REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE Professional Paper	3. DATES COVERED
4. TITLE AND SUBTITLE Legacy Software Testing – A Current Methodology		5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Ralph Gibson; Michael Chapman		5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Air Warfare Center Aircraft Division 22347 Cedar Point Road, Unit #6 Patuxent River, Maryland 20670-1161		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S) 11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

In the simulation world, software upgrade is more common than new development. Thus it is extremely important to ensure proper operation of the simulation model as it is enhanced. The big question is how does one keep the current model operating correctly while adding new capabilities? The Model Development Team at the Air Combat Environment Test and Evaluation Facility, Patuxent River, Maryland, has developed an approach that is useful in making sure that the current simulation model keeps its current capabilities operating correctly as well as testing any new capability that is added. The purpose of this paper is to present the approach used by the Model Development Team to answer the question put forth. This paper discusses the software development and maintenance criteria used as the overarching guide for testing. The testing process used and how this process ensures that the model meets the criteria is then given. Specific test examples and expected output are provided as a model testing approach. Finally, the future of development testing for the Model Development Team is presented.

15. SUBJECT TERMS

Legacy Software

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON Ralph Gibson / Michael Chapman
a. REPORT	b. ABSTRACT	c. THIS PAGE Unclassified		6	19b. TELEPHONE NUMBER (include area code) (301) 342-6901 / 342-6900

20010829 187

Legacy Software Testing – A Current Methodology

Ralph D. Gibson
Science Applications International Corporation
22299 Exploration Drive Suite 200
Lexington Park, MD 20653 USA

Dr. Michael D. Chapman
U. S. Navy
5.1.6.2 Bldg 2109 Suite S115
48150 Shaw Road
Patuxent River, MD 20670 USA

Abstract. In the simulation world, software upgrade is more common than new development. Thus it is extremly important to ensure proper operation of of the simulation model as it is enhanced. The big question is how does one keep the current model operating correctly while adding new capabilities? The Model Development Team at the Air Combat Environment Test and Evaluation Facility, Patuxent River Naval Air Station, Maryland, has developed an approach that is useful in making sure that the current simulation model keeps its current capabilities operating correctly as well as testing any new capability that is added. The purpose of this paper is to present the approach used by the Model Development Team to answer the question put forth. This paper discusses the software development and maintenance criteria used as the overarching guide for testing,. The testing process used and how this process ensures that the model meets the criteria is then given. Specific test examples and expected output are provided as a model testing approach. Finally, the future of development testing for the Model Development Team is presented.

INTRODUCTION

The Air Combat Environment Test and Evaluation Facility (ACETEF) at the Patuxent River Naval Air Station provides an integrated test and evaluation environment for the United States Department of Defense, U.S. Navy and other government agencies. The backbone of the environment is the Joint Integrated Mission Model (JIMM).

JIMM is a general purpose event-driven mission level conflict simulation model. This allows the model to simulate a variety of different conflicts without the need to modify the model itself. The user directs the simulation through a series of databases that determine the types and numbers of players in the simulation, the initial conditions and movement directives for the players, and how the players will interact. In order for the outcome of certain actions to have a weighted outcome (i.e. probability of kill, damage severity, and random movements), a random number is used in some of the decisions.

JIMM was initially developed in 1999, but is based on simulations that have been in use since the late 1980s. Throughout this time, several changes have occurred to the software that have enhanced the capability and increased the reliability of the simulation. To ensure that the changes added to the program work within the context of the entire simulation model, a comprehensive test plan was devised.

This paper will present the maintenance criteria that are used to guide the testing approach and ensure model acceptance within the user community. Next, the current Acceptance Test Plan is presented with explanations given for each category of test. Examples of the testing approach and expected outcome are provided. Finally, the move to increase the automated testing and analysis of the test suites is discussed.

MAINTENANCE CRITERIA

The testing performed on JIMM is used to determine the functionality and operation of the software. The reasoning behind the tests are based on meeting the five maintenance criteria set forth in the Software Management Plan. These criteria are:

Correctness of operation. This criterion states that the operations that the program is simulating are done correctly. Above all else, this is the most important. If the simulation does not provide correct answers to the users, then the analysis that is based on the simulation may result in incorrect decisions that could cost millions of dollars or cause unnecessary deaths. For the primary users, this means that radar, sensors, line of sight calculations, vehicle movements, and other simulated events are calculated or performed correctly and are equitable to similar real-world events.

Backward compatibility. This criterion ensures that any correctly written simulation that operates using the current or earlier version of JIMM will operate using any future version. In addition, any correctly written interface will follow the same guidelines as simulations.

Repeatability. The simulation should run exactly the same independent of graphics use, listing requirements, monitor status, or shared memory use. Additionally, stopping and restarting a simulation in the middle of the run should not alter the results of the simulation.

Performance. Each subsequent software release will execute an unaltered simulation at an equal or faster rate than previous releases.

Ease of maintenance. All changes to JIMM shall adhere to high software quality practices. When additions are made to the codes and poorly written code is found in the affected areas, the writer should make changes to the poorly written section to bring it up to standards.

The criteria are presented in operational order, in that the most important criteria, correctness, takes precedence over the other criteria and so forth. Failure to meet these criteria can delay the release of the current version until it can pass.

TESTING APPROACH

The purpose of the Acceptance Test Plan is to ensure that JIMM maintains the capability to execute correctly written scenarios as outlined in the software standards. The test plan is used to determine that old capabilities remain functional and new capabilities are correctly implemented and do not interfere or alter the operation of old capabilities.

During the course of development, some or all of the tests contained in the test plan may be run to provide immediate feedback about the progress of the integration and highlight any potential problems that may occur because of the code changes. The results of these test runs are not formally recorded. Any problems that are discovered during these runs are reported to the appropriate developer so they can provide corrections to their code.

A formal acceptance test occurs at the end of the current development cycle. The configuration manager, in conjunction with the software development manager and the model development manager, initiates the formal testing process by freezing the development version and passing a copy of the code to the test manager. At this time, no further additions may made to the development model. If, during the testing process, an error or incorrect implementation is discovered, the error is corrected and passed to the configuration manager. The configuration manager makes the correction to the model, freezes the model under a different configuration name, and passes it to the test manager. The testing process is then repeated from the beginning.

Installation Test. The installation test is to ensure that the software compiles and links without warnings using native and open compiler libraries. Because ACETEF uses different platforms and operating system, this test is performed for each of the different platform/operating system combinations.

Execution Test. This test ensures backward compatibility with previously written simulations. The model must execute without any core dumps or serious warnings. Any changes to the output must be explainable and correct. This test is performed using several different simulations.

First, a generic test scenario, Obruty Final Battle, is run. This scenario has two purposes, to test as many capabilities as possible in one simulation, and to provide examples for other users to see how to use the additional capabilities. The Obruty Final Battle test is performed in two phases; first, the simulation is watched using the graphics monitor to observe the operation of the simulation, and second, the scenario is run through as series of runs with a different random number seed. The graphics test enables the tester to observe the operation of the scenario, check for graphic anomalies that would not be apparent during non-graphic operations, and to test different graphics commands that are provided.

Second, an ACETEF scenario library containing over 15 previous scenarios designed using previous versions of the simulation are run using the test version of the program. The selected scenarios must run without modifications and produce equivalent results without errors. If there are changes, they must be explainable and correct.

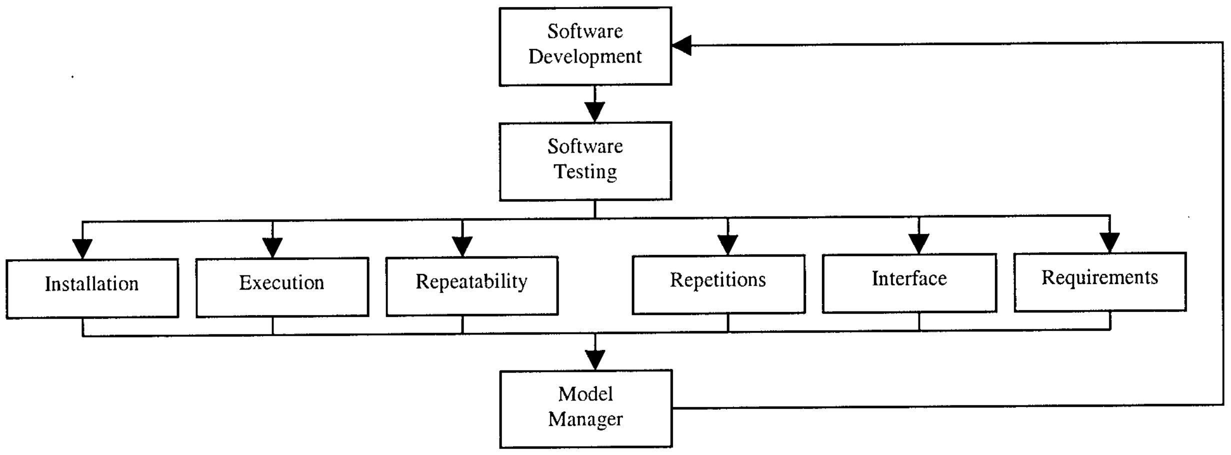

Figure 1: Overview of Acceptance Testing Process

Repeatability Test. Unless stated in the design document, modifications to the program, which address capabilities not used by a user in his scenario, should not affect the outcome of the simulation. This particular test is used extensively during integration to provide initial feedback on the various changes and how they might affect the program overall.

Repetitions Test. JIMM utilises random numbers during the simulation to affect the outcome (e.g. probability of kill, sensing chances, resource allocation). A series of runs using a different random number seed for each run allows more in-depth testing of the software. During the series of runs, one particular seed could alter the flow through the software and find the one path that causes a failure that might not have been found otherwise. As the model matures under successive interaction of the test cycle, the more common errors are weeded out. Thus, the number of repetitions needed to expose defects increase with each test cycle. This can cause schedule problems because of the time required to perform this test.

Interface Test. Aircraft system testing at ACETEF involves several different internal and external laboratories and facilities. The JIMM software is used as the engine for these tests. For reasons of cost or availability, full-scale integration tests between JIMM and the test facilities are not possible. Instead, a series of test surrogates are used to ensure correct operation. Each surrogate is specifically designed to exercise the information-passing algorithms used by a given facility.

Requirements Test. The requirements testing portion of the acceptance test plan is, by far, the largest series tests. Requirements testing is composed of a series of small test vignettes designed to test various aspects of JIMM functionality. Each vignette is designed to test a specific capability.

All current tests in the acceptance test plan are run against the development version to ensure that any modifications or additions to the software do not break currently operating capabilities. In addition, the current tests check for correct operation after errors in the code are corrected.

When developers add new capability to JIMM, they are responsible for designing a test or test series that ensures that the new capability operates correctly. The tests are added to the test plan so they can be used for testing future versions to ensure the capability still operates after changes are made to the new version. The test manager is responsible for determining that a developer's test series is sufficiently comprehensive.

The requirements tests are currently divided into eight different categories. Table 1 gives a list of the different categories and the current number of tests in each category.

Requirement Tests	Number of Tests
Orientation	27
Kinematics	43
Engagement	7
Resource Allocation	10
Nonlethal Engagement	45
Database Error Checks	9
Terrain	24
Features	64

Table 1: Number of Requirements Tests in Acceptance Test Plan

Orientation. The orientation vignettes provide a visual verification of entity pointing that is independent of movement. This series of tests were originally designed to showcase orientation bugs present in earlier versions of the program and were intended to "break" the model in as many ways as possible to ensure changes to the model fixed the problems. They have been retained in the test suite to ensure any future change to the software does not introduce any new problems.

Kinematics. The kinematics vignettes are designed to ensure that moving entities move in a way that makes physical sense. This is particularly important for three-dimensional movement. Vehicle behaviour should mirror those that we see; cars should turn flat, helicopters and missiles should move from the surface in a vertical manner, aircraft should have the correct bank angle when in a turn, etc. The various tests in this series are based upon the language available to the users to direct the different movement options.

Engagement. The engagement tests are used to test the various aspects of lethal engagement between entities in the simulation. When one player is trying to "kill" another, certain actions occur before, during and after the action. The engagement tests are used to ensure all actions taken by the attacking player are correct (in the correct position to attack), the intermediate signs of engagement appear (missile heat and smoke plume are present), and that the attacked player is affected correctly (does the target get hit, and, if so, does it live or die).

Resource allocation. This is, by far, the largest group of test vignettes in the test plan. The resource allocation series of tests are used to test the various aspects of the player "thinker" logic. The model contains a set of tactical criteria that is used to provide the player with logical steps to follow to determine a course of action. The actions can be to send a message to another player, move to a new location, provide intelligence information, or engage a target. The series of tests are designed to look at all possible combinations of options available within each tactical criterion to ensure the code handles each combination properly.

Non-lethal engagement. Not all engagements between two players are intended for one to end up dead. One player may want to degrade another player's ability to communicate or use radar to find other players. Non-lethal engagement vignettes test the ability of one player to disrupt communications and sensing, and that the signal strength is correct for the type, orientation, and position of the players.

Database error checks. When users write the databases that contain the information for the scenario, errors are occasionally introduced that cause the simulation to fail. Required information may be completely missing, or parts are accidentally left out. When this occurs, hopefully the simulation will exit gracefully with an error message printed that will tell the user what is needed to correct the error. Unfortunately, with legacy code of this complexity, such problems are generally not discovered until the end user makes such a mistake and the simulation ends abruptly without warning. When this occurs, the cause is investigated and, if it is a software problem, corrected in the code. If the problem is database related, the developer designs a model message to inform the user of the problem. This series of vignettes is the result of the various database errors that have been discovered. Each test in this series has specific errors in the appropriate database that will cause the program to terminate and print the correct message.

Terrain. Not all scenarios require terrain as part of the simulation. When they do, the user would like the players to operate correctly. The terrain vignettes test a variety of functions. Foremost in the series is the ability of the model to connect separate terrain blocks into one complete grid. Other tests in the series check movement across the terrain surface, crashing into the surface, terrain following and terrain avoidance, movement onto and off of the terrain grid, and line of sight terrain masking.

Features. This is the "catch all" category. When a particular vignette cannot be classified in one of the other categories, it is placed in this area. Vignettes in this category test such things as incidental damage (what beside the target is damaged when the target is hit), radar cross section changes when the player is looked at from different angles, and radar and visual signature changes when the player performs an action (increased radar size when bomb bay doors open or higher heat signature when the player lights its afterburner).

HOW HAS THIS HELPED?

With the advent of the comprehensive Acceptance Test Plan, the Model Development Team has been able to improve the software while ensuring that current capability has not been lost. The ATP has been performed on all major versions of JIMM, prior to their approval for use at ACETEF. Since the test plan requires that no failures occur, this has resulted in a delay in the release of each version. This delay has ranged from a couple of weeks to several months, in one extreme case. However, the ATP has detected many defects in the code, which otherwise would have affected operational use. These defects fall into three major categories: 1) unanticipated interactions between new and existing code; 2) errors in implementation of new capabilities; and 3) long hidden defects.

Unanticipated Interactions. These defects usually arise from the unexpected effects of a code modification. JIMM is legacy code, and was not very well structured, initially. It is not unusual for modifications in one section of the code to adversely impact other unrelated, or distantly related, sections of the code. By detecting these errors in test, we prevent key capabilities required by the facility from being lost. This insures robustness in the code, and instills confidence in the users in the viability of a new version.

Errors in Implementation. Errors in the implementation of new capabilities arise either due to development error in design or implementation, or integration error when incorporating a developer's code into the baseline. Detecting these errors in test insures that new capabilities will function properly during operational use, and again instills confidence in the user that using these new capabilities poses no significant risk.

Long Hidden Defects. Long hidden defects are usually errors in the code that manifest themselves spontaneously. By their very nature, these defects are random in occurrence. Their occurance can be due to the legacy nature of the code. With the continual restructuring of the code and other changes that result in altering the random number stream used for object interaction, a change in the simulation progress that can cause the code to venture down a seldom used path and with different values. Thus, they have an equal likelihood of occurring in test and operational use. However, the consequences of these defects are far less severe for test. During operational use, these defects can result in schedule slippage for a project, whereas finding them in test will only delay the release of a new version.

The increased testing has proven beneficial to the customer. Customer confidence in the software has increased in two ways; improved reliability and higher confidence in functionality of the software.

Improved Reliability. During the period prior to the development of the Acceptance Test Plan (ATP), the user community reported the majority of the software errors. During this period, the software code contained some artifacts from its previous iterations. As the code was rewritten into C++, some of the old coding style would not be compatible. Unfortunately, most of the code changes would not be immediately known because the initial unit testing would not use the changed code. It would be during the repetitions run, with its changing random number, that would eventually execute the changed code. Initially, the specific errors would occur during five of one hundred runs. With the continued use of the ATP, the number of errors has dropped to 2 per two thousand runs. Additionally, errors introduced during code modifications and enhancements, these changes appearing to operate correctly, have been discovered and corrected as a result of the ATP.

Higher Confidence in Functionality. The original JIMM documentation provided numerous instances of usable but untested capability. The user could not be sure if the capability worked at all, or if it did work, if it would perform as expected. As part of the continued growth of the ATP, many of untested capabilities are added to the test matrices, or have separate test vignettes written specifically written to test that capability. When the code does not operate correctly, error reports are written to initiate the process to correct the error. If the capability does work as documented, then the annotation that says that it is untested is removed.

THE FUTURE: AUTOMATION

As the scale of testing progressed, it became increasingly more difficult to run a full acceptance test in a reasonable length of time. To overcome this problem, the Automated Acceptance Test Plan (AATP) was created. The AATP consists of a series of UNIX C-Shell scripts that automatically process and run the test vignettes. Additional shell scripts, and a C Language analysis program, then analyze the test output. The results of the analysis are reported as Pass or Fail, usually with a failure code number that indicates the reason for failure. Automation has reduced the workload from approximately one and half weeks of manual testing to one to two hours of examining the test reports from an overnight automated run.

The automated tests fall into three basic formats. The first, and simplest, does a straight comparison of the JIMM output files from the test version with a standard expected output file. This standard file is the output file of a previous run, which has been manually analyzed for correct behaviour. Any deviation from the standard is flagged as a failure. In the event of failure, the changes would have to be analysed manually to determine if the deviation was the proper result of a modification to the code. If so, a new standard is created, otherwise the test version failed.

The second type of test uses JIMM's own analysis tools to high effect. JIMM allows the user to define a situation as one or more incidents within the program, indicating that certain actions or events have occurred. By careful design, it is possible to create a test vignette such that a situation will occur only if a given JIMM function is operating correctly. The AATP can then run the vignette, and the scripts will check for the appropriate situation. If the expected number of situations does not occur, the AATP fails the vignette. In this case, no further manually examination is necessary and the test version fails.

The final form of test is the most complex. It starts with the definition of JIMM situations, as above. In this case, however, the mere fact that a situation occurs is not sufficient evidence that the JIMM functionality under test is operating correctly. Additional analysis must be performed on the auxiliary data of the situation (timing, distance, P(k), etc.). The JIMM output is manipulated by a script into a data stream, which is then used as input for a C Language analysis program. The program then checks the data for the expected patterns. If these patterns are not found, no further manually examination is necessary and the test version fails.

CONCLUSION

The Acceptance Test Plan used by ACETEF is an attempt to introduce stability into the JIMM code, protect current functionality, ensure backward compatibility is maintained, and test the correctness of any additional JIMM capability.

The Acceptance Test Plan will continue to grow. At present, only about 60% of the current functionality of JIMM is being tested. As more capability is added, the developers must write tests to check their work. This will help prevent the test plan from falling farther behind. Efforts are underway to add more tests that check existing capability. The eventual goal is to have all functionality tested.

Without the development of the test plan, code modifications could have drastically altered the functionality of the simulation and made the model unusable. Continual testing is vital to the continued growth of the model, and the successful simulation of the scenarios.

REFERENCES

Joint Interim Mission Model User's Guide, Volume I, Version 1.2, 1999

Acceptance Test Plan (ATP) For SWEG, briefing given to the Simulated Warfare Environment Generator Users Group, May 18, 1999 by Dr. Michael D. Chapman.

Software Management Plan, Version 2.2. ACETEF Model Development Team, Patuxent River NAS, MD. April 27, 1999.

Mission Level Model Acceptance Test Plan for ACETEF, Version 1.5. ACETEF Warfare Simulation Team, Patuxent River NAS, MD. February 15, 2000.

BIOGRAPHY

Ralph Gibson is currently a Systems Analyst for Science Applications International Corporation working at Air Combat Environment Test and Evaluation Facility, Naval Air Warfare Center – Aircraft Division, Patuxent River Naval Air Station, Maryland. He is also the Configuration Management Manager for the Model Development Team. Ralph Gibson is tasked with maintaining the baseline version and incorporating all changes the Model Development Team develops for the mission model. The mission model is used by the facility for system integration test and evaluation for Navy aviation. Ralph Gibson is a graduate of the United States Air Force Academy with a degree in Astronautical Engineering. He is currently working on a Master of Software Engineering degree at the University of Maryland.

Michael Chapman works as a Computer Scientist, and Test Lead, for the Model Development Team at Air Combat Environment Test and Evaluation Facility, Naval Air Warfare Center – Aircraft Division, Patuxent River Naval Air Station, Maryland. He received a BS in physics from Renssalaer Polytechnic Institute in 1984, and a PhD in High Energy Particle Physics from the College of William & Mary in 1992.