram (sequence or collaboration) associated with each use case, and class diagram (composed of application domain classes and their contracts). This early use of analysis artifacts is very important as it helps devising a system test plan, size the system test task, and plan appropriate resources early in the life cycle (Briand and Labiche 2001).

A test case is a description of a test with the expected outcome. A set of test cases can be created based on the use case of the simulation systems to verify if the model is correctly implemented according to its requirements. The test cases are defined as instantiations of the use cases of the simulation system. An important advantage of creating test cases from specifications is that they can be produced earlier in the development lifecycle and be ready for use before any codes are developed. Additionally, when the test cases are generated early, simulation developers can often find inconsistencies and ambiguities in the requirements specification and design documents. This will definitely bring down the cost of modeling and simulation systems as errors are eliminated early during the life cycle.

This paper has described a novel method to test DHS M&S applications. Currently there is no existing procedure within DHS for VV&T of M&S applications. Many DHS M&S applications are developed by different contractors. As a user, DHS may not be familiar with all the simulation tools and associated programming techniques, but they know what they want, understand the requirements well, therefore, performing a system testing using use cases to create test cases is a very useful approach to verify the requirements. Further more, the test cases can be reused if multiple contractors develop similar M&S applications using different tools.

As a DHS M&S application, a hospital ER simulation model is introduced to apply the system testing technique that generates test cases from use case specifications. The ER simulator is a discrete event simulation model of an emergency patient’s flow in a hospital. The purpose of this simulation is to provide a small but realistic model of resources and patient’s flow and congestion in the ER of the hospital in response to an emergency incident including the deployment of resources and actions for triage and treatment of the injured, movement of casualties to hospitals, and treatment at the hospitals. Ensuring the model’s creditability is very critical. Only a correctly implemented model can provide valuable information for the hospital management teams to make the right decisions that will affect others including medical staff and patients. A system testing for the simulation model based on the UML use case model will assist to make sure the system meets the intended user needs and is implemented right.

This paper is organized as follows: next section introduces a prototype of the hospital ER simulation model. Then use cases for the ER model are discussed. An example activity model is generated based on the use cases. Test requirements and criteria for the use case and the activity diagram are discussed. Finally the test cases associated with the use case and activity model are identified.

THE ER SIMULATION MODEL

The emergency department simulator models the resources, patients flow and congestion in response to an emergency incident. The model demonstrates how the incident affects: dispatch of ambulance to transport of injured to the hospital, as well as the waiting time in different areas, and evaluates the resources needed according to different scenario. The simulation will allow hospital management teams to train by responding in real-time to crises that affect ER flow and evaluate the impact of their decisions.

The primary entities in the model are patients, medical records, and soiled linen; resources are medical staffs, and specialists, emergency vehicles, triage and exam rooms, test lab, and beds. Patients are modeled as first in – first out queues. The model allows the user to make modifications to selected model parameters through a graphical user interface. The user can change the number of patient arrival quantities and the average number of trauma and average number of cardiac patients per day. There are trauma rooms, cardiac rooms and specialty treatment rooms. Ambulatory and ambulance entrances exist as patient arrival points. The arrival of a cardiac or trauma patient, who will use more resources, will cause the backlog of regular patients (Shao and McLean 2008).

Model inputs

The inputs of the simulation model are listed as follows:
Patient's arrivals are modeled using statistical distributions.

- Number, location and type of casualties
- Availability of staff at work and off (on-call)
- Availability of resources
- Time and resources required for attending to each casualty type
- Probabilities of death from different casualty types over time.
- Hospital location
- Layout of the hospital
- Process stations
- Station capacities
- Processing times
- Patient arrivals rate
- Hospital shifts
- Medical resources
- Symptom-treatment profiles

Model outputs

The outputs of the simulation model may include the operation of the ER over time such as:

- System utilization
- Utilization of process stations and resources
- Updates of the status of the patients and medical staff
- Number of people treated and released, admitted, dead, waiting for treatment over time
- State of the staff and facilities (to determine their capability to deal with another incident)
- Run Time Interactions
- Simulated clock time – from Execution Control Supervisor
- Number of EMTs and ambulances dispatched over time to Traffic Simulation
- Number of ambulances and casualties arrivals over time from Traffic Simulation

Model logic

Figure 1 shows the model overview. There are two kinds of patients as arrival entities of the model: Ambulance and General. Ambulance patients are those patients who are in critical situation, such as trauma and cardiac patients. There are limited rooms and beds for ambulance patients. If all the rooms are occupied at the time; the patient has to be redirected to an alternate facility. After a patient is taken into the room, a Technician and Registered Nurse (RN) will treat the patient right away, create a medical record, and take the patient to the Medical Doctor (MD) for review. The MD will make a decision, and the patient will be moved to the nursing unit when the necessary procedures are done. General patients are ambulatory patients who can walk into the hospital and wait for an exam and treatment. They have to go through the triage process first. If all seats are taken, a triage-waiting area is provided. After the triage, patients are sent to the main waiting area waiting for calls to the different exam rooms based on their categories. Exam rooms include general exam, orthopedic exam, OB/Gyn exam, pediatric exam, and critical exam rooms. If it is not critical, the patient can be discharged. If further tests or X rays are needed, patients have to be in the queue for these procedures.

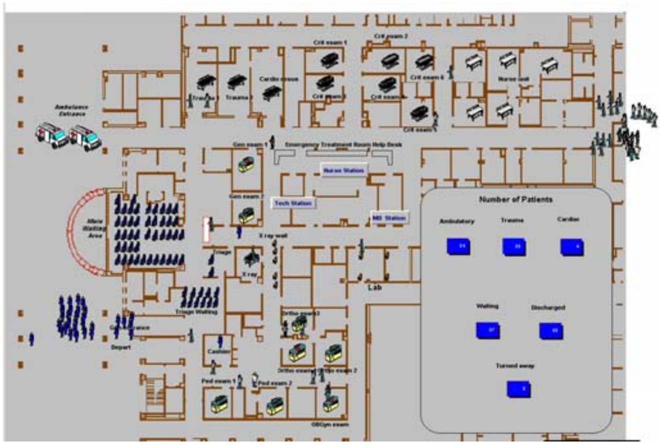

Figure 1: ER Model Overview

USE CASE

UML use cases are widely used to define the M&S application requirements. They are also used for the ER model. Use cases tell the user what to expect, the developer what to code, the technical writer what to document, and the tester what to test (McGregor 2007). They are used to describe sequences of actions that the simulation system performs as a result of input from the users; use cases help to express the workflow of the application. A use case describes interactions between users and system. This makes use cases independent from the implementation (EODiSP 2008) and reusable because they shall apply to every implementation of the system, regardless of what simulation tool is selected and the graphical user interface looks like.

Use cases represent the high level functionalities provided by the system to the user, so they are a good source for deriving system test requirements. When planning test cases for use cases, all possible execution sequences need to be identified and then covered during testing as they may be sources of different failures. But the use case diagram itself is not a very typical graph for testing; it is too high level and not many node and branches can be covered. However, a use case can be described in more detailed form as a table. The table provides details of operation and includes alternatives, which model choices or conditions during execution (Ammann and Offutt 2008).

Depicted in Figure 2 is the use case model diagram for the ER model. The ovals represent use case, and the stick figure represents actors that can be either humans or other software systems that interact with the simulation system. The lines represent communication between an actor and a use case. Each use case represents that functionality that is going to be implemented. In the context of the ER model, there are two kinds of actors (Shao and Lee 2007):

- *Simulation Analyst*: The *Simulation Analyst* is the core user of the system. The simulation analyst is responsible for executing the model and analyzing the simulation results on a daily basis. S/he might be involved in the simulation system development and is capable of performing data collection. The simulation analyst can define various scenarios for other users, verify the model based on the scenario, make suggestions regarding the length of the simulation run, the number of runs needed, and the initial conditions. S/he is responsible for analyzing the simulation results and documenting the findings.
- *Simulation User*: The *Simulation User* is the primary user of the system. By simulating different scenarios in a virtual environment using different settings, S/he is trained to respond to all kinds of situations. The response actions may include the deployment of resources, actions for triage, treatment of the injured, movement of casualties to other facilities, and transferring patients to another hospital/facility under different scenarios in the virtual world.

There are a total of 11 use cases in this use case model. The actor *Simulation Analyst* has five use cases; *define scenarios, initial/reset simulation, configure simulation environment, analyze simulation results*, and *turn-on facility layout*. The actor *Simulation User* has six use cases: *simulate patient arrival, simulate patient departure, simulate triage process, simulate emergency treatment, run simulation* and *simulate lab test and exam*.

As a sample, a detailed introduction of the *simulate patient arrival* use cases is provided in Table 1. The table will provide a basis for creating the activity diagram, which is more useful for testing.

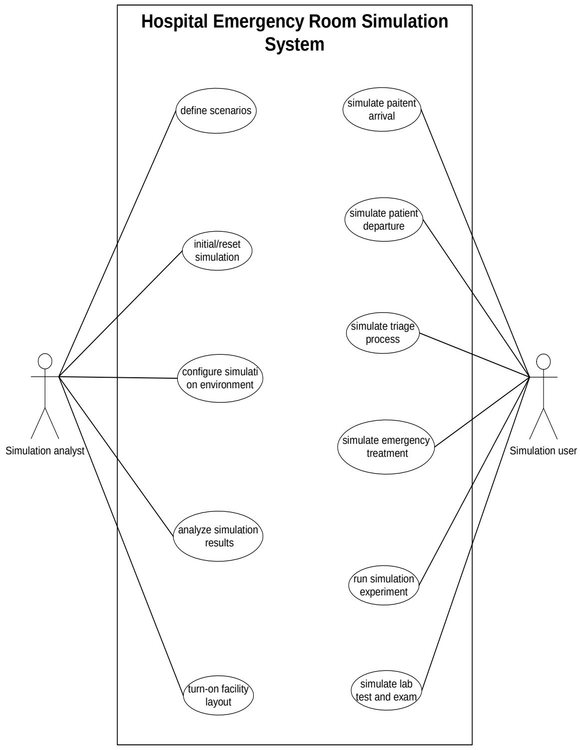

Figure 2: Use Case of ER Simulation System

Table 1: Use Case for Simulate Patient Arrival

Use Case Name	**Simulate Patient Arrival**
ID	1
Summary	Patient arrival rate and other characteristics are being entered and simulated
Actors	Simulation user
Preconditions	• Simulation software is launched • Simulation model is loaded • Simulation scenario is defined
Description	1 Simulation user starts to run the simulation model 2 Simulation system prompts user to select type of patient from a list (ambulatory patients, trauma patient, and cardiac patient) 3 Simulation user chooses the patient type 4 Simulation system prompts user to input number of patients 5 Simulation user inputs number of patients 6 Repeat step 2, 3, and 4 three times in order to enter all three kinds of patients 7 Simulation system executes with the patient type and arrival rate entered
Alternatives	If the user inputs invalid data, the simulation model will abort with an error message. Line 2, 3, 4, and 5: based on the different implementations, the user interface may vary, the way the user inputs data may be different.
Post conditions	• Patient type and quantities are entered into the system • Patient arrival rates are calculated and stored in the system • Simulation continues to run

ACTIVITY DIAGRAM

A UML activity diagram can be created based on a use case. An activity diagram shows the flow among activities. In many ways, UML activity diagrams are the object-oriented equivalent of flow charts and data flow diagram from structured development (Ambler 2004). Activities can be used to model a variety of things, including state changes, returning values, and computations. In this paper, the activity diagram is used to model the logic capture by the use cases as considering activities as user level steps. Two kinds of nodes are used: action states and sequential branches. The numeric items in the use case description presented in table 1 express steps that the actors undertake. These correspond to inputs to or outputs from the simulation model and appear as nodes in the activity diagram as action states. The alternatives in the use case represent decisions that the model or actors make and are represented as nodes in the activity diagram as sequential branches (Ammann and Offutt 2008). One activity diagram could represent several test cases because of decision points and data variations described in the activity diagrams.

The activity diagram for the "Simulate Patient Arrival" is shown in Figure 3. As described in section 2, there are three types of patients: General patients, trauma patients and cardiac patients. The user needs to input the number of patients for each type. Once all three types of patients are entered into the model, the system will check to see if the inputs are valid or not. If the input is valid, the simulation will continue to execute smoothly, otherwise, if any of the inputs is invalid, an error message will be displayed and the simulation will abort. In order to generate the test cases from the activity diagram that derived from the original use cases, we need to define the testing requirement and coverage criteria.

TESTING CRITERIA

There is no such thing as "complete testing" and "exhaustive testing." Coverage criteria are used to decide which test inputs to use and also provide useful rules for when to stop testing. The definition of test requirement and coverage criteria by (Ammann and Offut 2008) are: "Test Requirement: A test requirement is a specific element of a software artifact that a test case must satisfy or cover. Coverage Criteria: A coverage criterion is a rule or collection of rules that impose test requirements on a test set."

Ammann and Offutt introduced four distinct coverage criteria: Graphs, logical expressions, input space and syntax structures. In the use cases and activity diagram discussed in previous sections, where user language is used, there is no complicated predicate that contains multiple clauses, so logic coverage criteria is not useful. Also because there are no obvious data definition-use pairs, the data flow coverage criteria are not applicable. The two applicable criteria to use case graphs are node coverage and edge coverage. Test case values are derived from interpreting the nodes.

Another criterion for use case graphs is based on scenarios. A use case scenario is an instance of a use case, or a complete path through the use case. End users of the complete system can go down many paths as they execute the functionality specified in the use case. Multiple scenarios may be needed to completely describe a system. Following the basic flow would be one scenario. Following the basic flow plus first alternate flow would be another. The basic flow plus second alternate flow would be a third, and so on (Zielczynski 2006).

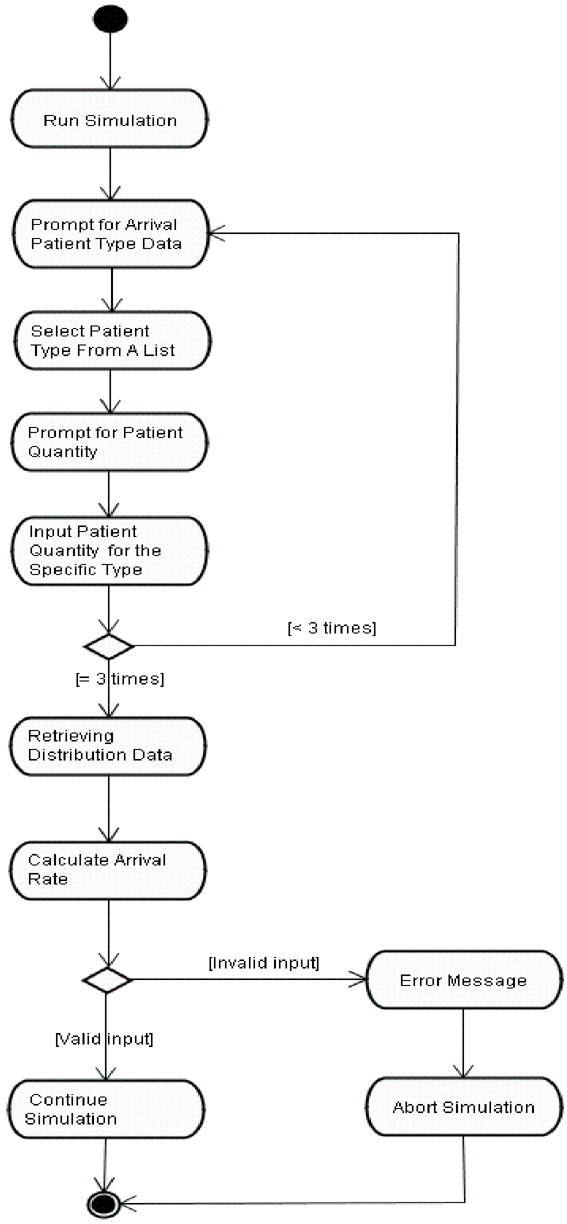

Figure 3: Activity Diagram for "Simulate Patient Arrival" Use Case

Figure 4 shows that every use case may have many scenarios; it is a one–to-many relationship. One scenario may also have many test cases, so it is also a one–to–many relationship. In this paper, we applied the scenario criteria to generate test cases for the ER model.

To create test cases from activity diagrams, every path or transition need to be considered. Test procedure for these test cases are used to verify successful and/or acceptable implementation of the simulation system requirements. This provides good traceability to original requirements, to test and verify requirements and to discover inconsistency in the requirements. Missing test cases are only a result of an incomplete use cases model (Hasling et al. 2008).

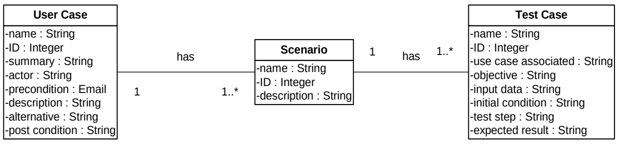

Figure 4: Relationship Diagram of Use Case, Scenario and Test Cases

TEST CASE

The purpose of a test case is to identify conditions that will be implemented in a test and expected results. Test cases are needed to verify acceptable implementation of the system requirement, which is a use case model in this paper. (Samurin 2008) defines test case as "a set of test inputs, executions, and expected results developed for a particular objective: to exercise a particular program path or verify compliance with a specific requirement."

An excellent test case should satisfy the following criteria (McGregor 2007):

- Reasonable probability of catching an error
- Exercises an area of interest
- Doesn't do unnecessary things
- Not redundant with other tests
- Makes failures obvious
- Allows isolation and identification of errors

Here is the three-step process for generating test cases from a fully detailed use case (Heumann 2001):

- For each use case, generate a full set of use case Scenarios such as a use case description table and activity diagrams.
- For each scenario, identify at least one test case (basic flow) and the conditions that will make it execute.
- For each test case, identify the data values that are used to test.

Based on the use case description, each combination of basic and alternate flows and the scenarios can be identified. Test cases can be created as soon as a use case is available, well before any code is written.

As an example, test cases for simulate patient arrival are created in Table 2 and Table 3. Table 2 presents the normal basic flow process, we need to make sure this scenario works correctly, and then we need to cover the major alternative path that the user can take through this use case and think about what could go wrong. Table 3 shows the invalid input scenario.

Data coverage for the test can also be specified. If you want to create tests with every possible data variation in every possible test path, you may end up with too many tests, impossible for you to handle. Therefore, sample the data variation choice in each test path is a practical way to do (Heumann 2001). We used the Input Domain Modeling (IDM) method discussed in (Ammann and Offutt 2008) and category - partitioning technique to decide the testing data values in the test steps. The details are not discussed in this paper.

Table 2: Test Case for "Simulate Patient Arrival" Use Case

Test Case Name	**Simulate Patient Arrival –Normal Basic flow process**
Use case name	Simulate Patient Arrival basic flow
Objective	To verify using valid patient arrival data
Input data	• Ambulatory patients: 250 • Trauma patient: 10 • Cardiac patient: 6
Initial conditions	1. The hospital ER simulation model is running. 2. Graphic user interface prompt for patient type selection
Test steps	1. Simulation user selects an unselected patient type from the list (ambulatory patients, trauma patient or cardiac patient). 2. Simulation System prompt "Enter the avg number of daily ambulatory patients (default avg =150);" "Enter the avg number of daily trauma patients (default avg=4);" or "Enter the avg number of daily trauma patients (default avg=4);" based on the patient 3. Simulation user enters a valid input (not one of the three 0, A, or 10000) 4. Repeat 1, 2, and 3 steps three times to cover all the three patient types. 5. Simulation system runs smoothly with the valid inputs entered.
Expected results	After the user input valid data, the simulation model will continue to run using the input to calculate patient arrival rate.

Table 3: Test Case for "Simulate Patient Arrival" Use Case

Test Case Name	**Simulate Patient Arrival – Enter invalid input**
Use case name	Simulate Patient Arrival alternate flow
Objective	To verify using invalid patient arrival data
Input data	• Ambulatory patients: 0 • Trauma patient: A • Cardiac patient: 200000
Initial conditions	• The hospital ER simulation model is running. • Graphic user interface prompt for patient type selection
Test steps	1. Simulation user selects an unselected patient type from the list (ambulatory patients, trauma patient, or cardiac patient). 2. Simulation System prompt "Enter the avg number of daily ambulatory patients (default avg =150);" "Enter the avg number of daily trauma patients (default avg =4);" or "Enter the avg number of daily trauma patients (default avg =4);" based on the patient types. 3. Simulation user enters an invalid input (one of the three 0, A, or 10000) 4. Repeat 1, 2, and 3 steps three times to cover all the three patient types.
Expected results	If any of the input is invalid, a error message will be displayed and simulation will abort.

CONCLUSION

M&S techniques are increasingly used to solve problems and aid decision making in many different fields, and are particularly useful for DHS applications because the actual system simulated may be impossible to be built, or has not been built yet, or testing an actual system is too dangerous or costly (Cook and Skinner 2005). Results of simulations are expected to provide reliable information for the decision makers to make wise decisions and predictions, but potential errors may be introduced in the process of the M&S development lifecycle. It is critical to build the right model and that the model is built right.

System testing is an effective methodology to help ensure the functionality of a software system. It can also apply to M&S applications. A well-defined concrete and testable system-level specification is needed for that purpose. Use cases are usually used to specify the requirements for a simulation system. The collection of use cases can cover the complete functionality of the simulation system and provide information necessary to generate test cases for system testing. Since use cases are associated with the front end of the M&S development lifecycle, testing can get started much earlier in the lifecycle, allowing simulation developers to identify and fix defects that would be very costly if found in the later stages. This also provides good traceability to original requirements, to test and verify requirements and to discover any inconsistency in requirements.

Using a use case model for test generation has been done in software development. This paper identified the importance of testing in early stages of the lifecycle of M&S, and presented the test methodology based on the UML use case diagram for DHS M&S applications. As a case study, a hospital emergency room (ER) simulation model was introduced. Use cases for the ER model were developed, and the use case description, activity diagram associated with the use case are created. Functional system test requirements and testing criteria of the ER model were discussed. We showed how activity diagrams can be used to capture scenarios and allow the specification of a use case to be tested. By executing the testing cases, we got expected results and improved the model based on the testing results. Problems such as array size and error messages have been fixed. The ER simulator is a relatively simple model; it's a good example to try out this system testing approach. This system testing approach can also be applied to more complex DHS or manufacturing simulation models.

This paper demonstrated a novel approach to test DHS M&S applications for the DHS community. Currently no procedure exists within DHS for VV&T of M&S applications. As a user, DHS may not familiar with all the simulation techniques, but understand the requirements well. Therefore, using use cases to create test cases is a very useful approach to verify the requirements. Further more, the test cases can be reused if multiple contractors develop similar M&S applications using different tools.

REFERENCES

Abdurazik, A. and J. Offutt. 2000. "Using UML Collaboration Diagrams for Static Checking and Test Generation". *In Proceedings of the Third International Conference on the Unified Modeling Language (UML '00),* York, UK, 383--395.

Ammann, P.; and J. Offutt. 2008. *Introduction to Software Testing.* Cambridge University Press, New York.

Balci, O. 2007. "Validation, Verification, and Testing Techniques throughout the Life Cycle of a Simulation Study." *Annals of Operation Research*, 53 (1994), 121-173.

Booch, G.; I. Jacobson; and J. Rumbaugh. 1999. *The Unified Modeling Language User Guide*. Addison-Wesley.

Briand, L. and Y. Labiche. 2001. "A UML-Based Approach to System Testing". *In Proceedings of the Fourth International Conference on the Unified Modeling Language (UML 2001).* (Toronto, Canada, October 2001).

Collard, R. 1999. "Test Design: Developing Test Cases from Use Cases", *Better Software Magazine*. (Vol. 1, Issue 4). 1-10.

Cook, D. and J. Skinner. 2005. "How to Perform Credible Verification, Validation, and Accreditation for Modeling and Simulation." *The Journal of Defense Software Engineering.*

Fowler, M. 2005. *UML Distilled Third Edition A Brief Guide to the Standard Object Modeling Language.* Addison-Wesley, Boston.

Gomaa, H. 2003. *Designing Concurrent, Distributed, and Real-Time Applications with UML*, Addison-Wesley, Boston.

Heumann, J. 2001. "Generating Test Cases from Use Cases." *The Rational Edge*.

Kaner, C. 2002. "Black Box Software Testing – Professional seminar".

McGregor, J. 2007. "Test Early, Test Often." *Journal of Object Technology* (vol. 6, no. 4), 7-14. http://www.jot.fm/issues/issue_2007_05/column.

Sargent, R. 2007. **"**Verification and validation of simulation models**".** In Proceedings of the 2007 Winter Simulation Conference. IEEE, Picataway, N.J., 124--137.

Shao, G. and C. McLean. 2008. "Emergency Room Simulation Prototypes for Incident Management Training". *In Proceedings of Industrial Simulation Conference 2008*, (Lyon, France, June 2008). 323-327.

Shao, G. and Y. Lee. 2007. "Applying Software Product Line Technology to Simulation Modeling of Emergency Response Facility." *Journal of Defense Modeling and Simulation, The Society for Modeling and Simulation International*, Vol. 4(4), October 2007, http://www.scs.org/pubs/jdms/vol4num4/vol4num4.html.

Tamres, L. 2002. *Introducing Software Testing*, Addison-Wesley, Boston.

Zielczynski, P. 2006. "Traceability from Use Cases to Test Cases." *The Rational Edge*.

Hasling, B.; H. Goetz; and K. Beetz. 2008. "Model Based Testing of System Requirements Using UML Use Case Models". *In Proceedings of 2008 International Conference on Software Testing, Verification, and Validation.* 367--376.

EODiSP. 2008. "Use Cases and Test Cases." http://www.pnp-software.com/eodisp/resources/archive/useAndTest.

Samurin, A. 2008. "Test case template for those who are using or would like to implement the Use Case modeling technique." http://www.geocities.com/xtremetesting/TCtemplate.html.

IEEE. 1989. "IEEE Standard Glossary of modeling and simulation Terminology." IEEE STD 610.3.

IEEE. 1990. "IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries." IEEE New York, NY.

Ambler, A. 2004. *The Object Primer: Agile Model-Driven Development with UML 2.0.* Cambridge University Press, New York.

Donald, R. and D. Brown. 2005. "Development of Metrics to Evaluate Effectiveness of Emergency Response Operations". *In Proceedings of the 10th International Command and Control Research and Technology Symposium.*

Lisa, P. 2006. "Simulation Model for Bioterronrism Preparedness in an Emergency Room". *In Proceedings of the 2006 Winter Simulation Conference.*