2005

Analysis of Perturbation-based Testing Methodology as applied to a Real-Time Control System Problem

Richard A. Stutler
Virginia Commonwealth University

School of Engineering
Virginia Commonwealth University

This is to certify that the thesis prepared by Richard A. Stutler entitled Analysis of Perturbation-based Testing Methodology as applied to a Real–Time Control System Problem has been approved by his committee as satisfactory completion of the thesis requirement for the degree of Master of Science in Computer Science.

Dr. Branson W. Murrill, Associate Professor of Computer Science, School of Engineering

---

Dr. Ju Wang, Assistant Professor of Computer Science

Dr. Karla Mossi, Assistant Professor of Mechanical Engineering

---

Dr. David Primeaux, Interim Chairman, Department of Computer Science

---

Dr. Robert J. Mattauch. Dean and Commonwealth Professor of Engineering

---

Dr. F. Douglas Boudinot, Dean of the School of Graduate Studies

Date: _____

Analysis of Perturbation-based Testing Methodology as applied to a Real–Time Control System Problem

A Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science at Virginia Commonwealth University.

by

RICHARD A. STUTLER
B.S., Electrical Engineering, Virginia Polytechnic Institute & State University, 1975
M.S., Computer Science Virginia Commonwealth University, 2005

Director: Dr. Branson W. Murrill
Associate Professor of Computer Science, School of Engineering

Virginia Commonwealth University
Richmond, Virginia
May 2005

Acknowledgements

I would first like to thank my wife Sally for her support and understanding while I tied up the computer running tests day and night. I would also like to thank those software developers and testers in the Tomahawk program who provided me insight into the current algorithms and testing processes. In particular, Mike Kelly was instrumental in repeatedly setting up my tests on the Tomahawk system and helping to interpret the results. Finally I would like to thank my advisor Dr. Murrill for his help and suggestions with this research.

Table of Contents

Page

List of Tables

Page

List of Figures

Page

Abstract

Analysis of Perturbation-based Testing Methodology
as applied to a Real–Time Control System Problem

By Richard A. Stutler, Master of Science

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at Virginia Commonwealth University.

Virginia Commonwealth University, 2004.

Major Director: Dr. Branson W. Murrill
Associate Professor of Computer Science, School of Engineering

Perturbation analysis is a software analysis technique used to study the tail function of a program by inserting an error into an executing program using data state mutation. The impact of this induced error on the output is then measured. This methodology can be used to evaluate the effectiveness of a given test set and in fact can be used as a means to derive a test set which provides coverage for a given program. Previous research has shown that there is a "coupling effect" such that test sets that identify simple errors will also identify more complex errors. Thus the research would indicate that this methodology would facilitate the generation of test sets that would detect a wide range of possible faults. This research applies a perturbation analysis technique to the Cell Pre-selection algorithm as used in the Tomahawk Weapons Control System.

CHAPTER 1 Introduction

Software testing is an expensive process. Surveys have shown that as much as 32% of the projected budget and 45% of the project time have been typically allocated to testing [GAL04]. Nevertheless, it remains the primary method through which confidence in software quality is achieved. Therefore, automation of the testing process is desirable to both reduce development costs and time, and to improve confidence in the software developed. First, it should be noted that complete testing is impossible for several reasons: [Kan00]

- We can't test all of the inputs to the program.
- We can't test all of the combinations of inputs to the program.
- We can't test all of the paths through the program.
- We can't test for all of the other potential failures, such as those caused by user interface design errors or incomplete requirements analyses.

Therefore it is necessary to define what level of testing is "good enough" and to generate a method of determining when this goal has been achieved.

1.1 Testing Strategies

The first goal in software testing is usually to find test inputs that exercise specific program features. After these inputs have been defined, it is desired to quantify how well the series of test inputs actually tests the piece of code. The goal is to uncover as many

potential faults as possible with the test set. Therefore a set of tests that has the potential to uncover many faults is better than a set that can only uncover a few. Unfortunately, it is almost impossible to say quantitatively how many potential faults are uncovered by a given test set. The variety of the types of faults themselves makes this determination particularly challenging. This difficulty has lead to the development of a number of alternative test adequacy criteria, which allows the tester to distinguish good test sets from bad ones. Once an adequacy criterion has been established it is necessary to discover a set of "good" tests that satisfy the criterion. Finding these inputs by hand is extremely time consuming, especially when the software is complex. Therefore it is once again desirable to attempt to automate this process. Constraint-based testing coupled with a fault based testing methodology is a novel way of generating test data to detect specific types of common programming faults [DO93]. In addition, an algorithm for a tool that applies a combinatorial design approach to the selection of` candidate test cases was presented in [YA00].

Several test adequacy criterion have been proposed. The most straightforward method would be to test the program for all possible input cases to see if it produces the correct outputs. Unfortunately for anything other than a trivial exercise, it is impractical to test all cases [HUA75]. Since the number of potential inputs is very large, the tester has to rely on a relatively small sub-sample of the available inputs for testing. These inputs are selected to satisfy some test criterion. A classification of test adequacy criteria can be performed using the underlying testing approach. Employing this strategy, the approaches to software testing can be separated into three basic categories [ZHM97]:

(1) *structural testing:* specifies testing requirements in terms of the coverage of a particular set of elements in the structure of the program or the specification;
(2) *fault-based testing:* focuses on detecting faults or defects in the software.
(3) *error-based testing:* requires test cases to check the program on certain error-prone points according to knowledge about how the program could be expected to depart from the specifications.

1.2. Structural Testing

Structural testing is based on a control flow graph and is usually implemented by employing some form of statement, branch or path coverage criteria.

Statement coverage requires that a test set be developed such that every statement in the program is executed at least once. With this criterion, the percentage of executed statements is calculated to indicate how adequately the testing has been performed. Statement coverage is usually considered a minimal basic requirement for adequate testing. It should be noted that statement coverage is such a weak criteria, it does not even cover all conditional branches and could therefore miss many potential errors. For this reason an alternate criterion involving branch coverage is usually applied.

The *branch coverage* criterion requires that test data be selected that will cause the execution of every possible branch at each decision at least once. The percentage of the branches executed during testing is a measurement of test adequacy. It should be noted that for well-formed single-entry/single-exit structured code, branch coverage subsumes statement coverage. In other words, a test set that satisfies the branch coverage criterion must also satisfy statement coverage. Even if all branches are exercised, this

does not mean that all combinations of branches are checked. If the computations in one branch are dependent on the computations in a previous branch then that relationship has not necessarily been tested. Therefore, there exists a stronger requirement for checking all combinations of branches, called path coverage.

The *path coverage* criterion requires that all the execution paths from the program's entry to its exit be executed during testing. Since there can be an infinite number of different paths in a program with loops, it is usually necessary to define path coverage in terms of the length of the path covered. Using this criterion a test set would be developed to exercise all paths of length i.

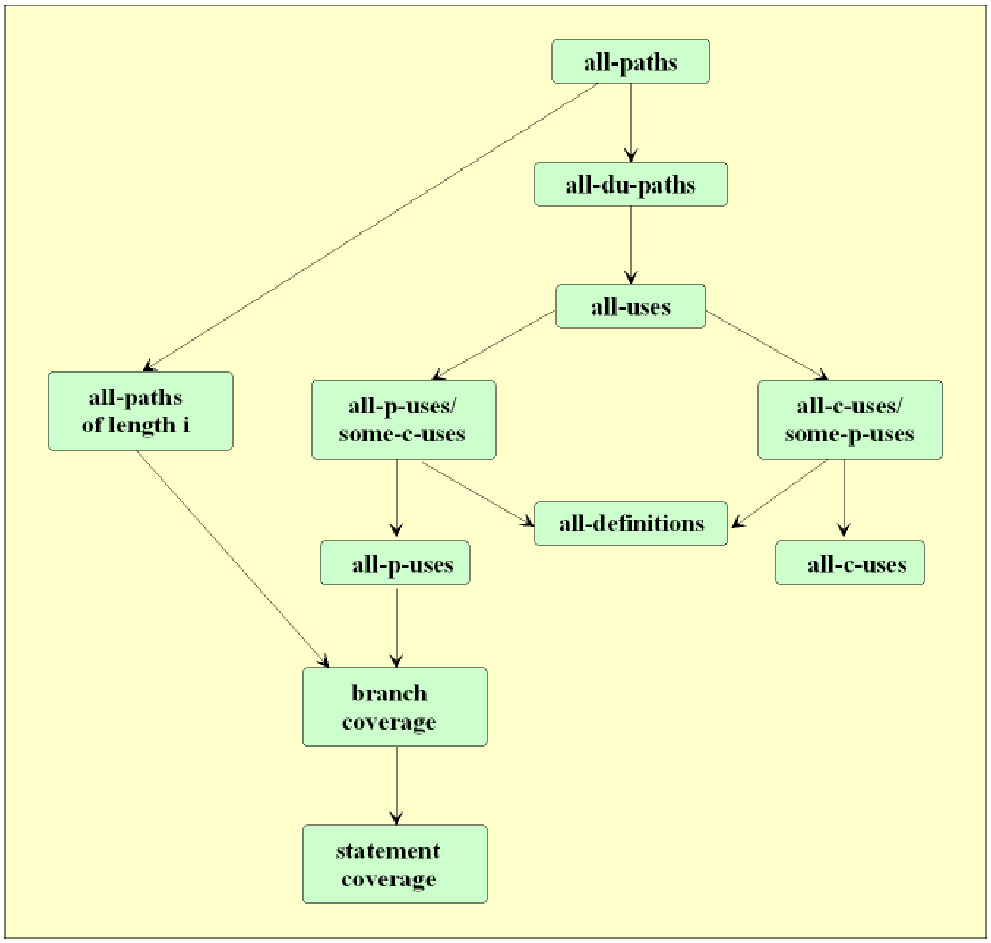
The preceding examples illustrate testing methods based on program control flow. Other structural testing methods involve deriving test cases that exercise data flow relationships. These methods determine definition and use (def-use) cases for each program variable, and derive test cases, which exercise these relationships. The def-use cases are actually characterized as a c-use if the variable is used in a computation and as a p-use if the variable is used in a predicate.

Figure 1 illustrates a hierarchy for all of the structural testing criteria discussed here. In this figure stronger criterion at the top of the chart subsume those below them. The relationships are indicated with an arrow pointing to those criteria subsumed by the more robust criteria.

Unfortunately, structural testing methodologies may not detect *computation errors*, which occur when the correct path through the program is taken, but incorrect output is generated. Structural testing may also miss errors due to semantic issues.

Therefore even if all of a program's paths are tested, the path also has to be executed with data that is going to reveal a fault. Otherwise there is no way to ensure the program is doing what is intended. These limitations motivate the need for a different approach to testing.

Figure 1. Test Criteria Hierarchy



1.3. Fault-Based Testing

As was mentioned above, fault-based testing focuses on detecting faults in the software. This usually involves injecting intentional faults into the program and then exercising the faulty program with a set of tests to determine how well those tests detect the errors. A fault-based adequacy criterion measures the adequacy of a test set according to its effectiveness or ability to detect faults. The primary purpose of this type of testing is to detect computational errors, or those errors that occur when the correct path through the program is taken, but the output is incorrect because of faults in the computations along that path. [Zei89] One of the other major benefits of fault injection is its ability to test rare events and conditions, which have been shown to be the cause of the majority of failures within critical systems. [TX02]

There are three necessary and sufficient conditions for a fault to cause a failure, described by the *fault-failure* model [MuMo94]:

- First, the fault must be executed. In other words, the test set has to exercise the portion of code that actually contains the fault.
- Second, the execution of the fault must result in the introduction of an incorrect value into the subsequent data state, called an *infection.* There are many possible instances of faults that do not result in incorrect data for a particular input.
- Third, the infection in the data state must *propagate* along the execution path and be observed as erroneous output. The incorrect data, which is generated as a result of the fault, has to result in an incorrect answer in the output.

Error flow analysis is the study of how errors infect and propagate in software. The primary methods used to perform error flow analysis include fault analysis, mutation

analysis, and perturbation analysis [MMO02]. Both of the last two methods are based on dynamic error flow analysis, where faults are artificially injected into the program with the output of the program evaluated to detect the error. The primary difference is that mutation testing injects a fault using syntactic mutation while perturbation testing injects a fault by changing the results of an operation to a different value of the same data type.

Mutation testing is based on injecting a syntactic fault into a program by applying a mutation operator and determining whether testing identifies this fault. If the test case distinguishes between the mutated program and the original program it is said to kill the mutant. Mutation operators are generally defined to produce syntactically correct variants of individual instructions in the original program. The testing process involves mutating the original program, recompiling the mutated version and comparing the output for a given set of tests or by actually generating additional tests to kill mutants.

Mutation testing has been researched quite thoroughly and has been shown to be effective in producing high-quality test sets [Car93] [MaWo92] [MR01] [OfPa97] [OL94] [ORZ93] [Wood90]. However, while mutation testing is a powerful technique, a number of problems have limited its practical impact. One of these problems is that while a mutant is syntactically different from the initial program, it may have the same behavior as the original program. These mutants that do not change the results of the program are known as *equivalent* mutants. Since they exhibit this equivalent behavior, it is essential to identify them and remove them from further analysis. Unfortunately, this is often very difficult. Another type of mutant that makes detection of equivalent mutants even more difficult is “stubborn” or difficult to kill mutants. These types of mutants are

non-equivalent but there are a restricted number of test cases that can be used to kill them. Some approaches to detecting and eliminating equivalent mutants have been introduced [OfPa97] but this remains a difficult task.

Another problem with mutation testing is that the operators are language-dependent, and therefore mutation operations may change if the program is re-written in a different language. Also, since the mutant operators modify the program text, the tail function of the mutated code is not necessarily the same as the tail function of the original code. Finally, since each mutation requires the original program to be recompiled, the mutation function can be relatively expensive to implement.

It should also be noted that for the most general class of programs, mutation and the three data flow-based criteria (def-use) are incomparable. In other words, except for a very simple class of programs, mutation does not subsume the c-use, p-use, or all-uses criterion [MaWo92].

1.4. Perturbation Analysis

Perturbation analysis is a technique used to study the tail function of a program by inserting an error into the data state and determining the impact of this induced error on the output. The intention is to modify program states created by the original code, without actually mutating existing code statements. This is often achieved through the use of inserted instrumentation code. These code instrumentations are typically applied to programmer-defined variables. They can either change the value of a variable to a value based upon the current value, or can change the variable to a value picked at random.

Ideally, any mutable bit should be available to the perturbation function and the perturbation function should be able to be called at any point in the execution of a program. However, for the purposes of this experiment, only two types of variables are handled by a perturbation function that is invoked after an instruction computes an operation. For integers, the perturbation function returns a random value, which includes all possible integers while boolean expressions are perturbed by simply inverting the value from true to false and vice versa.

Since the purpose of perturbation analysis is to analyze tail functions, it is important not to change the tail function by the perturbation. Therefore, the perturbation function is only executed once, usually the first time the function is encountered. It would also be possible to perturb the data at some other specified iteration to investigate the tail function from that point.

Perturbation methodology can be used not only to evaluate the effectiveness of a given test set it can also be used to derive a test set which provides adequacy coverage for a given program. In this instance, the adequacy coverage of a test set would be defined as a percentage of perturbations detected by the test set. In addition, perturbation analysis can be used to evaluate or compare different test strategies and to perform an analysis of the error flow properties of paths.

When used to derive a test set, perturbation analysis suffers from the same problem with equivalencies as mutation testing. An argument can be made that this effort can be skipped if each location has a fairly high kill rate; however, the random nature of perturbation also allows the use of sampling to estimate the proportion of

surviving perturbations that are equivalent perturbations. This ability to utilize sampling is not available with mutation [MMO02].

In some sense, mutation can be categorized as a subset of general perturbation. Using this categorization, it can be seen that a mutation operator is a syntactic specification of a perturbation function [MMR97]. Viewing mutation operators as specific perturbation functions suggests that general mutation effects could be achieved using perturbation. In effect, this would achieve similar results at less cost because the recompilation of mutants is not necessary. In fact, all the mutation operators can be simulated by perturbation if the perturbation function is required to produce the values that the mutant operators at a location would have produced [MMO02].

Previous research has shown that there is a "coupling effect" of perturbation testing such that test sets that will reveal simple errors will also reveal more complex errors [Oli97]. Thus the research would indicate that a perturbation analysis methodology would facilitate the generation of test sets that would detect a wide range of possible faults.

This paper evaluates the test set generated by perturbation analysis for an existing algorithm in a ship-deployed real-time control system and compares it to the test set used to validate the program performance in the actual environment. Chapter 2 will present a brief outline of the Tomahawk missile allocation algorithm being tested. Chapter 3 will provide a description of the perturbation analysis tool that was developed to facilitate this study. In Chapter 4 the results of the testing are presented including the results of automatically generating a perturbation adequate test set for the JAVA prototype and

using that test set in the actual ship-deployed code. It also presents analysis to further characterize the generated test sets. Finally Chapters 5 and 6 summarize the results and present some avenues for future work.

CHAPTER 2 Cell Pre-selection Algorithm

The United States naval strike warfare capability is continually evolving and improving. One aspect of this improvement is the development of the Tactical Tomahawk cruise missile and weapon control system capabilities. The Tactical Tomahawk Weapons Control System (TTWCS) provides information management, engagement planning, and launch control for the Tactical Tomahawk missile. It is used on the launch platform to construct and execute Tomahawk missions against tactical targets. The tasking usually originates from off the ship and is forwarded when necessary to prosecute a specific target or set of targets. One of the many challenges associated with this process is determining how to allocate missiles to the various missions. In the past, missiles were assigned to plans manually, which required a significant amount of time and experience on the part of the user. The Tomahawk missile allocation for a given strike warfare task

Figure 2. Tomahawk Missile Launch

must account for several requirements and capabilities [FJT95] [MF03]. For example, on board U.S. submarines the allocation must account for:

- Various priorities of the tasks to which Tomahawk missiles are required to be allocated
- The ability to launch Tomahawk missiles vertically out of capsules and horizontally out of torpedo tubes
- The ease/difficulty in moving a Tomahawk missile from its stowage location in the torpedo room to the torpedo tube from which it is to be launched
- The Missile Mission Matching Expanded Missile Identification lists indicating the appropriate Tomahawk variants for each task (the lists are ordered by the cost effectiveness of the variant to perform the task and can be modified by the operator)
- The missile faults, the missile maintenance due dates, and the number and length of time each Tactical Tomahawk missile has been powered up
- The ability to include/exclude Reload missiles in the automatic missile-to-task allocation processing
- Need to perform Tomahawk missile-to-task allocation to facilitate the ability of the TTWCS to allocate missiles to subsequent tasking

as well as accounting for several constraints:

- The number of Tomahawk missiles required to perform each task
- A Tomahawk missile can only be assigned to perform a single task.
- A Tomahawk missile can only be assigned to perform a task that it is capable of performing. Different missiles have different capabilities and the task allocation function must keep track of each missile's capability and status
- At any time, only one Tomahawk missile per torpedo tube can be prepared for horizontal launch from the torpedo tube.

As the number of missiles and missions assigned to a platform has increased, it has become more difficult to perform this function. The Cell Pre-selection component of TTWCS automates the missile allocation process by optimizing a set of weighted criteria relative to a set of mandatory constraints. The result is an optimal or near optimal allocation of missiles to strike plans.

In order to investigate possible approaches to performing the U.S. submarine Cell Pre-selection function, a prototype was written in JAVA. This prototype consists of about 6500 source lines of code in more than 50 class modules. For experimental flexibility, it is designed to run on a standard Windows computing platform using unclassified data sets. Unfortunately, for our purposes, the submarine tactical code is not a direct port of the prototype. This may decrease the applicability of tests developed for the prototype when applied to the actual submarine tactical code. Nevertheless, since both implementations do in fact perform the same allocation function the results for a given input should be the same. Therefore, it is hoped that lessons learned in testing the prototype could be applied to testing the actual code.

The JAVA prototype is designed to input missile loadouts and potential tasking in the form of an input test file and to use that input to determine an optimal allocation of missiles for that tasking. This primarily algorithmic nature of the program along with the relative ease to generate alternate test inputs was another attribute that facilitated perturbation testing.

CHAPTER 3 Perturbation Tool Description

Performing perturbation analysis for a program is actually a multi-step process. First the code has to be instrumented to perform the actual perturbations as needed. Then the perturbations are individually exercised to determine their effects on the program execution. In order to facilitate these processes, a tool was developed. This tool actually consists of several components. The first part of the tool supports the instrumentation and performs the actual perturbation functions. This element of the tool is embedded within the program under test and is described more fully in section 3.2. The second portion of the tool performs the perturbation analysis, generates potential test sets, and supervises the calling of the instrumented code. This analysis tool is described in section 3.3. As outlined above, the first step is the instrumentation of the actual code.

3.1. Instrumentation of the Code

The first step in the perturbation analysis is to instrument the program under test. This rather involved step includes manually adding instrumentation calls for each boolean and integer value in the program. For each perturbation instance (both boolean and integer) the user needs to define a unique variable name to be associated with this perturbation. It is then necessary to add a call to the `PERT.Var` function including the

variable name and the current or non-perturbed value. For instance, if the program included the following expression:

```
if ( x < 0 ) …
```

The instrumentation for this line of code could be as follows:

```
if ( PERT.Var(“var1”, ( x < 0 ) ) …
```

Where “`var1`” is the name of this specific boolean perturbation variable and `(x < 0)` is the value the variable would be if it were not perturbed. Of course if x is an integer, it would result in a nested set of perturbation instrumentation calls as follows:

```
if ( PERT.Var(“var1”, ( PERT.Var(“var2”,x)<0) ) ) …
```

This statement therefore results in two perturbation variables. The first “`var1`” is a boolean perturbation while the second “`var2`” is an integer perturbation variable with the current value of `x`. This nesting of perturbations often results in a complex and difficult to read source file. For example, the following is an actual before-and-after instrumentation example for a portion of the Cell Pre-selection code:

ORIGINAL CODE:

```
if(Allocations.containsAllocationFor(nextAlloc.getMissile())||
      Allocations.getConflicts(nextAlloc).size() > 0)
```

CODE AFTER INSTRUMENTATION:

```
if( PERT.Var("backTrackDeep3",
      ( PERT.Var("backTrackDeep8",
Allocations.containsAllocationFor(nextAlloc.getMissile()))||
         ( PERT.Var("backTrackDeep9",
                PERT.Var("backTrackDeep10",
                  Allocations.getConflicts(nextAlloc).size()) > 0) )
                              ) ) )
```

The **PERT** class is embedded within the program under test and is compiled and linked with it. It basically consists of two routines that can be called by the program under test. The first routine reads in the **PERT.DAT** file to determine which perturbation

is to be exercised and the value to be used for an integer perturbation. This initialization needs to be called before any other instrumentation functions are used. The other routine performs the actual perturbation by providing a static function called by the program under test. This function is `PERT.Var` and supports perturbation of either an integer or a boolean value. The `PERT.Var` function takes as arguments the variable name and the current or non-perturbed valued of the variable. If the intent is to perturb a specific integer variable, the function returns a random integer between the previously specified max and min values (note: this integer value is actually supplied in the PERT.DAT file). If the variable is not intended to be perturbed then the function returns the current or unperturbed value of the variable. The `PERT.Var` function works for boolean values in a similar fashion returning the inverse of the current value if the variable is to be perturbed.

Parameters are passed to the **PERT** tool via the specific data file **PERT.DAT**. This use of an intermediary data file allows the original program to be started without modifying its normal command line parameters. The **PERT.DAT** file consists of one line and includes the name of the variable to be perturbed and the random value for an integer perturbation. The **PERT.DAT** file is of the following form:

```
PERT_LOC:  Name_of_Variable_to_Perturb    Random_Int_Value
```

3.2. Perturbations Definition File

The perturbation tool uses a setup file to initialize the perturbation variable list and to establish the parameters for the integer perturbations. The file consists of individual lines for each perturbation variable. The line starts with **VAR_NAME:** followed by a string referring to a unique variable name. After the name are two integer fields which specify the minimum and maximum values for integer perturbation (these are set to 0 for boolean perturbations). The final field is a flag used to indicate whether the variable should be included in the perturbation analysis functions. This flag was used to account for non-accessible perturbation variables. Rather than totally removing these variables from the instrumented file, it was felt that the flag provided a more effective means of handling these while still maintaining the visibility of the identified perturbations.

This file is of the format:

```
VAR_NAME:   Perturbation_       Integer Limits   Pert?      // comments
               Variable_Name        Lo    Hi    (1=Y/0=N)
```

Where the perturbation variable name is the name as referenced in the instrumented source file, Lo and Hi are minimum and maximum limits on integer perturbations, and “Pert?” is used as a flag to optionally ignore a perturbation variable in the table. Note that comments can also be included. Appendix A includes the perturbation definition file used to instrument the JAVA prototype of Cell Pre-selection.

3.3. Analysis Tool

A tool was developed in order to generate and analyze the perturbation-adequate test sets. The tool was written in the JAVA program language to perform the perturbation analysis on a previously instrumented JAVA program. It lets the user generate a perturbation-adequate test set from nothing or add to and thereby enhance an existing test set. Alternately, the tool can be used to analyze an existing test set or a portion of an existing test set. It can also be used to verify the perturbation names in the instrumented file. It is invoked as follows:

java pertTool parameterFile

where parameterFile is the file that contains all of the setup parameters

The following parameters are specified in the setup file:

TEST_DIRECTORY:	Directory to save or find test sets
NUMBER_TRIES:	Number of attempts to kill a perturbation with randomly generated test inputs before giving up
LOG_FILE:	Name of log file
EVALUATE_ONLY?:	Y to evaluate a given test set, or N to generate (or enhance) a test set
PERT_TO_TEST:	If it is desired to look at individual perturbation variables, include a list of each variable name on a separate line (up to 50), otherwise use a '*' to indicate the desire to address all entries in the perturbations file.
NUMBER_PERTS:	Number of perturbations of each variable to generate during perturbation analysis
REPORT_TITLE:	Title of Log File Report
PERT_DIRECTORY:	Directory to save PERT.DAT
LIST_OF_PERTS:	File where list of perturbation variables and parameters is specified as outlined in Section 3.1
TEST_SET_STRT:	For evaluation of a test set, start with this test (0 is default)
TEST_SET_STOP:	… and stop with this test (-1 to test all)

VERIFY_PERTS?: Y to verify the perturbation variable names in the instrumented file while testing (default is N)

NUMBER_KILLS: Number of times a specific perturbation must be killed before being considered a "real" kill. (default is 1)

EVALUATE_TESTS: Y to do a complete evaluation of the test set.

If the user is generating or enhancing a test set, the tool first looks at the tests in the specified test directory. These tests are saved as individual files of the form; TF.txt.00xxxx, where xxxx indicates the test number for each individual test. For each perturbation variable in the perturbation file, the tool first generates the results of running a test against the original version of the program and then it generates the results using a perturbed version. If the results are different, the perturbation is indicated as killed and the program looks at the next variable in the perturbation file. If the results are the same, the program continues to compare the results of all of the existing tests. If none of the existing tests "kill" the perturbation, the program proceeds to generate random tests in an attempt to find a test that kills the perturbation. The program will generate "NUMBER_TRIES" tests before giving up and declaring the perturbation as a survivor.

If the intent is to use the tool to analyze an existing test set, the tool simply attempts to kill a perturbation of each perturbation variable using the existing set of tests. It repeats this a number of times (with a different random perturbation) as specified by the "NUMBER_PERTS" input parameter. It then computes the percentage of perturbations that were killed in terms of all perturbations tested. In addition to evaluating an entire test set, the tool also allows the user to evaluate part of the test set by specifying the range of tests to be evaluated (TEST_SET_START and

TEST_SET_STOP). In addition, by setting the EVALUATE_TESTS flag to 'Y' the user can develop a specific report which checks all of the tests in the test set against all of the perturbations. As opposed to the normal operation, this analysis does not stop when a perturbation is first detected as killed but continues to test a perturbation against all of the tests to more fully characterize the behavior of the included tests in the test set.

Another capability of the tool is the ability to verify the perturbation names. This was included to ensure incorrect perturbation names were not included in the instrumentation file. Basically, by setting this flag, each perturbation name as it is encountered will be checked against the list of correct perturbation names. If a difference is encountered, the program displays an error message. This flag should only be set when new perturbations are introduced into the instrumented program since it causes the program to run slower while verifying the perturbation names.

Testing discovered that the JAVA prototype has a non-deterministic feature attributable to the ordering of the collections that is being performed by JAVA (for further information see Chapter 4). Since there are essentially unordered collections in the program, the iterator function may not always return equivalent elements in the same order. Therefore, the same test set does not always generate the same results. These results would be detected by the comparison algorithm as different and therefore could result in a potential equivalent perturbation being killed. A final capability that was added to the tool was to adjust the number of times a perturbation would be killed before it was actually considered a "real" kill. This forces the program to detect a perturbation as being killed several times before actually announcing it as killed. The algorithm runs a

perturbation against a specific test multiple times and compares the output to the unperturbed output. The perturbation is considered killed only if it is killed for all of the test iterations, if it survives once it is considered not killed.

CHAPTER 4 Experimental Approach and Results

Testing and experimenting with actual Tomahawk code is difficult at best. Most of the code is written to run on dedicated shipboard computers and testing requires lab access. In addition, much of the data is classified which prohibits testing the code outside of a secure environment. Since the JAVA prototype was constructed to run on other than military equipment and uses an unclassified input data file to exercise its functionality; the availability of this prototype was critical to the testing process.

The approach towards applying perturbation analysis for the Tomahawk program was as follows:

(1) Instrument the JAVA prototype to allow for perturbations of the data states and identify and eliminate equivalent perturbations from those under analysis.

(2) Perform a perturbation analysis on the JAVA prototype version of the system to develop a perturbation-adequate test set for this program. The perturbation-adequacy criteria is defined to be a percentage of perturbations that are killed in terms of all perturbations tested. For the purposes of this study an adequacy criteria of 99% was considered appropriate.

(3) Perform a perturbation analysis on the existing test sets used to validate the JAVA prototype and the actual shipboard version to assess the perturbation adequacy of those tests.

(4) Use the path analysis tool in the Tomahawk lab to assess the path coverage potential of the derived perturbation-adequate test set.

(5) Finally, play the derived perturbation-adequate test set against the actual system in the lab to determine if it detects known or unknown errors.

One of the difficulties in using the perturbation method for this program is that providing the same test set does not always generate the same results. For example, below is illustrated the results of running the original JAVA prototype twice with test number 20 in Test Set 6. The differences in the allocations are highlighted in bold.

INITIAL RESULTS WERE AS FOLLOWS -----
Performed 7 allocations
Allocation[task=13,launcher=CLS9,missile=8,manual=false,chosen=false]
Allocation[task=21,launcher=CLS6,missile=5,manual=false,chosen=false]
Allocation[task=47,launcher=CLS12,missile=11,manual=false,chosen=false]
Allocation[task=62,launcher=**CLS5**,missile=**4**,manual=false,chosen=false]
Allocation[task=67,launcher=CLS15,missile=14,manual=false,chosen=false]
Allocation[task=69,launcher=CLS7,missile=6,manual=false,chosen=false]
Allocation[task=74,launcher=**CLS13**,missile=**12**,manual=false,chosen=false]

SECONDARY RESULTS WERE AS FOLLOWS -----
Performed 7 allocations
Allocation[task=13,launcher=CLS9,missile=8,manual=false,chosen=false]
Allocation[task=21,launcher=CLS6,missile=5,manual=false,chosen=false]
Allocation[task=47,launcher=CLS12,missile=11,manual=false,chosen=false]
Allocation[task=62,launcher=**CLS13**,missile=**12**,manual=false,chosen=false]
Allocation[task=67,launcher=CLS15,missile=14,manual=false,chosen=false]
Allocation[task=69,launcher=CLS7,missile=6,manual=false,chosen=false]
Allocation[task=74,launcher=**CLS5**,missile=**4**,manual=false,chosen=false]

As can be seen the program produced slightly different allocations. These differences can be attributed to the ordering of the collections that is being performed by JAVA. Since there are essentially unordered collections in the program, the iterator function may not always return equivalent elements in the same order. Therefore, although both versions are correct, they would be detected by the comparison algorithm as different and therefore could result in a potential equivalent perturbation being killed.

Detecting whether the difference in the perturbation result is due to a difference in the tail function or due to this unordered collection would require a much more complicated comparison function. Therefore, this effect was initially ignored for this study although further analysis was conducted to evaluate the magnitude of the effect on the results.

4.1. Analyzing the JAVA Prototype

The first step was to instrument the JAVA prototype and then perform perturbation analysis to derive a perturbation-adequate test set. Instrumentation of the prototype was relatively straightforward, however it did result in some code that was difficult to read and therefore verify. Since it is extremely important not to introduce additional errors in the act of inserting the instrumentation calls, it is important that the instrumentation be easy to verify. However, compound instrumentations can make the code very confusing. For this reason, it is recommended that a tool be developed for future studies that would automatically insert the instrumentation data calls. This tool would also facilitate the expanded use of the perturbation analysis methodology.

After the instrumentation calls were inserted into the code, the perturbation analysis tool was initiated to develop a set of tests that would kill all of the perturbations of the perturbation variables. This turned out to be a very time consuming process. It was found that about 30% of the initially defined perturbation variables could not be killed with more than 50000 randomly generated test sets. In order to determine why perturbation of these variables could not be killed, an in-depth analysis of the code was required. This included going back to the original author of the JAVA prototype to

determine what was keeping the perturbations from generating output errors for the random test set inputs.

As it turns out, many of these were due to the prototype nature of the program. Since the program was built to test the algorithm, it did not necessarily include extensive error checking for the inputs. It also included redundant if statements to check for situations in the input data which could not exist. In fact, when the original programmer was presented with the data regarding which perturbation variables failed to be killed, he was excited about the possibility of using this tool as a way to identify and eliminate needless conditional checks in the code. For the purposes of this study, it was decided to declare those perturbation variables that could not be killed as non-accessible variables and not to include them in further analysis. It should be noted that the PERT tool permits extra perturbation variables to remain in the file (see Appendix A) by allowing a flag to be set to keep them from being included in the analysis.

As outlined in Chapter 2, the JAVA prototype consists of about 6500 source lines of code in more than 50 classes. Perturbation instrumentation was limited to 38 processes in 12 of the classes. As shown in Appendix A, this resulted in 108 actual perturbation variables. The processes were selected because they were involved in the actual missile allocation functions. Other classes and processes that perform tasks such as input and formatting of missile tasking or error handling were not included in this analysis since the prototype version of these functions would not be consistent with the actual ship-board implementation. It should also be noted that the computational time

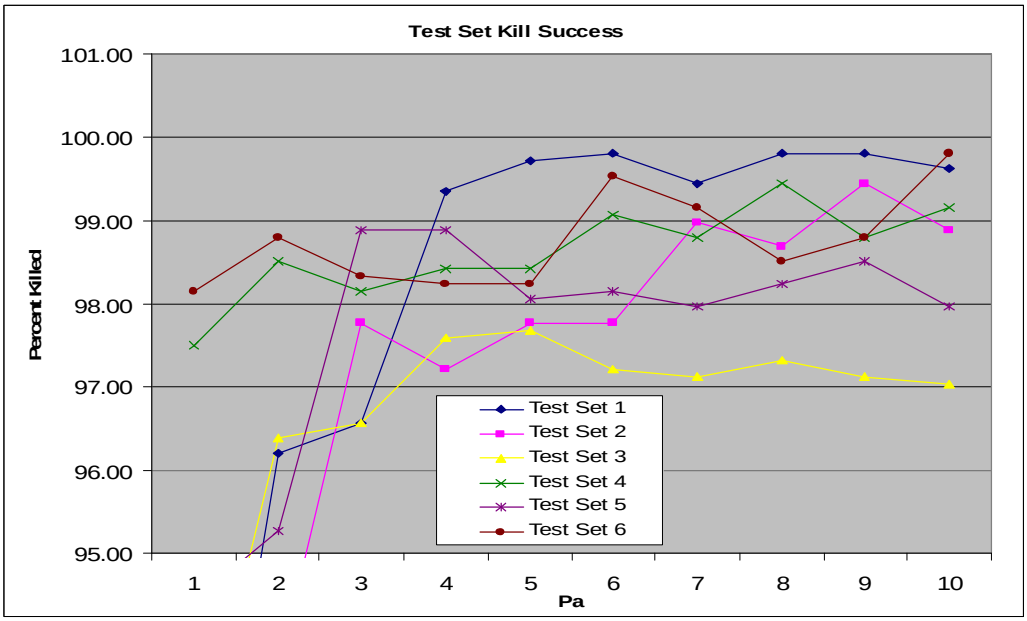
required to generate the perturbation-adequate test sets and to perform the analysis of given tests set is discussed later.

After the non-accessible variables were removed from consideration, the tool was used to develop a perturbation adequate test set for all of the accessible variables. The process used was to derive an initial test set and then use the tool to analyze the test set with the "NUMBER_PERTS" variable set to 10. This means that each analysis generates 10 perturbations per perturbation variable. Then the tool was used to augment the test set and perform the same analysis. This iterative process (referred to as a *build*) was repeated 10 times, each time adding tests to the test set. This entire process was performed six times to generate six different test sets as shown in Table 1.

Table 1. Test Set Analysis

	Test Set 1		Test Set 2		Test Set 3		Test Set 4		Test Set 5		Test Set 6	
Pa	TS Size	Percent Killed	TS Size	Percent Killed	TS Size	Percent Killed	TS Size	Percent Killed	TS Size	Percent Killed	TS Size	Percent Killed
1	17	90.19	22	92.31	29	92.22	20	97.50	23	94.44	21	98.15
2	24	96.20	25	93.61	33	96.39	22	98.52	26	95.28	23	98.80
3	25	96.57	29	97.78	36	96.57	22	98.15	27	98.89	23	98.33
4	26	99.35	30	97.22	36	97.59	22	98.43	27	98.89	24	98.24
5	27	99.72	30	97.78	37	97.69	24	98.43	27	98.06	26	98.24
6	28	99.81	33	97.78	37	97.22	26	99.07	28	98.15	27	99.54
7	29	99.44	34	98.98	39	97.13	26	98.80	28	97.96	28	99.17
8	29	99.81	34	98.70	40	97.31	26	99.44	28	98.24	28	98.52
9	30	99.81	34	99.44	40	97.13	27	98.80	29	98.52	29	98.80
10	30	99.63	34	98.89	40	97.04	27	99.17	30	97.96	30	99.81

Figure 3. Test Set Comparison



As can be seen, all six times the process resulted in a test set that grew for each iteration of the analysis. For the purposes of this experiment it was decided to use a test set which provided at least 99% coverage for the defined accessible perturbation variables. As can be seen in Figure 3; Test Sets 1, 2, 4, and 6 seemed to generate this level of coverage at least once during their multiple build cycles.

Further analysis was performed to determine which of these test sets provided optimum coverage with a minimal set of tests. First, all of the tests sets were evaluated by performing an extended analysis of the test sets using 100 perturbations for each perturbation variable. The results of this analysis are outlined in Table 2.

Table 2. Analysis of 100 Perturbations

100 Perturbations		
Test Set	Test Set Size	Average Kill Percentage
1	30	99.79%
2	34	98.87%
4	27	98.55%
6	30	99.81%

Table 3. Surviving Perturbations for Test Set 1

Test Set 1 Individual Survival Percentage	
Perturbation	TOTAL Number Times Survived
backTrackDeep2	1%
compareFaults10	1%
comparePrimaryMaint1	2%
compareRedundMaint1	6%
getNonTempAllocStack1	1%
main3	3%
Pair1	2%
TaskReader1	2%
TaskReader3	4%
TaskReader4	1%

Table 4. Surviving Perturbations for Test Set 6

Test Set 6 Individual Survival Percentage	
Perturbation	TOTAL Number Times Survived
backTrackDeep2	1%
compareFaults10	1%
comparePrimaryMaint1	5%
compareRedundMaint1	6%
main3	1%
Pair1	3%
TASKcompareTo6	1%
TaskReader1	2%
TaskReader4	1%

As can be seen from Table 2, Test Sets 2 and 4 did not exhibit 99 percent coverage over the 100-perturbation interval. For this reason further analysis was only performed on Test Sets 1 and 6.

This further analysis consisted of investigating the survival rates for each of the individual perturbation variables. Tables 3 and 4 show the results of this analysis. It should be noted that those variables not shown in Tables 3 and 4 never survived. As can be seen, both test sets provide adequate coverage for all of the perturbations. In fact none of the individual perturbations exhibit a survival rate greater

than 6% and most are in the 1-2% range. The behavior of some of the variables was particularly interesting. For instance compareRedundMaint1 survived 6% of the time for both test sets. Further analysis for this particular perturbation variable was therefore performed.

Table 5. Detailed Perturbation Analysis for Test Set 1

	Number Times Survived									
Perturbation Variable build # =>	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**	**P10**
ALLOCcompareTo1	1	1								
ALLOCcompareTo9	4	1								
backTrackDeep2	1	1	1	1		1	1			
compareFaults10	1									
compareFaults11	1		1							
compareFaults6	5									
compareFaults7	3	6								
compareFaults8	4									
compareFaults9	4	3								
comparePrimaryMaint1	5	6	4		1					
compareRedundMaint1	3	5	5	1	1		1		1	
createCapAllocList4	5									
getNonTempAllocStack1	2		3	1				1		
LOCaccessor5	3	4	6	1			1		1	
main3	7	3	2							1
MRprocess3	8	2	4							
Pair1	6	3	4	1			1			1
TASKcompareTo1	4							1		
TASKcompareTo5							1			
TASKcompareTo6	6									
TASKcompareTo10	7									
TASKcompareTo11	4									
TaskReader1	6	2	2	1			1			
TaskReader3	7	3	2	1		1				1
TaskReader4	2									
TaskReader5	7	1	3		1					1
Total	106	41	37	7	3	2	6	2	2	4

Before performing detailed analysis on the compareRedundMaint1 perturbation variable, additional test results for test sets 1 and 6 were evaluated. Table 5 presents the detailed description for the perturbation builds for test set 1. Once again, perturbation variables that are not shown, never survived. As can be seen the test set appears to rapidly converge to one that effectively kills all of the variables. In fact, by the 4th build it appears that the test provided better than 99% coverage and those perturbation variables that were still randomly surviving were only rarely doing so.

Table 6 presents similar results for test set 6. As can be seen, this test set did not seem to converge as quickly. Even by the 9th cycle several of the perturbation variables were frequently surviving.

Table 6. Detailed Perturbation Analysis for Test Set 6

	Number Times Survived									
Perturbation Variable build # =>	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**	**P10**
ALLOCcompareTo1								2		
backTrackDeep2	2		2	1			1	1	4	1
compareFaults10	1	1								
compareFaults11					1					
comparePrimaryMaint1	1	1	1	1	4		2	3	4	
compareRedundMaint1	2	4	3	2	1			2	3	1
main3	2	2	2	2		1		1	1	
MRprocess3	2	1	4	1	3					
Pair1				2			1	2	1	
TASKcompareTo5	2									
TASKcompareTo6		1		1						
TASKcompareTo9							1	1		
TaskReader1	2		1	3	1	1	1	2		
TaskReader3		2	4	1	2		1	1		
TaskReader4	4	1		2	5	2	1	1		
TaskReader5	2		1	3	2	1	1			
Total	20	13	18	19	19	5	9	16	13	2

Table 7. Test Distribution for Test Sets 1 & 6

Test Set 1		
Test No	Total Kills	Percent
0	3027	28.0%
1	7616	70.5%
2	2814	26.1%
3	3617	33.5%
4	4609	42.7%
5	3818	35.4%
6	3653	33.8%
7	3331	30.8%
8	3531	32.7%
9	4402	40.8%
10	3670	34.0%
11	3411	31.6%
12	3029	28.0%
13	3952	36.6%
14	3738	34.6%
15	2500	23.1%
16	3499	32.4%
17	4140	38.3%
18	3280	30.4%
19	3276	30.3%
20	3194	29.6%
21	3919	36.3%
22	3822	35.4%
23	6199	57.4%
24	4336	40.1%
25	7745	71.7%
26	2776	25.7%
27	6997	64.8%
28	4558	42.2%
29	3966	36.7%

Test Set 6		
Test No	Total Kills	Percent
0	4038	37.4%
1	2822	26.1%
2	3329	30.8%
3	796	7.4%
4	3280	30.4%
5	3340	30.9%
6	3900	36.1%
7	4238	39.2%
8	4701	43.5%
9	3424	31.7%
10	4406	40.8%
11	3974	36.8%
12	6518	60.4%
13	4100	38.0%
14	3665	33.9%
15	3306	30.6%
16	3398	31.5%
17	4111	38.1%
18	4080	37.8%
19	3779	35.0%
20	7061	65.4%
21	4195	38.8%
22	7087	65.6%
23	3535	32.7%
24	6762	62.6%
25	4377	40.5%
26	3625	33.6%
27	3895	36.1%
28	3459	32.0%
29	5380	49.8%

Table 7 provides a breakdown of how often each test is actually used to kill each perturbation variable. For this breakdown, each test set was analyzed using 100 random perturbations for each variable and the tool option EVALUATE_TESTS was used to perform a complete evaluation of the test set. Since there are 108 perturbation variables, this results in more than 10000 possible perturbations for each test. From these tables, it is evident that both test sets have slightly different characteristics for their individual test usage statistics.

Tables 8 and 9 present the actual detailed test results of the program when testing perturbations for the "compareRedundMaint1" variable using test sets 1 and 6. These tables present the random perturbation value of the variable, whether the perturbation was killed, and the test that actually killed the perturbation. As can be seen, most of the survivals were a result of the random nature of the program. In fact, in no instance did a variable "always" survive for a given perturbation value. Conversely, it can be seen that in no instance did a specific test "always" kill a given perturbation. From these Tables it is obvious that a given value of the perturbation is often killed by different test cases.

Table 8. Compare RedundMaint1 Perturbations for Test Set 1

Test 1. compareRedundMaint1		
Pert Value	K/S	Test
0	K	1
0	K	10
0	K	1
0	K	10
0	K	10
0	K	10
0	K	1
0	K	1
1	K	1
1	K	25
1	K	1
1	K	1
1	K	1
1	K	27
1	K	1
2	K	1
2	K	25
2	K	27
2	K	1
2	K	1
2	K	1
2	S	
2	K	1
2	K	1
2	K	27
2	K	23
2	S	
2	K	1
2	K	1
3	K	1
3	K	1
3	K	25
3	K	1
3	K	1
3	K	23
3	K	1
3	K	23
4	K	1
4	K	1
4	K	1
4	K	1
4	S	
4	K	25
4	K	23
4	K	1
4	K	23
4	K	1
4	K	1
4	K	1
4	K	1
4	K	1
4	K	1
5	K	1
5	K	23
5	K	1
5	K	25
5	K	1
5	K	25
5	K	1
5	S	
5	K	23
5	K	1
5	K	23
6	K	25
6	K	1
6	K	1
6	K	1
6	K	1
6	K	1
6	K	25
6	K	1
6	S	
6	K	1
6	K	23
6	K	1
7	K	1
7	K	23
7	K	23
7	K	1
7	K	25
7	K	23
7	K	1
7	K	1
7	S	
7	K	1
8	K	27
8	K	1
8	K	1
8	K	23
8	K	23
8	K	1
8	K	1
8	K	23
8	K	1
8	K	23
9	K	25
9	K	27
9	K	25
9	K	23
9	K	1

Table 9. Compare RedundMaint1 Perturbations for Test Set 6

Test 6. compareRedundMaint1		
Pert Value	K/S	Test
0	K	7
0	K	7
0	K	7
0	K	7
0	K	7
0	K	7
0	K	7
0	K	7
0	K	7
1	S	
1	K	24
1	K	20
1	K	22
1	K	12
1	K	20
1	K	12
1	K	12
1	K	12
1	K	20
1	K	12
2	K	20
2	K	12
2	K	20
2	K	12
2	K	12
2	K	12
3	K	29
3	K	22
3	K	22
3	K	12
3	K	20
3	K	12
3	K	20
3	K	12
3	K	20
3	S	
3	K	20
3	K	12
3	K	12
4	K	29
4	K	12
4	K	22
4	K	12
4	K	12
4	K	12
4	K	12
4	K	12
4	K	20
4	K	12
4	K	20
4	K	12
5	K	12
5	K	12
5	S	
5	K	22
5	K	20
5	K	12
5	K	20
5	K	20
5	K	12
5	K	12
5	K	12
6	K	12
6	K	20
6	K	24
6	K	20
6	K	22
6	K	20
6	K	12
6	K	12
6	K	20
7	K	20
7	K	12
7	S	
7	S	
7	K	12
7	K	20
7	K	12
7	K	12
7	K	12
7	K	12
7	K	20
7	K	12
8	K	12
8	K	12
8	K	29
8	K	12
8	S	
8	K	20
8	K	12
8	K	12
8	K	12
8	K	20
9	K	12
9	K	12
9	K	12
9	K	12
9	K	20
9	K	12
9	K	12
9	K	12

In order to ensure that the "killed" perturbation variables were not an artifact of the unordered collections, further analysis was performed on the characteristics of Test Sets 1 and 6. These test sets were analyzed to determine the degree to which the test results could be influenced by the random nature of the algorithm. This analysis consisted of executing each test 200 times and determining the number of times the program came up with a different answer. The results for test set 1 as shown in Table 10 show that 4 tests resulted in different results. In fact, test numbers 1 and 25 showed a significant potential for this behavior. Test 1 generated different results 43% of the time and test 25 generated different results 54% of the time. Tests 23 and 27 in test set 1 exhibited differing test results 29% and 31% of the time.

This large difference in results is concerning for this particular test set. For example, in Table 8 it is shown that the `compareRedundMaint1` perturbation variable with a value of 2 survived twice and was killed with tests 1, 23, 25, and 27. However since these tests are shown to exhibit a highly unstable output, the detected kills may be a result of this attribute instead of the tail function of this perturbation.

The results for test set 6 are also illustrated in Table 10. This shows that 3 tests resulted in different results. Test number 20 seemed to have the most potential for different results. Test 20 generated different results 26 % of the time while test 24 generated different results 23% of the time. Test number 29 only generated different results 6% of the time and no other tests in test set 6 were found to result in different answers.

Table 10. Test Differences Due to Non-Determination for Test Sets 1 & 6

Test Set 1 (each test run 200 times)			Test Set 6 (each test run 200 times)		
Test No	Diff	Percent of Time Diff Found	Test No	Diff	Percent of Time Diff Found
0	0	0%	0	0	0%
1	86	43%	1	0	0%
2	0	0%	2	0	0%
3	0	0%	3	0	0%
4	0	0%	4	0	0%
5	0	0%	5	0	0%
6	0	0%	6	0	0%
7	0	0%	7	0	0%
8	0	0%	8	0	0%
9	0	0%	9	0	0%
10	0	0%	10	0	0%
11	0	0%	11	0	0%
12	0	0%	12	0	0%
13	0	0%	13	0	0%
14	0	0%	14	0	0%
15	0	0%	15	0	0%
16	0	0%	16	0	0%
17	0	0%	17	0	0%
18	0	0%	18	0	0%
19	0	0%	19	0	0%
20	0	0%	20	52	26%
21	0	0%	21	0	0%
22	0	0%	22	0	0%
23	57	29%	23	0	0%
24	0	0%	24	45	23%
25	107	54%	25	0	0%
26	0	0%	26	0	0%
27	62	31%	27	0	0%
28	0	0%	28	0	0%
29	0	0%	29	12	6%

It should be noted that since a random ordering process in the JAVA interpreter drives this behavior, perturbation of the program might result in other tests exhibiting a

similar behavior. In any case, since test set 6 was exhibiting perturbation variables being killed in excess of 94% of the time (as shown in Table 4), and since the tests were only generating different results less than 26% of the time, it is believed that even for this worst case analysis, the random process for this test set does not play a significant role in its evaluation.

Table 11. Timing of Test Set 1 Generation

	Build Time to Complete	Number Inputs	Average Inputs/Sec	Eval Time to Complete	Number Inputs	Average Inputs/Sec
P1	4:03:29.00	15803	00.924	1:22:59.37	5044	00.987
P2	3:13:44.00	12307	00.945	1:38:34.30	6193	00.955
P3	2:41:24.95	10578	00.916	1:40:50.12	6336	00.955
P4	2:24:12.00	9406	00.920	1:09:15.00	4318	00.962
P5	0:24:06.65	1546	00.936	1:11:32.54	4467	00.961
P6	1:40:46.73	6584	00.918	1:11:07.03	4441	00.961
P7	0:27:18.00	1767	00.927	1:08:15.40	4258	00.962
P8	0:06:54.74	431	00.962	1:24:58.89	5244	00.972
P9	1:13:42.18	4790	00.923	1:27:35.87	5412	00.971
P10	0:08:59.48	539	01.001	1:35:35.69	5618	01.021
TOTAL	16:24:37.72		00.937	13:50:44.21		00.971

Another item of interest was the actual time required to generate the test sets. It should be noted that these tests were run on a 1.2 GHz Athelon portable laptop computer. On the average it took slightly less than 1 second (0.96 sec) to generate a random test case and to run both the original and perturbed version of the prototype against that test case. (In point of fact, the majority of this time seemed to be a result of running the test against the prototype code.) Therefore, the amount of time it took to build a specific set of test cases was a direct result of the number of inputs it took to resolve the perturbations. Table 11 illustrates the time each step of the build process took for test set 1. As can be seen the entire build process took about 16 hours while the incremental

evaluation used almost 14 hours. Thus the entire build-evaluate for 10 cycles took more than 30 hours.

Table 12. Timing of Test Set 6 Generation

	Build Time to Complete	Number Inputs	Average Inputs/Sec	Eval Time to Complete	Number Inputs	Average Inputs/Sec
P1	5:57:33.54	23372	0.918	1:52:45.40	7194	0.940
P2	1:07:48.50	4441	0.916	1:52:00.12	7099	0.947
P3	1:27:43.87	5690	0.925	1:54:43.57	7095	0.970
P4	1:21:34.91	5169	0.947	1:57:23.46	7448	0.946
P5	3:19:18.37	13022	0.918	2:00:07.92	7606	0.948
P6	4:05:45.38	16010	0.921	1:54:14.85	7253	0.945
P7	0:17:08.32	1050	0.979	2:00:30.88	7450	0.971
P8	2:45:14.62	10812	0.917	2:05:55.13	7981	0.947
P9	1:29:29.90	5842	0.919	2:03:04.35	7794	0.947
P10	2:03:34.76	8006	0.926	1:55:49.79	7037	0.988
TOTAL	23:55:12.18		0.929	19:36:35.47		0.955

It should be noted that the build cycle for test set 1 (at 30 hours) was the fastest encountered. The build-evaluate cycle for test set 6 (see Table 12) took over 43 hours while other test sets took as much as 70 hours. This was due to the fact that test set 1 converged relatively early to a set of tests which killed the perturbations and thus the time to build and evaluate follow-on cycles was much shorter. Test set 6 did not converge until much later and therefore took more time for the follow-on cycles.

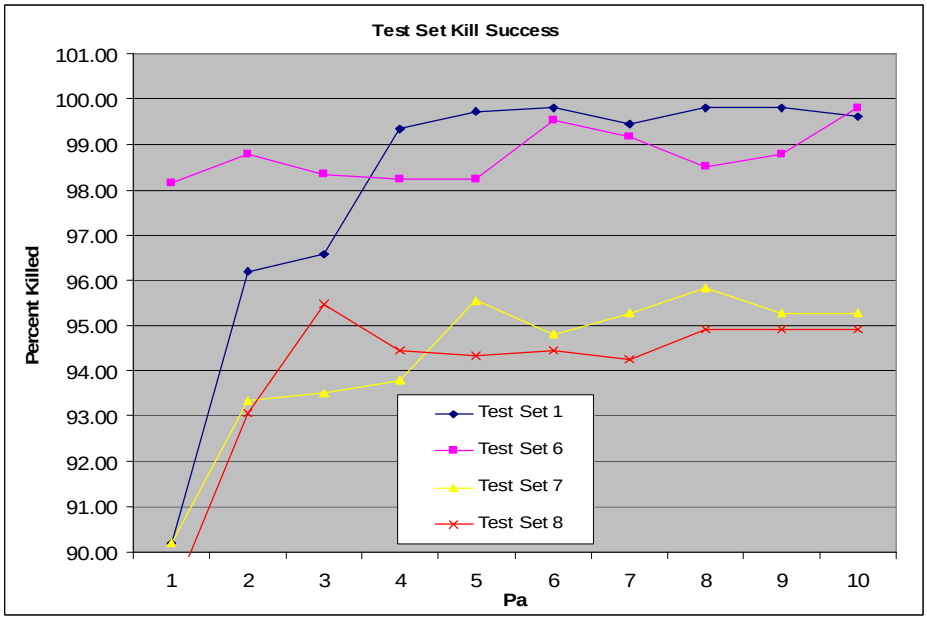
In a final attempt to account for the non-deterministic nature of the Java prototype, the program was modified to not officially kill a perturbation until it has been killed several times in a row. This modification in fact means that new tests are not added to the test set until they have been demonstrated to effectively kill a perturbation. Figure 4 and Table 13 show the results of building two independent test sets (Test sets 7 and 8) with the numKills variable set to 5. In order to compare the results of these test

sets with previously analyzed test sets, numKills was set to 1 for evaluating the test sets. As can be seen, this did not result in test sets that were better than those previously developed. In fact, in most cases the resulting test set was actually inferior to the initial test sets.

Table 13. Extended Test Set Analysis

	Test Set 1		Test Set 6		Test Set 7		Test Set 8	
Pa	TS Size	Percent Killed	TS Size	Percent Killed	TS Size	Percent Killed	TS Size	Percent Killed
1	17	90.19	21	98.15	22	90.19	25	88.98
2	24	96.20	23	98.80	26	93.33	25	93.06
3	25	96.57	23	98.33	27	93.52	25	95.46
4	26	99.35	24	98.24	30	93.80	25	94.44
5	27	99.72	26	98.24	31	95.56	25	94.35
6	28	99.81	27	99.54	31	94.81	26	94.44
7	29	99.44	28	99.17	31	95.28	26	94.26
8	29	99.81	28	98.52	31	95.83	27	94.91
9	30	99.81	29	98.80	31	95.28	28	94.91
10	30	99.63	30	99.81	31	95.28	28	94.91

Figure 4. Extended Test Set Comparison



The most probable reason for this behavior is that this modification prevents tests from being added until they can be shown to effectively kill a specific perturbation on there own. However, in the analysis above it was found that many of the perturbations were only effectively killed by a combination of several tests. For example, Table 8 shows that in Test set 1 the "`compareRedundMaint1`" perturbation variable was not always killed by a single test even for the same perturbation value. When the perturbation value was 2 for this variable, test 1 killed the perturbation 8 times and failed to kill it 6 times (it was actually killed 4 additional times by a combination of tests 23, 25, 27). Since the modified procedure prevents a test from being added even if it kills the perturbation 80% of the time, test 1 would never be added to the test set. In fact, no individual tests could be found to kill this perturbation with this method and therefore no test was added to the derived test set.

4.2. Analyzing Existing Test Sets with the JAVA Prototype

The previously existing test sets, those used to verify the JAVA prototype as well as those used to evaluate the shipboard delivery were also analyzed using the perturbation tool.

There were six individual test cases used to validate the JAVA prototype. It should be emphasized that this test set was chosen to exercise certain features in the algorithm and was not necessarily intended to verify the entire program. Therefore, it is not surprising that the perturbation analysis tool found that this test set was only 47% effective from a perturbation adequacy criterion.

The next step in the process was to analyze the test sets used to validate the delivered shipboard code. Once again it should be noted that these test sets were primarily constructed to evaluate the algorithm and not to verify the entire program. There were actually two sets of tests that were used to validate the pre-selection algorithm. These test sets were evaluated for perturbation adequacy using the analysis tool. The first test set (Mills Test Set) consisted of 50 tests and resulted in a perturbation adequacy of 72% while the second test set (Fones Test Set) which included 38 tests resulted in a perturbation adequacy of 73% (as compared with the 99% coverage by the derived test set). As was mentioned previously, since both of these test sets were developed as a proof of the algorithm, it is not particularly surprising that they do not provide wide coverage of the code.

4.3. Testing the Shipboard Code with a Perturbation-Adequate Test Set

The next step was to apply the derived test set against actual Tomahawk code. This investigation was performed on Test Set 6. Since the format of the test set is different for the actual Tomahawk code from the test sets used for the JAVA prototype, the first step was to convert the tests to the proper format. It should be noted that this conversion involved more than just changing file formats. Evolving requirements for the Tomahawk code that were not captured in the prototype drove much of the differences. For example, the prototype allowed a task to be specified which used multiple missile types but the actual system does not allow this. Another complication arose from differences in the way launch times are used in the actual system (they are ignored in the

prototype). In order to facilitate the conversion, a tool was developed which captured and thereby standardized the basic rules and assumptions that were used in the conversion. For comparison purposes, Appendix D presents both the JAVA test file and the converted test file for several selected tests.

The converted test set was then run with the actual Tomahawk program. This initial testing revealed some errors in the conversion and required the conversion tool be modified and the validation of the test sets be performed once again. After the third round of testing verified the correct operation of the test set, the program was run with the path verification tool developed by the Tomahawk software development team in support of the Tomahawk program. This tool provides an indication of the path coverage of a given test set against a section of Tomahawk code. For comparison purposes, the Fones Test Set (see Section 4.2) was also tested using the path analysis tool.

Appendix E provides the results of the validation tests. As can be seen the JAVA prototype and the actual Tomahawk system did not always come to the same results. For instance for the derived test 6.0, the JAVA program resulted in allocating 5 missiles, while the actual Tomahawk code allocated only 4. The difference in this case was a result of the difference in the allocation requirements for the two programs. For this example, the test included tasking with multiple different missiles types. This is permissible in the JAVA prototype but not in the actual Tomahawk program. Therefore, the conversion program establishes the desired missile type to be the first (highest priority) type in the task. In this example, this results in the Tomahawk program not identifying a compatible missile type for one of the tasks and therefore the program does

not allocate a missile for this task. It should be noted that the Fones test set uses all of the same missile type and therefore does not have this problem. However, this still did not result in the Fones tests generating identical results. This was determined to be a result of some of the detailed requirements changes for possible missile allocations. Since the JAVA prototype was an early proof of principal prototype and the requirements have changed somewhat since the JAVA prototype was developed, this was not unexpected. The case that they usually produced "similar" results was deemed sufficient to show the correct conversion of the JAVA derived tests. In all events the program produced valid allocations for the indicated tasking.

The path coverage tests were much more interesting and more useful to the study presented here. Table 14 illustrates the differences detected as a result of the path analysis instrumentation tests. The derived test set provided 70.5% (341 of the 484 possible paths) coverage of the code associated with the cell pre-selection algorithm. In comparison, the Fones test set provided only 59.1% (286 paths) path coverage. Although this difference may not be significant, it does illustrate that the derived test set provides more than what was previously considered adequate path coverage. It is anticipated that this level of path coverage could be improved further if the perturbation analysis was performed on the actual code instead of the JAVA prototype.

Table 14. Comparison of Path Coverage

Service Name	Total Paths	Paths Tested		
		Fones	Derived	Diff
AbstractAlgorithm::intervalsOverlap	1	0	1	1
AbstractAlgorithm::pruneAllocatedMissiles	12	5	8	3
AbstractAlgorithm::pruneBlockedMissilesStep	1	0	1	1
AbstractAlgorithm::pruneIncapableMissiles	3	2	3	1
SortedMissile::SortedMissile	7	4	7	3
SortedNonTacticalMissile::operator<	9	5	6	1
SortedTacticalMissile::computeResourceTriples	5	0	4	4
SortedTacticalMissile::hasWarrantyExpired	1	0	1	1
SortedTacticalMissile::operator<	11	0	9	9
SortedTacticalMissile::setPrimary	1	0	1	1
SortedTacticalMissile::SortedTacticalMissile	1	0	1	1
UsSubAlgorithm::computeMaintenanceStatusRankings	11	6	10	4
UsSubAlgorithm::computeMaintenanceStatusWeight	19	11	17	6
UsSubAlgorithm::computeMissileToTaskWeight	5	4	3	(1)
UsSubAlgorithm::computeSetRepresentations	7	5	7	2
UsSubAlgorithm::isProperSubset	5	4	5	1
UsSubAlgorithm::loadConstraintOneTaskPerMissile	9	8	9	1
UsSubAlgorithm::loadConstraintTubeUsage	11	10	11	1
UsSubAlgorithm::missilesAvailable	3	3	2	(1)
UsSubAlgorithm::pruneBlockedMissiles	25	0	17	17
XaBasedAlgorithm::solveProblem	3	3	2	(1)
All Other Services	334	216	216	0
TOTAL	484	286	341	55

CHAPTER 5 Future Work

One of the more exciting aspects regarding this study is that there are several areas for follow-on efforts. In fact, the method seems promising enough to justify additional work in the following subject areas.

The first and most obvious area would be automating the instrumentation function. This enhancement would automatically place calls to the PERT functions in the source code. Once the code was compiled it could then be used to test the application's perturbation behavior. This enhancement would be somewhat difficult to implement since it would require developing a pre-compiler to analyze the code and determine how to insert the necessary calls. As was previously mentioned, the Tomahawk development team at Dahlgren has already developed an automated tool that inserts instrumentation into a C++ source file to determine path coverage. It is believed that extending this tool to perform perturbation instrumentation would be possible and is currently being investigated.

Another area for further study would be to extend the perturbation function for other data types. The version evaluated in this report only performs perturbation on integer and boolean variables but there are several other data types that might be interesting to perturb. In particular, since the JAVA prototype of the cell pre-selection algorithm used stacks and unordered collections extensively, the ability to perturb these

data types would have been useful. It should be noted that this type of perturbation would be much more complex than the simple perturbations investigated here.

Using an artificial intelligence (AI) technique to generate the test sets could show significant improvement in the tool's ability to generate perturbation-adequate test sets. The tool currently generates test sets completely at random until it stumbles on an effective test. In the case of complex test sets this purely random generation may take an inordinately long time or may never find a valid test to kill a specific perturbation. Since this function is basically a search through a complex data set, using AI techniques to perform this search might be more efficient. This would require developing a scoring method for the "goodness" of a derived test and an analysis of the various AI search techniques such as genetic algorithms, neural networks, and simulated annealing to evaluate which method more rapidly converges to an optimal solution.

Finally, developing a more comprehensive result comparison for determining perturbation kills would be a useful area to research. As was mentioned previously, the unordered collections in JAVA result in the program producing different results for the same input. This would also be evident in any program employing random numbering or artificial intelligent algorithms to search through complex spaces. In short, it is necessary to differentiate between results that differ due to perturbation of the tail function versus those that differ due to a non-deterministic algorithm. An initial attempt was made to perform a broader comparison by not registering a kill until the perturbation had been killed multiple times. Unfortunately this did not result in an acceptable mechanism to address this problem and other methods should be investigated.

CHAPTER 6 Summary

In summary, the perturbation analysis technique was found to be an effective method that could be used not only to validate test set coverage, but also to automatically generate original test sets.

When the perturbation technique is used as a means to validate test set coverage, this would be in addition to path coverage such as that currently being validated for the Tomahawk system tests. It seems that providing another vector to ensure adequate test set coverage would be very useful, especially in critical weapons control system software. This additional verification could increase confidence in the software functionality and could effectively be used to validate adequate testing criteria.

Using the tool to automatically generate a set of test cases proved to be a very interesting exercise. In fact, it was observed that the tool derived a test set which provided equivalent path coverage to a manually derived test set which was developed to validate the proof of the algorithm. It should be particularly noted that the perturbation derived test set could be generated with significantly less effort and actually use fewer test cases. In addition, unlike the manually derived tests, the perturbation derived test set can be easily re-derived as program requirements evolve.

A major limitation to the use of this methodology is in the area of automatically generating test sets for a non-deterministic application. This currently requires extensive

analysis to verify the validity of differences in the outputs. Before the tool could effectively be applied to this type of application, a more extensive tail analysis comparison function would need to be developed.

One of the appealing side effects was the discovery that the tool could be used to find portions of the code that were not used. The JAVA prototype had several areas of code that were not needed and were not used by the algorithm. This code redundancy was a natural effect of the evolution of the program in that it was built simply to test the algorithmic concept and was not under a formal maintenance plan. The code redundancy crept in over time and was not evident through the use of any other tool. The developer saw significant potential in using this type of tool to clean up prototype code.

Another use of the perturbation analysis technique could be to test the robustness of the program. As part of a safety analysis, this tool could validate that the system test set includes “fail safe” tests that would exercise catastrophic error conditions.

Further analysis of the technique in actual weapon control system environments is recommended to validate its application in these areas.

References

[Car93] R. Carver, "*Mutation-Based Testing Of Concurrent Programs*", International Test Conference Proceedings, 1993, 845-853.

[DO93] R. A. DeMillo, A. J. Offutt, "*Experimental Results From An Automatic Test Case Generator*", ACM Transactions on Software Engineering and Methodology (TOSEM) 2-2, 1993, pp 109-127.

[FJT95] C. Fennemore, C. Johnson, R. Taft, S. Tallant, W. Tripp, "*Tomahawk Pre-desgination*", NSWCDD/TR-95/119, June 1995.

[GAL04] D. Galin, "*Software Quality Assurance – From Theory To Implementation*", Pierson Education Limited, 2004.

[HUA75] J. Huang, "*An Approach to Program Testing*", ACM Computing Serveys, Vol. 7, No. 3 September 1975.

[Kan00] C. Kaner, "*The Impossibility of Complete Testing*", Law of Software Quality Column, Software QA magazine, at: http://www.kaner.com/imposs.htm.

[MaWo92] A. Mathur, W. Wong, "*A Formal Evaluation of Mutation and Data Flow Based Test Adequacy Criteria*", Technical Report SERC-TR-133-P, Software Engineering Research Center, Purdue University, December 1992.

[MF03] J. Mills, C. Fennemore, "*Submarine Pre-desgination Test Cases*", NSWCDD/TR-03/4, February 2003.

[MM96] L. Morell, B. Murrill, *"Using. Perturbation Analysis to Measure Variation in the Information Content of Test Sets"*, Proceedings of the 1996 International Symposium on Software Testing and Analysis (ISSTA '96), pp 92-97.

[MMN92] K. Miller, L. Morell, R. Noonan, S. Park, D. Nicol, B. Murrill, J. Voas, "*Estimating the Probability of Failure When Testing Reveals No Failures*", IEEE Transactions on Software Engineering, Vol 18, No 1, Jan 1992.

[MMO02] B. Murrill, L. Morell, E. Olimpiew, "*A Perturbation-Based Testing Strategy*" Engineering of Complex Computer Systems, 2002. Proceedings. Eighth IEEE International Conference on 2-4 Dec. 2002: 145-152.

[MMR97] L. Morell, B. Murrill, R. Rand, "*Perturbation Analysis Of Computer Programs*" Computer Assurance, 1997. COMPASS '97. 'Are We Making Progress Towards Computer Assurance?'. Proceedings of the 12th Annual Conference on 16-19 June 1997: 77-87.

[MuMo94] B. Murrill, L. Morell, *"An Experimental Approach to Analyzing Software Semantics Using Error Flow Information"*, Proceedings of the 1994 International Symposium on Software Testing and Analysis (ISSTA '94).

[MR01] Murnane, T., Reed, K. "*On The Effectiveness Of Mutation Analysis As A Black Box Testing Technique*" Software Engineering Conference, 2001. Proceedings. 2001 Australian , 27-28 Aug. 2001: 12-20.

[OL94] Offutt, A.J., Lee, S.D. "*An Empirical Evaluation Of Weak Mutation*" IEEE Transactions on Software Engineering 20.5 (May 1994): 334-344.

[OfPa97] A. Offutt, J. Pan, *"Automatically Detecting Equivalent Mutants and Infeasible Paths"*, The Journal of Software Testing, Verification, and Reliability, Vol 7, No. 3, pages 165-192,September 1997.

[Oli97] E. Olimpiew, *"An Investigation of the Software Testing Coupling Effect Using Perturbation Analysis"*, Masters Thesis, Virginia Commonwealth University, May 1997.

[ORZ93] Offutt, A.J., Rothermel, G., Zapf, C. "*An Experimental Evaluation Of Selective Mutation*" Software Engineering, 1993. Proceedings., 15th International Conference on 17-21 May 1993: 100-107.

[TX02] P. Townend, J. Xu, *"Assessing Multi-Version Systems Through Fault Injection"*, Proceedings of the Seventh IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS'02) IEEE Computer Society Press, 2002, pp. 105-112.

[Wood90] Woodward, M.R. "*Mutation Testing-An Evolving Technique*" IEE Colloquium on Software Testing for Critical Systems (19 June 1990).

[YA00] T. Yu-Wen, W. S. Aldiwan, *"Automating Test Case Generation For The New Generation Mission Software System*", Aerospace Conference Proceedings, 2000, pp 431-437.

[ZHM97] H. Zhu, P. Hall, J. May, *"Software Unit Test Coverage and Adequacy"*, ACM Computing Surveys, Vol. 29, No. 4, December 1997: pp 366-427.

[Zei84] S. Zeil, *"Perturbation Testing For Computation Errors"*, Proceedings of the 7th international conference on Software engineering, March 1984, Pages: 257 – 265.

[Zei89] S. Zeil, "*Perturbation Techniques For Detecting Domain Errors*", IEEE Transactions on Software Engineering, Volume: 15, Issue: 6, June 1989 Pages: 737 – 746.

APPENDIX A

Perturbation Variables Definitions

This appendix includes a copy of the perturbations definition file. It is formatted according to Section 3.1.

```
Variable Perturbations are of the following format ...
Format:     Variable Name           Lo    Hi  Pert?(1 = Yes 0 = No)
                                                      // comment ...
******* Perturbations from Scenario.java
VAR_NAME: main1                     0     0     1
VAR_NAME: main2                     0     0     1
VAR_NAME: main3                     -5    10    1
VAR_NAME: backTrackDeep1            0     0     1
VAR_NAME: backTrackDeep2            -5    10    1
VAR_NAME: backTrackDeep3            0     0     1
VAR_NAME: backTrackDeep4            0     0     1
VAR_NAME: backTrackDeep5            0     0     1
VAR_NAME: backTrackDeep6            0     0     1
VAR_NAME: backTrackDeep7            -5    10    1
VAR_NAME: backTrackDeep8            0     0     1     // Hard to Find!!
VAR_NAME: backTrackDeep9            0     0     0     // Equivalent
VAR_NAME: backTrackDeep10           -10   10    0     // Equivalent
VAR_NAME: getNonTempAllocStack1     -5    10    1
VAR_NAME: getNonTempAllocStack2     0     0     1
VAR_NAME: allocate1                 0     0     1
VAR_NAME: allocate2                 0     0     1
VAR_NAME: allocate3                 0     0     1
VAR_NAME: allocate4                 0     0     1
VAR_NAME: missileIter1              0     0     0     // Equivalent
VAR_NAME: missileIter2              0     0     0     // Equivalent
VAR_NAME: missileIter3              0     0     0     // Equivalent
VAR_NAME: missileIter4              0     0     1
VAR_NAME: missileIter5              0     0     1
VAR_NAME: missileIter6              0     0     1
VAR_NAME: launcherIter1             0     0     1     // Hard to Find!!
******* Perturbations from Missile.java
VAR_NAME: Pair1                     -100  100   1
VAR_NAME: Pair2                     0     0     1
VAR_NAME: compareMissile1           0     0     1
VAR_NAME: compareMissile2           0     0     1
VAR_NAME: compareMissile3           0     0     1
VAR_NAME: compareMissile4           0     0     0     // Equivalent
VAR_NAME: compareMissile5           0     0     0     // Equivalent
VAR_NAME: compareMissile6           0     0     0     // Equivalent
VAR_NAME: compareFaults1            0     0     1
VAR_NAME: compareFaults2            0     100   1
VAR_NAME: compareFaults3            0     100   1
VAR_NAME: compareFaults4            0     0     1
VAR_NAME: compareFaults5            0     0     1
VAR_NAME: compareFaults6            0     0     1
VAR_NAME: compareFaults7            0     0     1
VAR_NAME: compareFaults8            0     0     1
VAR_NAME: compareFaults9            0     0     1
VAR_NAME: compareFaults10           0     10    1
VAR_NAME: compareFaults11           0     10    1
VAR_NAME: comparePrimaryMaint1      0     10    1
VAR_NAME: compareRedundMaint1       0     10    1
VAR_NAME: isCapableFor1             0     0     1
VAR_NAME: isCapableFor2             0     0     1
```

```
VAR_NAME: isCapableFor3             0     0     1
********** Perturbations from Allocation
VAR_NAME: conflictsWith1            0     0     0     // Equivalent
VAR_NAME: conflictsWith2            0     0     0     // Equivalent
VAR_NAME: conflictsWith3            0     0     0     // Equivalent
VAR_NAME: compareAllocation1        0     100   1
VAR_NAME: compareAllocation2        0     100   1
VAR_NAME: compareAllocation3        0     0     1
VAR_NAME: compareAllocation4        0     0     1
VAR_NAME: compareAllocation5        0     0     1
VAR_NAME: compareAllocation6        0     0     1
VAR_NAME: ALLOCcompareTo1           0     10    1
VAR_NAME: ALLOCcompareTo2           0     0     1
VAR_NAME: ALLOCcompareTo3           0     0     1
VAR_NAME: ALLOCcompareTo4           0     10    1
VAR_NAME: ALLOCcompareTo5           0     0     1
VAR_NAME: ALLOCcompareTo6           0     0     1
VAR_NAME: ALLOCcompareTo7           0     0     1
VAR_NAME: ALLOCcompareTo9           0     0     1
********** Perturbations from Task
VAR_NAME: setPrefRedundantRole1     0     0     1
VAR_NAME: setPrefRedundantRole2     0     0     1
VAR_NAME: setPrefRedundantRole3     0     0     1
VAR_NAME: TASKgetPriority1          0     0     1
VAR_NAME: TASKcompareToA1           -1    2     0     // Equivalent
VAR_NAME: createCapAllocList1       0     0     1
VAR_NAME: createCapAllocList2       0     0     1
VAR_NAME: createCapAllocList3       0     0     1
VAR_NAME: createCapAllocList4       0     0     1     // Hard to Find!!
VAR_NAME: createCapAllocList5       0     1000  0     // Equivalent
VAR_NAME: TASKcompareTo1            0     100   1
VAR_NAME: TASKcompareTo2            0     100   1
VAR_NAME: TASKcompareTo3            0     0     1
VAR_NAME: TASKcompareTo4            0     0     1
VAR_NAME: TASKcompareTo5            0     100   1
VAR_NAME: TASKcompareTo6            0     100   1
VAR_NAME: TASKcompareTo7            0     0     1
VAR_NAME: TASKcompareTo8            0     0     1
VAR_NAME: TASKcompareTo9            0     1000  1
VAR_NAME: TASKcompareTo10           0     1000  1
VAR_NAME: TASKcompareTo11           0     0     1
********** Perturbations from MissileMovementCostTable
VAR_NAME: getCost1                  0     0     1
VAR_NAME: getCost2                  0     0     1
VAR_NAME: getCost3                  0     0     0     // Equivalent
VAR_NAME: getCost4                  0     0     0     // Equivalent
VAR_NAME: getCost5                  0     0     1
VAR_NAME: getCost6                  0     0     1
VAR_NAME: getCost7                  0     0     1
VAR_NAME: getCost8                  0     100   1
********** Perturbations from Allocations
VAR_NAME: addAllocation1            0     0     1
VAR_NAME: removeAllocation1         0     0     1
VAR_NAME: containsAllocation1       0     0     0     // Equivalent
```

```
VAR_NAME: containsAllocFor1          0     0     0     // Equivalent
VAR_NAME: getAllocationFor1          0     0     0     // Equivalent
VAR_NAME: containsAllocFor2          0     0     1
VAR_NAME: getAllocationFor2          0     0     1
VAR_NAME: getAllocationsFor1         0     0     1
VAR_NAME: getAllocationSet1          0     0     0     // Equivalent
VAR_NAME: getAllocationSet2          0     0     0     // Equivalent
VAR_NAME: getAllocationSet3          0     0     0     // Equivalent
VAR_NAME: getChosenAllocSet1         0     0     0     // Equivalent
VAR_NAME: getConflicts1              0     0     1
VAR_NAME: getConflicts2              0     0     1
VAR_NAME: getConflicts3              0     0     1
VAR_NAME: getConflicts4              0     0     1
********** Perturbations from Location
VAR_NAME: LOCaccessor1               0     0     1
VAR_NAME: LOCaccessor2               0     0     1     // Hard to Find!!
VAR_NAME: LOCaccessor3               0     0     1
VAR_NAME: LOCaccessor4               0     0     0     // Equivalent
VAR_NAME: LOCaccessor5               0     0     1     // Hard to Find!!
********** Perturbations from AllocationInterval
VAR_NAME: overlaps1                  0     0     1
VAR_NAME: overlaps2                  0     0     1
VAR_NAME: overlaps3                  0     0     1
********** Perturbations from MissileReader
VAR_NAME: MRprocess1                 -100  1000  0     // Equivalent
VAR_NAME: MRprocess2                 -100  1000  0     // Equivalent
VAR_NAME: MRprocess3                 -100  1000  1
********** Perturbations from MissileComponent
VAR_NAME: MCsize1                    -1000 1000  0     // Equivalent
VAR_NAME: MCisFaulted1               0     0     1
VAR_NAME: MslComponent1              0     0     1
VAR_NAME: MCcontainsKey1             0     0     0     // Equivalent
********** Perturbations from TaskReader
VAR_NAME: TaskReader1                -100  100   1
VAR_NAME: TaskReader2                0     0     1
VAR_NAME: TaskReader3                -100  100   1     // Hard to Find!!
VAR_NAME: TaskReader4                -100  100   1
VAR_NAME: TaskReader5                -100  100   1
********** Perturbations from XMIDList
VAR_NAME: XMIDList1                  0     0     1
VAR_NAME: XMIDList2                  -10   10    1
VAR_NAME: XMIDList3                  0     0     0     // Equivalent
VAR_NAME: XMIDList4                  0     0     1
VAR_NAME: XMIDList5                  0     0     0     // Equivalent
VAR_NAME: XMIDList6                  0     0     0     // Equivalent
```

APPENDIX B

Perturbation Instrumentation Tool Listings

This appendix includes the code listing for the instrumentation portion of the tool kit used in the perturbation study. This is more fully described in Section 3.2 of the report.

```
//---------------------------------------------------------------------
//   Richard Stutler
//      Perturbation functions.  This static class performs the actual
//      perturbation of the variables.  It determines the variable name
//      to perturb and parameters from the perturbation file.
//---------------------------------------------------------------------

package ssn.data;                                   // include in the ssn.data package

import java.io.*;
import java.lang.*;
import java.util.*;

public class PERT  {

        private static String PertName;                     // name of variable
        private static int numTimes;                        // number of times variable has been accessed
        private static int PertVal;                         // value of random perturbation variable
        private static String pertDatafile="PERT.DAT";      // parameter data file
        private static boolean verifyingPerts = false;      // set to true to verify all perturbation variable names
        private static int NumVars;                         // number of perturbation variables
        private static String pertVarFile="..\\..\\perturbations.txt";
        private static String VarName[];                    // name of variableS

   //---------------------------------------------------------------------
   //  Initializes the Perturbation functions from the specified perturbation FileName
   //      First reads in the perturbation file to determine the variable to perturbate
   //          and the parameters under which it should be perturbated.
   //      Also sets the access counter for the variable to 0
   static void PerturbStart() throws IOException  {
        BufferedReader in;
        String line;
        StringTokenizer ST;    // used to parse input commands

        File pertFile = new File(pertDatafile);
        if (pertFile.exists()) {                                        // only do this if file exists
               in = new BufferedReader(new FileReader(pertDatafile));   // use BufferedReader to input from file
               line = in.readLine();
               while (line!=null)  {
                      if(line.startsWith("PERT_LOC:"))  {               // get the specific variable to perturb
                             ST = new StringTokenizer(line.substring(10)," \t");
                             PertName = ST.nextToken();
                             PertVal= Integer.valueOf(ST.nextToken()).intValue();
                      }
                      if(line.startsWith("VER_PERT:"))  {               // set the verify perturbations flag
                             ST = new StringTokenizer(line.substring(10)," \t");
                             pertVarFile = ST.nextToken();              //     and get the perturbations file name
```

```
                        verifyingPerts = true;
                }
                line = in.readLine();                                   // read all the lines in
        }
        numTimes=0;                                                     // initialize the number times counter
        in.close();                                                     // close the input file
    }
    if (verifyingPerts) {                                               // set up the variable name array if verifying
        File pertVars = new File(pertVarFile);
        if (pertVars.exists()) {                                        // only do this if file exists
            in = new BufferedReader(new FileReader(pertVarFile));
            line = in.readLine();
            while (line != null) {                                      // first count number of perturbation variables
                if(line.startsWith("VAR_NAME:")) NumVars++;
                line = in.readLine();
                }
            VarName = new String[NumVars];                              //   and size the array as appropriate
            in.close();
            in = new BufferedReader(new FileReader(pertVarFile));       // restart reader
            line = in.readLine();
            int ptr = 0;
            while (line!=null) {                                        // scan entire file
                if(line.startsWith("VAR_NAME:"))  {                     // read in the name for each variable
                    ST = new StringTokenizer(line.substring(10)," \t");
                    VarName[ptr]=ST.nextToken();
                    ptr++;
                }
                line = in.readLine();
            }
            in.close();
        }
        else {
            System.out.println(" !!! ERROR: File Name for verifying Perts not found");
            verifyingPerts=false;
        }
    }
} // PerturbInit

//---------------------------------------------------------------------
//  These two functions do the actual perturbations ...
//    integer : Does a random integer perturbation based on the min and max integer values.
//    boolean : Returns the inverse of the input
//    used the overload capability in JAVA to impliment both with same name.
static int Var (String Name , int Default)  {
    int RV = Default;
    if (verifyingPerts) verifyName(Name);
    if(PertName.equals(Name) && numTimes<1){            // this is variable to perturb & first time
```

```
                RV = PertVal;                                   //   so return the random integer
                numTimes++;                                     // increment the access counter
        }
        return RV;
    } // PertInt
    static boolean Var (String Name , boolean Default)  {
        boolean RV = Default;
        if (verifyingPerts) verifyName(Name);
        if(PertName.equals(Name) && numTimes<1) {           // this is variable to perturb & first time
                RV = !Default;                                  //   so return the compliment of the input
                numTimes++;                                     // increment the access counter
        }
        return RV;
    } // PertBool

    //-------------------------------------------------------------------------
    //  Verify that the specified name is already in the pert array.
    static void verifyName(String Name)  {
        boolean fnd = false;                                    // found flag
        for (int i =0; i<NumVars; i++)  {                       // look at all names
                if (Name.equals(VarName[i])) fnd = true;        // set flag if found
        }
        if (!fnd)                                               // display error if not found
                System.out.println(" !!! ERROR: Invalid Perturbation Variable Name " + Name);
    }

} // class Perturb
```

APPENDIX C

Perturbation Analysis Tool Listings

This appendix includes the code listings for the analysis portion of the tool kit used in the perturbation study. This is more fully described in Section 3.3 of the report. The following files are included:

pertTool.java	The main program module
manageTF.java	Manages the Test files including setting up the arrays and developing random test files
manageLog.java	Handles writes to the log file
managePERT.java	Manages the perturbation functions including setting up the arrays and determining which variable to perturb.
spFun.java	Some special general purpose functions

```
//-----------------------------------pertTool-----------------------------------
//   Richard Stutler
//   CSMC 698 - Thesis
//
//     Several functions are actually included here -----
//         main:   The main program module
//         processCommandLine:       Reads inputs from the command line and parses input
//                                   from the parameter file by using "readParameters".
//         killPerts:        Kills the perturbations by adding to the test set
//         evaluatePerts:    Evaluates the specified test set against the perturbations
//         pertKilled:               Runs the original and perturbed programs to determine if there is a
//                                   difference and the difference persists for multiple times
//------------------------------------------------------------------------------

import java.io.*;
import java.lang.*;
import java.util.*;

class pertTool   {
       static String testDirectory="";             // directory to store the test files
       static int maxTriesToKill=0;                // How many times try to kill?
       static String logFileName="";               // file name for log data
       static String reportTitle="";               // default title for log report
       static String pertList="";                  // file for perturbation list
       static String pertDir="";                   // directory to save PERT.DAT file
       static String pertDataFile="PERT.DAT";      // file name where specific PERT info saved
       static int numInputs=0;                     // number of inputs
       static boolean evaluateOnly=false;          // flag to force evaluate test set only
       static int noPertLoops=1;                   // number of perturbation loops
       static String pertToTest[];                 // array of perturbations to test
       static int killCnt=0;                       // number of perts killed
       static int surviveCnt=0;                    // number of perts survived
       static Calendar startTimeC, stopTimeC;      // calander for start and stop times for instrumentation
       static String outRegular ="regout.txt";     // temporary output files for tests
       static String outPerturb ="prtout.txt";     //
       static boolean verifyPerts = false;         // set to true to verify all perturbation variable names
       static int EVALstrt=0;                      // for partial evaluation of an existing test set start
       static int EVALstop=-1;                     //   and stop with these tests (-1 means go to end)
       static int numKills = 1;                    // Number of times a perturbation must be killed before
                                                   //    actually considered killed (non-deterministic addition)
       static manageTF  TF;                        // test files class
       static managePERT PERT;                     // perturbations class
       static boolean evaluateTestsBool = false;   // set to true to do a full test evaluation
```

```
//---------------------------------------------------------------
// Main program module
public static void main (String [] args) throws IOException      {

     pertToTest = new String[50];                                // limit the number of specific perts to 50
     pertToTest[0]="*";                                          // default is to test all '*'
     if (processCommandLine(args)==0) return;                    // get the command line args
     startTimeC = Calendar.getInstance();                        // get start time for timestamp
     TF    = new manageTF(1000,testDirectory);                   // initialize Test File array (max of 1000 tests)
     if (EVALstop<=0)  EVALstop = TF.numTFs()-1;                 // setup the evaluate start and stop values
     PERT = new managePERT(pertList,pertDir+pertDataFile);       // and Pert arrays
     manageLog.InitLogFile(logFileName, TF, PERT);               // start log file
     if (!pertToTest[0].equals("*")){                            // if not testing all perts
            int j;                                               //      this will set the doit flag
            for (j = 0; j<PERT.getNumVars(); j++)                //      for each as appropriate
                   PERT.setDoIt(j,false);                        // set doit flag for all to false
            for (j = 0; j<50; j++) {                             //     then selectively reset it
                   if (pertToTest[j]!=null) {                    //     for only those specified
                          int R = PERT.findName(pertToTest[j]); //       1. find name in PERT
                          if(R!=-1) PERT.setDoIt(R,true);       //       2. and set doit flag
                   }
            }
     }
     numInputs=0;  killCnt=0;
     if (evaluateOnly==false)
            manageLog.writeToLogFile("Attempting to kill perturbations by adding to test set");
     else   {
            manageLog.writeToLogFile("Evaluating existing test set for perturbations");
            manageLog.writeToLogFile("   Evaluating Tests  " + EVALstrt + " to " + EVALstop);
            manageLog.writeToLogFile("");
     }
     for (int i = 0 ; i<noPertLoops ; i++)  {
            System.out.println("------------ Testing Pa : " + i + " out of " + (noPertLoops-1) + " ---------------");
            if (evaluateOnly==false)  killPerts(i);
            else  {
                   if (!evaluateTestsBool) evaluatePerts(i);
                   else evaluateTests(i);
            }
     }
     System.out.println("Done");
     stopTimeC = Calendar.getInstance();                         // get end time for timestamp
     System.out.println("Analysis started at: " + startTimeC.get(Calendar.HOUR) + ":" +
                   startTimeC.get(Calendar.MINUTE) + ":" +       startTimeC.get(Calendar.SECOND) );
     System.out.println("Analysis ended at:   " + stopTimeC.get(Calendar.HOUR) + ":" +
                   stopTimeC.get(Calendar.MINUTE) + ":" + stopTimeC.get(Calendar.SECOND) );
     manageLog.closeLogFile();
}  // main
```

```
//---------------------------------------------------------------
// killPerts routine
//    This routine attempts to kill the perturbations.  It will first use all of the existing test sets
//    and if it cannot kill the perturbation it will build new test sets to try to kill the  perturbation.
public static void killPerts(int CNT)  throws IOException {
      boolean kill = false;
      for (int i = 0 ; i<PERT.getNumVars() ; i++)  {
             kill = false;
             if (PERT.getDoIt(i))  {
                int PPP = PERT.setLocTest(i);
                System.out.print("Trying to kill : " + PERT.getPertName(i) + " with value " + PPP + " ==> ");
                System.out.print("Existing Test No: ");
                for (int j = 0 ; j<TF.numTFs() ; j++) {          // first test against existing test set
                    System.out.print(spFun.NumberToString(j+1,4,' ')+ "\b\b\b\b");
                    TF.getTF(j,TF.tfName);                       // get next test from the test set
                    numInputs++;
                    kill = pertKilled(TF.tfName);
                    if (kill)  {                                 // perturbation was killed !!
                           System.out.println("\rPa:" + CNT + "  Killed perturbation : " + PERT.getPertName(i));
                           killCnt++;
                           PERT.setKilled(i,j);
                           j=TF.numTFs()+1;                      // perturbation killed so force exit
                    }                                            //    otherwise fall through and continue testing
                }                                                //        against existing tests
                if (!kill)  {                                    // none of the existing tests killed this perturbation
                System.out.print("\rTrying to kill : " + PERT.getPertName(i) + " with value " + PPP + " ==> ");
                    System.out.print("  New Test No: ");         //    therefore need to generated new tests
                    for (int k = 0 ; k<maxTriesToKill ; k++)  {
                           System.out.print(spFun.NumberToString(k+1,7,' ')+ "\b\b\b\b\b\b\b");
                           TF.buildRandomTestFile(TF.tfName);    // build a new random test file
                           numInputs++;
                           kill = pertKilled(TF.tfName);
                           if(kill)  {                           // difference in outputs was detected
                                  System.out.println("\rPa:" +CNT + "  Killed perturbation : " + PERT.getPertName(i));
                                  killCnt++;
                                  int T = TF.addTF(TF.tfName);  // add test file to test set
                                  PERT.setKilled(i,T);
                                  k=maxTriesToKill+1;           // force exit
                           }
                    }
                }
                if (!kill)  {                                    // perturbation still not killed
                    System.out.println("\rPa:" +CNT + "  I GIVE UP TRYING TO KILL PERTURBATION : "+ PERT.getPertName(i));
                    surviveCnt++;
                }
                manageLog.writeToLogFile("Perturbation : " + PERT.getPertName(i) + " with PERT value of " + PPP);
```

```
               if (kill) manageLog.writeToLogFile("     Killed with Test No : " + TF.getTFname(PERT.getKilledWith(i)));
               else   manageLog.writeToLogFile("     SURVIVED");
            }
            else System.out.println("Bypassing tests for : " + PERT.getPertName(i));
      }
} // killPerts


//-----------------------------------------------------------------
// evaluatePerts routine
//    This routine evaluates the perturbations.  It will use all (or a specified portion) of the
//    existing test set to attempt to kill the perturbation and develop statistics for the kill ratios.
public static void evaluatePerts(int CNT)  throws IOException      {
      for (int i = 0 ; i<PERT.getNumVars() ; i++)  {
            int R=0;
            if (PERT.getDoIt(i))  {
               int PPP = PERT.setLocTest(i);
               System.out.print("Trying to kill : " + PERT.getPertName(i) + " with value " + PPP + " :: ");
               System.out.print("Existing Test No: ");
               for (int j = EVALstrt ; j<=EVALstop ; j++) {             // test against existing test set
                    System.out.print(spFun.NumberToString(j,4,' ')+ "\b\b\b\b");
                    TF.getTF(j,TF.tfName);                              // get next test from the test set
                    numInputs++;
                    spFun.DosCom("runBoth.bat " + TF.tfName + " " + outRegular + " " + outPerturb);
                    R = compare(outPerturb, outRegular);
                    if (R!=0)  {                                        // perturbation was killed !!
                           System.out.println("\rPa:" +CNT + "  Killed perturbation : " + PERT.getPertName(i));
                           killCnt++;
                           PERT.setKilled(i,j);
                           j=TF.numTFs()+1;                             // perturbation killed so force exit
                    }                                                   //      otherwise fall through and continue
               }
               if (R==0)  {                                             // perturbation still not killed
                    System.out.println("\rPa:" +CNT + "  I GIVE UP TRYING TO KILL PERTURBATION : "+ PERT.getPertName(i));
                    surviveCnt++;
               }
               manageLog.writeToLogFile("Perturbation : " + PERT.getPertName(i) + " with PERT value of " + PPP);
               if (R!=0) manageLog.writeToLogFile("     Killed with Test No : " + TF.getTFname(PERT.getKilledWith(i)));
               else   manageLog.writeToLogFile("     SURVIVED");
            }
            else System.out.println("Bypassing tests for : " + PERT.getPertName(i));
      }
} // evaluatePerts
```

```
//---------------------------------------------------------------
// evaluateTests routine
//    This routine performs a complete evaluation of the test set.  It will use all of the
//    existing test set to kill the perturbation and develop statistics for the kill ratios.
public static void evaluateTests(int CNT)  throws IOException      {
      manageLog.writeToLogFile("*******Performing a complete test evaluation*********");
      for (int i = 0 ; i<PERT.getNumVars() ; i++)  {
             int R=0;
             if (PERT.getDoIt(i))  {
                int PPP = PERT.setLocTest(i);
                for (int j = EVALstrt ; j<=EVALstop ; j++) {             // test against existing test set
                     TF.getTF(j,TF.tfName);                              // get next test from the test set
                     numInputs++;
                     spFun.DosCom("runBoth.bat " + TF.tfName + " " + outRegular + " " + outPerturb);
                     R = compare(outPerturb, outRegular);
                     if (R!=0)  {                                        // perturbation was killed !!
                            System.out.println("Pa:"+CNT+"  Killed perturbation : "+PERT.getPertName(i)+" with test "+j);
                            killCnt++;
                            PERT.setKilled(i,j);
                            manageLog.writeToLogFile("Perturbation :  "+PERT.getPertName(i)+"\t with PERT value of "+PPP+
                                                "\t   Killed with Test : " + TF.getTFname(j));
                     }
                     else  {
                            System.out.println("Pa:"+CNT+"  Perturbation : "+PERT.getPertName(i)+" Survived with test "+j);
                            manageLog.writeToLogFile("Perturbation :  "+PERT.getPertName(i)+"\t with PERT value of "+PPP+
                                                "\t Survived with Test : " + TF.getTFname(j));
                     }
                                                                         //      and continue
                }
             }
             else System.out.println("Bypassing tests for : " + PERT.getPertName(i));
      }
} // evaluateTests
```

```
//-----------------------------------------------------------------
// pert Killed? routine
//    This routine runs the original and perturbed programs to determine if there is a difference.  If there
//    is a difference (and the difference persists for numKills times) then the perturbation is considered
//    killed and the routine returns a true.  Otherwise the perturbation survived and the routine returns a
//    false.  The number trys was added to account for the non-deterministic nature of the JAVA prototype.
public static boolean pertKilled(String testFN) throws IOException  {
      boolean RV = true;
      for (int keepTrying=0; keepTrying<numKills; keepTrying++ )  {
                  // need to kill numKills times to be considered a "real" kill
                  // note this calls a batch file to run both versions of the program (perturbed and  unperturbed)
                  //      with the given test file and redirects their outputs to the specified file names ...
              spFun.DosCom("runBoth.bat " + testFN + " " + outRegular + " " + outPerturb);
              int R = compare(outPerturb, outRegular);
              if(R==0)  {                          // if the same, the perturbation survived
                     RV=false;                     //     so return immediatly
                     keepTrying = numKills+1;
              }
              keepTrying++;
      }
      return RV;                                   // if got here, the perturbation was killed numKills times
} // pertKilled

//-----------------------------------------------------------------
// processCommandLine
//     Read the Command line.  Calls readParameters to read in all of the
//    command parameters from the specified setup file.
static int processCommandLine (String [] args) throws  IOException {
      int RV = 0;    // return value for function

      if (args.length == 1)  {                                  // use this parameters file
              File FF = new File(args[0]);
              if (FF.exists()) RV = readParameters(args[0]);
              else  System.out.println("Pert File error:  The parameter file does not exist");
      }
      if(RV==0) {
              System.out.println("Command line error, command should be of the form:");
              System.out.println("    pertTool parameterFile");
              System.out.println("      where parameterFile : is file which contains all of");
              System.out.println("                            the setup parameters?");
      }
      return(RV);
} // processCommandLine
```

```
//---------------------------------------------------------------
//  Read the command parameters in from the setup file.
static int readParameters(String FN)  throws IOException {
     StringTokenizer ST;                                        // used to parse input commands
     BufferedReader in = new BufferedReader(new FileReader(FN)); // start reader
     String line = in.readLine();
     int ptr = 0;
     while (line!=null) {
            if(line.startsWith("TEST_DIRECTORY:"))  {           // get test directory
                   ST = new StringTokenizer(line.substring(15)," \t");
                   testDirectory = ST.nextToken();
                   File FF = new File(testDirectory);
                   if (!FF.exists())  {
                          boolean success = FF.mkdir();
                          if (!success) {
                                 System.out.println("Error -- could not create directory " + testDirectory);
                                 return 0;
                          }
                   }
            }
            if(line.startsWith("  NUMBER_TRIES:"))  {           // Number of tries to kill a perturbation
                   ST = new StringTokenizer(line.substring(15)," \t");
                   maxTriesToKill = Integer.valueOf(ST.nextToken()).intValue();
            }
            if(line.startsWith("      LOG_FILE:"))  {           // Name of log file
                   ST = new StringTokenizer(line.substring(15)," \t");
                   logFileName = ST.nextToken();
            }
            if(line.startsWith("EVALUATE_ONLY?:"))  {           // Set flag to evaluate existing test set
                   ST = new StringTokenizer(line.substring(15)," \t");
                   String TMP = ST.nextToken();
                   if (TMP.startsWith("Y")||TMP.startsWith("y")) evaluateOnly = true;
                   else evaluateOnly = false;
            }
            if(line.startsWith(" VERIFY_PERTS?:"))  {           // Flag which causes inline verification of perts
                   ST = new StringTokenizer(line.substring(15)," \t");
                   String TMP = ST.nextToken();
                   if (TMP.startsWith("Y")||TMP.startsWith("y")) verifyPerts = t rue;
                   else verifyPerts = false;
            }
            if(line.startsWith("  PERT_TO_TEST:"))  {           // List of up to 50 specific perts to test
                   ST = new StringTokenizer(line.substring(15)," \t");
                   pertToTest[ptr++] = ST.nextToken();
                   if (ptr>49) {
                          System.out.println("Pert File error: cannot specify more than 50 individual perts to test");
                          return 0;
                   }
```

```
            }
            if(line.startsWith("  NUMBER_PERTS:"))  {           // Number of pert loops
                    ST = new StringTokenizer(line.substring(15)," \t");
                    noPertLoops = Integer.valueOf(ST.nextToken()).intValue();
            }
            if(line.startsWith("  REPORT_TITLE:"))  {           // Title for log file
                    int P = line.indexOf('/');
                    if (P==-1) P=line.length();
                    line = line.substring(15,P);
                    reportTitle = line.trim();
            }
            if(line.startsWith("PERT_DIRECTORY:"))  {           // Directory to save PERT.DAT file
                    ST = new StringTokenizer(line.substring(15)," \t");
                    pertDir = ST.nextToken() + "/";
            }
            if(line.startsWith(" LIST_OF_PERTS:"))  {           // File where total list of Perts specified
                    ST = new StringTokenizer(line.substring(15)," \t");
                    pertList = ST.nextToken();
            }
            if(line.startsWith(" TEST_SET_STRT:"))  {           // For evaluation of a test set start with this test
                    ST = new StringTokenizer(line.substring(15)," \t");
                    EVALstrt = Integer.valueOf(ST.nextToken()).intValue();
            }
            if(line.startsWith(" TEST_SET_STOP:"))  {           //     ... and  stop with this test (-1 to test all)
                    ST = new StringTokenizer(line.substring(15)," \t");
                    EVALstop = Integer.valueOf(ST.nextToken()).intValue();
            }
            if(line.startsWith("  NUMBER_KILLS:"))  {           // Number of kills before considered a real kill
                    ST = new StringTokenizer(line.substring(15)," \t");
                    numKills = Integer.valueOf(ST.nextToken()).intValue();
            }
            if(line.startsWith("EVALUATE_TESTS:"))  {           // Flag to do Full Test Evaluation
                    ST = new StringTokenizer(line.substring(15)," \t");
                    String TMP = ST.nextToken();
                    if (TMP.startsWith("Y")||TMP.startsWith("y")) {
                            evaluateOnly = true;
                            evaluateTestsBool = true;
                    }
                    else evaluateTestsBool = false;
            }
            line = in.readLine();
      }
      in.close();
      return 1;
} // readParameters
```

```
  //---------------------------------------------------------
  //  This function compares the two output files to determine if they are the same.  Returns a 0 if
  //     they are the same or a 1 if they are different.  Also retuns a -1 if an error is encounterd.
  static int compare(String PrtFN, String RegFN) throws IOException  {
        int RV=0;                                                   // return value

        if ( new File(PrtFN).exists() && new File(RegFN).exists() ) {       // only compare if both files exist
                BufferedReader inPrt = new BufferedReader(new FileReader(PrtFN));
                BufferedReader inReg = new BufferedReader(new FileReader(RegFN));
                String linePrt = inPrt.readLine();                  // read a line in from perturbation output
                String lineReg = inReg.readLine();                  // read a line in from regular output
                if (linePrt==null && lineReg!=null) RV=1;           // if linePrt is null and lineReg isn't return diff
                while ( (linePrt!=null)&&(RV==0) ) {                // stop if at end of pert file or if different
                        if (linePrt.startsWith(" !!! ERROR:"))       {       // This error detected by the perterbation
validation
                                System.out.println(linePrt);        //      function ... display line with error message
                                return -1;                          //      and return a -1
                        }
                        if (linePrt.startsWith("load time") ||      // ignore diffs in these lines
                            linePrt.startsWith("setup time") ||
                            linePrt.startsWith("solve time") ||
                            linePrt.startsWith("Total Time") );             //   and just do nothing
                        else  {
                                if(lineReg==null) RV=1;                     // if lineReg is empty and linePrt isn't, return
diff
                                else if(linePrt.compareTo(lineReg)!=0)
                                        RV=1;                       // if lines are not the same return diff
                        }
                        linePrt = inPrt.readLine();                 // read next lines in
                        lineReg = inReg.readLine();
                }
        }
        else RV = -1;                                               // if both files don't exist return a -1
        return RV;
  } // compare

} // class pertTool
```

```
//-------------------------manageTF-----------------------
//   Richard Stutler
//     Manage the Test files
//     This class includes routines to build the test file arrays
//     as well as build the actual random input test files.
//------------------------------------------------------------------

import java.io.*;
import java.lang.*;
import java.util.*;

public class manageTF   {

       private String directory;                    // directory to store test set in
       private String [] fileName;                  // Name of the file Test file
       private int ptrTF;                           // Poiinter to next Available testset
       public  String tfName="TF.txt";              // root name for the test files

   //-----------------------------------------------------
   //  Class constructor
   //  Builds the Test File arrays
   public manageTF(int max, String D)  throws IOException  {
       File test;
       String FN;
       fileName = new String[max];                  // size the array
       directory = D;
       ptrTF = 0;
       for (int x=0; x<max; x++)  {                 // read in the existing files
              FN = D + "/" + tfName + "." + spFun.NumberToString(x,6,'0');
              if (new File(FN).exists()) {          // if Test file is there, add
                     fileName[ptrTF]=FN;            //      to the array
                     ptrTF++;                       //      and increment the pointer
              }
       }
       System.out.println("Read in " + ptrTF + " test files");
   } // manageTF
```

```
//-----------------------------------------------------
   //  Adds the testset as defined by the File to the test set.
   //    Includes copying the test set to the test set directory.
   public int addTF(String FN)  throws IOException  {
        String TO=directory + "/" + FN + "." + spFun.NumberToString(ptrTF,6,'0');
        spFun.Fcopy(FN,TO);                          // copy the test set to test directory
        fileName[ptrTF]=TO;                          // save the file name
        int RV = ptrTF++;                            // increment the pointer
        return RV;                                   // return the test file number
   } // addTF

   //-----------------------------------------------------
   //  Gets the test set requested and copies it to the requested file.
   public int getTF(int i, String FN)  throws IOException  {
        int RV = spFun.Fcopy(fileName[i],FN);        // copy the test set from test directory
        return RV;                                   // return -1 if failed or 1 if ok
   } // getTF

   //-----------------------------------------------------
   //  Routines to set or get local data.
   public int numTFs()                  { return ptrTF;       }    // Returns the current number of Tests in the set
   public String getTFname(int i)  { return fileName[i]; }         // Returns the name of the specified Test
```

```
//-----------------------------------------------------------------
//   This class builds a test file for the ssn prototype.
//   It is called with the name of the test file to create.
//   It creates a random file within the established parameters.
public void buildRandomTestFile(String FN) throws IOException  {
     boolean mslLoc[];                    // a boolean array ... one for each possible missile location
     String tmp;                          // a temporary string array
     int RND;                             // Random Number Return

     PrintWriter out = new PrintWriter(new BufferedWriter(new FileWriter(FN)));
     mslLoc = new boolean [40];    // initialize the missile boolean array
     for (int i =0; i< 40 ; i++)  mslLoc[i]=spFun.RanBool();

     out.println("# test file : " + ptrTF);
     out.println("");
     out.println("# ****** GLOBAL VALUES ****** ");
     if (spFun.RanBool()) out.println("preferLauncher : canisters");
                     else out.println("preferLauncher : tubes" );
     if (spFun.RanBool()) out.println("preferRedundantRole : readySpare");
                     else out.println("preferRedundantRole : backup");
     tmp = "";
     if (mslLoc[0]) tmp = tmp + "TT1 ";                          // Only announce tubes if missles
     if (mslLoc[1]) tmp = tmp + "TT2 ";                          //      are in them
     if (mslLoc[2]) tmp = tmp + "TT3 ";
     if (mslLoc[3]) tmp = tmp + "TT4" ;
     if (tmp!="") out.println("torpedoTubes : " + tmp);
     if (spFun.RanBool()) out.println("block3PrepTime : " + spFun.RanInt(25,360));
     if (spFun.RanBool()) out.println("block4PrepTime : " + spFun.RanInt(5,36 0));

     out.println("");
     out.println("# ****** MISSILES ****** ");
     for (int i=17 ; i<28 ; i++)  mslLoc[i] = false;             // fix since these locations are not
     for (int i=29 ; i<40 ; i++)  mslLoc[i] = false;             //      included in this version of ssn
     for (int i=0  ; i<40 ; i++)  if(mslLoc[i])  out.println(buildMissile(i));

     out.println("");
     out.println("# ****** TASKS ****** ");
     int NoTsks = spFun.RanInt(1,100);                           // how many tasks do you want to create?
     for (int id = 1 ; id<NoTsks ; id++) {
            out.print("task : " + spFun.NumberToString(id,3,' ') + " : ");
                   // Following is a fix since TacTom is a block 4 and all of the rest
                   //   are block 3 missiles and a task cannot mix missle blocks
            if (spFun.RanInt(1,5)==2) tmp = "TACTOM 1 ";         // 1 in 5 chance that a block 4, TacTom
            else  {                                              //      missle, is chosen for task
                   tmp = "";
                   int p = spFun.RanInt(1,5);                    // get random number of missle types
                   for (int i = 0; i<p ; i++) {                  //    associated with each type and buil
```

```
                        String X = "";                          //    missle string of block 3 types for
                        int t = spFun.RanInt(1,14);             //    each priority
                        if(t < 10) X = "C"+t;
                        if(t==10) X = "CII";
                        if(t==11) X = "CIII";
                        if(t==12) X = "DI";
                        if(t==13) X = "DII";
                        tmp = tmp + X + " " + (i+1) + " ";
                }
        }
        out.print(spFun.FormatString(tmp,44,' ','l') + " : ");
        tmp = "";
        if (spFun.RanBool()) tmp = tmp + "GPS ";            // determine random missile components
        if (spFun.RanBool()) tmp = tmp + "DSMAC ";          //    can be any combination of the three
        if (spFun.RanBool()) tmp = tmp + "SDL";
        out.print(spFun.FormatString(tmp,13,' ','l') + " : ");
        out.print(spFun.NumberToString(spFun.RanInt(0,30),2,' ') + " ");          // earliest launch time
        out.print(spFun.NumberToString(spFun.RanInt(0,90),2,' ') + " ");          // planned launch time
        out.print(spFun.NumberToString(spFun.RanInt(0,130),3, ' ') + " : ");      // latest launch time
        if (spFun.RanBool()) out.print("execute : ");                            // either execute or launch
                        else  out.print("launch  : ");
        int r = spFun.RanInt(0,5);                                                // select a random role
        if (r==0) out.println("primary");
        if (r==1) out.println("callForFire");
        if (r==2) out.println("backup");
        if (r==3) out.println("readySpare");
        if (r==4) out.println("pool");
    }

    out.println("");
    out.println("# ****** MANUAL ALLOCATIONS ****** ");
    int man = spFun.RanInt(0,5);                            // one in 5 chance will generate a manual allocation
    if (man == 1) {                                         // manualAllocation : task mslID [ launcherLocation ] : ...
        out.print("manualAllocation : ");
        int Tsk = spFun.RanInt(1,NoTsks);                   // random task number
        int Loc = spFun.RanInt(0,40);                       // random location
        out.print(spFun.NumberToString(Tsk,3,' ')+ " ");
        out.print(spFun.NumberToString(Loc,3,' ')+ " ");
        if (spFun.RanBool())                                // don't always add location
                out.println(buildLocString(Loc)+ " ");
    }
    out.println("");

    out.close();                                            // close output file
} // buildRandomTestFile
```

```
//---------------------------------------------------------------------------
//   Given a location index, return the string representation
private String buildLocString(int loc) {
     String RV="";
     if(loc==0)  RV = "  TT1" ;
     if(loc==1)  RV = "  TT2" ;
     if(loc==2)  RV = "  TT3" ;
     if(loc==3)  RV = "  TT4" ;
     if(loc==4)  RV = " CLS5" ;
     if(loc==5)  RV = " CLS6" ;
     if(loc==6)  RV = " CLS7" ;
     if(loc==7)  RV = " CLS8" ;
     if(loc==8)  RV = " CLS9" ;
     if(loc==9)  RV = "CLS10" ;
     if(loc==10) RV = "CLS11" ;
     if(loc==11) RV = "CLS12" ;
     if(loc==12) RV = "CLS13" ;
     if(loc==13) RV = "CLS14" ;
     if(loc==14) RV = "CLS15" ;
     if(loc==15) RV = "CLS16" ;
     if(loc==16) RV = "  UA1" ;
     if(loc==17) RV = "  UB2" ;    //  Included but not used for this version of ssn JAVA prototype
     if(loc==18) RV = "  UC3" ;    //       "     "   "    "   "    "     "     "  "   "       "
     if(loc==19) RV = "  UD4" ;    //       "     "   "    "   "    "     "     "  "   "       "
     if(loc==20) RV = "  UE5" ;    //       "     "   "    "   "    "     "     "  "   "       "
     if(loc==21) RV = "  UF6" ;    //       "     "   "    "   "    "     "     "  "   "       "
     if(loc==22) RV = "  UA7" ;    //       "     "   "    "   "    "     "     "  "   "       "
     if(loc==23) RV = "  UB8" ;    //       "     "   "    "   "    "     "     "  "   "       "
     if(loc==24) RV = "  UC9" ;    //       "     "   "    "   "    "     "     "  "   "       "
     if(loc==25) RV = " UD10" ;    //       "     "   "    "   "    "     "     "  "   "       "
     if(loc==26) RV = " UE11" ;    //       "     "   "    "   "    "     "     "  "   "       "
     if(loc==27) RV = " UF12" ;    //       "     "   "    "   "    "     "     "  "   "       "
     if(loc==28) RV = "  LA1" ;
     if(loc==29) RV = "  LB2" ;    //       "     "   "    "   "    "     "     "  "   "       "
     if(loc==30) RV = "  LC3" ;    //       "     "   "    "   "    "     "     "  "   "       "
     if(loc==31) RV = "  LD4" ;    //       "     "   "    "   "    "     "     "  "   "       "
     if(loc==32) RV = "  LE5" ;    //       "     "   "    "   "    "     "     "  "   "       "
     if(loc==33) RV = "  LF6" ;    //       "     "   "    "   "    "     "     "  "   "       "
     if(loc==34) RV = "  LA7" ;    //       "     "   "    "   "    "     "     "  "   "       "
     if(loc==35) RV = "  LB8" ;    //       "     "   "    "   "    "     "     "  "   "       "
     if(loc==36) RV = "  LC9" ;    //       "     "   "    "   "    "     "     "  "   "       "
     if(loc==37) RV = " LD10" ;    //       "     "   "    "   "    "     "     "  "   "       "
     if(loc==38) RV = " LE11" ;    //       "     "   "    "   "    "     "     "  "   "       "
     if(loc==39) RV = " LF12" ;    //       "     "   "    "   "    "     "     "  "   "       "
     return RV;
} // buildLocString
```

```
//-------------------------------------------------------------------------
   //   Build a missle string for the test file - Given the missile id
   private String buildMissile(int id) {
        String RV;
        String tmp;

        RV = "missile : " + spFun.NumberToString(id,3,' ') + " : ";
        RV = RV + buildLocString(id) + " : ";
        int t = spFun.RanInt(0,15)+1;                       // select a random xmids
        tmp = "";
        if(t <10) tmp = "C"+t;
        if(t==10) tmp = "CII";
        if(t==11) tmp = "CIII";
        if(t==12) tmp = "DI";
        if(t==13) tmp = "DII";
        if(t==14) tmp = "TACTOM";
        if(t==15) tmp = "NON_TOMAHAWK";
        RV = RV + spFun.FormatString(tmp,12,' ','l') + " : ";
        tmp = "";                                           // now build random set of components
        if (spFun.RanBool()) {                              // decide if this is desired componet
               if(spFun.RanBool()) tmp = tmp + "GPS ";              //   and if it is, decide if regular
               else  tmp = tmp + "faultedGPS ";             //   or faulted
        }
        if (spFun.RanBool()) {                              // same as above
               if(spFun.RanBool()) tmp = tmp + "DSMAC ";
               else  tmp = tmp + "faultedDSMAC ";
        }
        if (spFun.RanBool()) {                              //   "   "   "
               if(spFun.RanBool()) tmp = tmp + "SDL ";
               else  tmp = tmp + "faultedSDL";
        }
        RV = RV+ spFun.FormatString(tmp,34,' ','l') + " : ";
        RV = RV + spFun.RanInt(1,10) + " : " + spFun.RanInt(15,30) + " : ";
        RV = RV + "2005";                                   // due date year
        RV = RV + spFun.RanInt(1,13);                       //  "    "  month
        RV = RV + spFun.RanInt(1,32);                       //  "    "  day
        return RV;
   } // buildMissile

} // manageTF
```

```
//---------------------------------------------------------------------
//   Richard Stutler
//      Manage the Log file
//      This class includes routines to generate the log file, to
//      add data into it, and to close it.
//---------------------------------------------------------------------

import java.io.*;
import java.lang.*;
import java.util.*;
import java.text.NumberFormat;

public class manageLog  {

  private static PrintWriter pwLogFile;              // Writes to logfile are via PrintWriter Class
  private static boolean logValid;                   // Flag to indicate ok to write to logFile
  private static manageTF   TF;                      // test file class
  private static managePERT PERT;                    // perturbation file class
  private static int initNumTestSets;                // initial number of test sets

   //-----------------------------------------------------------------
   // Open log file for output and print header information
   public static void InitLogFile(String fileName, manageTF T, man agePERT P) throws IOException {
        int i;
        TF = T;
        PERT = P;
        if (fileName.equals(""))  {                  // reset the log file
               logValid = false;
               return;
        }
        else  {
               logValid = true;
               pwLogFile = new PrintWriter(new BufferedWriter(new FileWriter(fileName))) ;  // Writes are via PrintWriter Class
                      // Display header information at top of file
               pwLogFile.println("\t\t\tPertabation Analysis and Testing Tool");
               pwLogFile.println("\t\t\t-----------------------------------");
               pwLogFile.println("");
               pwLogFile.print("Title: " + pertTool.reportTitle);
               pwLogFile.println("\t\t" + (pertTool.startTimeC.get(Calendar.MONTH)+1) + "/" +
                                            pertTool.startTimeC.get(Calendar.DATE) + "/" +
                                            pertTool.startTimeC.get(Calendar.YEAR));   // Display current date
               pwLogFile.println("");
               pwLogFile.println("         Max Number of random inputs requested : " + pertTool.maxTriesToKill);
               pwLogFile.println("                              Perturbation order is : " + pertTool.noPertLoops);
               pwLogFile.println("  Number of kills to be considered a real kill : " + pertTool.numKills);
               pwLogFile.println("");
```

```
            pwLogFile.println("There are " + TF.numTFs() + " existing test sets");
            pwLogFile.println("Saved in Directory: " + pertTool.testDirectory);
            pwLogFile.println("");
            pwLogFile.print("Started perturbation testset generation at: ");
            pwLogFile.print(pertTool.startTimeC.get(Calendar.HOUR) + ":" +
                                            pertTool.startTimeC.get(Calendar.MINUTE) + ":" +
                                            pertTool.startTimeC.get(Calendar.SECOND));  // Dis play Time started
            pwLogFile.println("    . . . . . .");
            initNumTestSets=TF.numTFs();
    }
} // initLogFile

//---------------------------------------------------------------
//  Writes strings to the log file
public static void writeToLogFile(String SS) throws IOException {
    if(logValid)  pwLogFile.println("  " + SS);          // only write if log file is valid
}  // writeToLogFile

//---------------------------------------------------------------
// Saves the final data to the logfi le and closes the file
public static void closeLogFile() throws IOException {
    if(logValid)  {
            pwLogFile.print ("Perturbation analysis Ended at: ");
            pwLogFile.println(pertTool.stopTimeC.get(Calendar.HOUR) + ":" + pertTool.stopTimeC.get(Calendar.MI NUTE) +
                              ":" + pertTool.stopTimeC.get(Calendar.SECOND));     // Display Time ended
            pwLogFile.println("");
            pwLogFile.println("Analysis summary");
            pwLogFile.println("------------------------");
            if (pertTool.evaluateOnly==false) {
                    pwLogFile.println(" Total perturbations: "+ PERT.getNumPerts()    );
                    pwLogFile.println("              Killed: "+ PERT.getNumKilled()   );
                    pwLogFile.println("           Surviving: "+ PERT.getNumSurvived() );
                    double percent = (double)((double)PERT.getNumKilled() /(double)PERT.getNumPerts())*100;
                    pwLogFile.println("Percent Killed: "+ percent + " %");
            }
            else  {
                    pwLogFile.println("Evaluation of test set using Pa = " + pertTool.noPertLoops);
                    pwLogFile.println("      Killed Perturbations: "+ pertTool.killC nt   );
                    pwLogFile.println("   Surviving Perturbations: "+ pertTool.surviveCnt );
                    double percent = (double)((double)pertTool.killCnt/(double)(pertTool.killCnt+pertTool.surviveCnt))*100;
                    pwLogFile.println("Percent Killed: "+ percent + " %");
            }
            pwLogFile.println("");
            pwLogFile.println(" Test Set Size ");
            pwLogFile.println("      Original: "+ initNumTestSets);
            pwLogFile.println("     Resulting: "+ TF.numTFs());
            pwLogFile.println("");
```

```
                pwLogFile.println("Number of Inputs generated: " + pertTool.numInputs);
                  pwLogFile.print("              Elapsed Time: ");
                pwLogFile.println(spFun.MillsecToString(pertTool.stopTimeC.getTimeInMillis()  -
                                                  pertTool.startTimeC.getTimeInMillis()));  // time diff is elapsed time
                pwLogFile.println("");
                pwLogFile.println("Surviving  pertubations");
                pwLogFile.println("----------------------");
                for (int i = 0 ; i< PERT.getNumVars() ; i++)  {
                        if (!PERT.getKilled(i)) {
                                if (PERT.getDoIt(i)) pwLogFile.println("  survived: " + PERT.getPertName(i) );
                        }
                }
                pwLogFile.println("");
                pwLogFile.close();            // close file
        }
    } // closeLogFile

} // class manageLog
```

```
//---------------------------------------------------------------------
//   Richard Stutler
//      Manage the Pertrubation class
//      This class includes routines to read in the perturbation file
//      as well as set the location to perturb in the file.  Also includes
//      routines to access the local variables.
//---------------------------------------------------------------------

import java.io.*;
import java.lang.*;
import java.util.*;

public class managePERT  {

        private int NumVars;                // number of perturbation variables
        private String VarName[];           // name of variable
        private boolean doIt[];             // flag to indicate whether this variable should be perturbed
        private int Lo[];                   // minimum value of a perturbed integer
        private int Hi[];                   // maximum value of a perturbed integer
        private boolean killed[];           // flag to indicate whether this variable has been killed
        private int killedWith[];           // index of test set which killed this perturbation
        private int LocToPert;              // specific variable to perturb this time
        private String FileName;            // name of perturbation list file
        private String PERTfile;            // name of file where specific PERT info saved

   //---------------------------------------------------------------------
   //   Class Constructor
   //   Initializes the Perturbation arrays from the specified perturbation FileName
   //      First reads in the perturbation file to determine the variables to perturbate
   //          and the parameters under which they should be perturbated.
   public managePERT(String FN, String PFN) throws IOException  {
        BufferedReader in;
        String line;
        StringTokenizer ST;                                     // used to parse input commands
        NumVars = 0;
        FileName = FN;
        PERTfile = PFN;                                         // name of temporary data file to pass data to PERT
        File pertFile = new File(FN);
        if (pertFile.exists()) {                                // only do this if file exists
               in = new BufferedReader(new FileReader(FN));     // use BufferedReader to input from file
               line = in.readLine();
               while (line != null) {                           // count number of perturbation variables
                      if(line.startsWith("VAR_NAME:")) NumVars++;
                      line = in.readLine();
                      }
               in.close();
               VarName = new String[NumVars];                   // size the arrays as appropriate
```

```
            Lo = new int[NumVars];
            Hi = new int[NumVars];
            doIt = new boolean[NumVars];
            killed = new boolean[NumVars];
            killedWith = new int[NumVars];
            in = new BufferedReader(new FileReader(FN));        // restart reader
            line = in.readLine();
            int ptr = 0;
            while (line!=null) {                                // scan entire file
                   if(line.startsWith("VAR_NAME:"))  {          // read in the parameters for each variable
                          ST = new StringTokenizer(line.substring(10)," \t");
                          String N = ST.nextToken();            // get variable name
                          if (findName(N)!=-1)                  // see if name already exists
                                 System.out.println("*** ERROR = duplicate perturbation name found ***");
                          VarName[ptr]=N;
                          Lo[ptr]=  Integer.valueOf(ST.nextToken()).intValue();                      // get lo
                          Hi[ptr]=  Integer.valueOf(ST.nextToken()).intValue();                      // get hi
                          if (Integer.valueOf(ST.nextToken()).intValue()==0)  doIt[ptr]=false;       // get doIt flag
                                                                    else doIt[ptr]=true;
                          killed[ptr]=false;                    // initialize killed flag
                          ptr++;
                   }
                   line = in.readLine();                        // get next line in file
            }
            in.close();
     }
} // managePERT
```

```
//-------------------------------------------------------------------------
 //  Sets the variable to perturb in the PERT.DAT file
 public int setLocTest(int loc) throws IOException  {
     String line;
     PrintWriter out = new PrintWriter(new BufferedWriter(new FileWriter(PERTfile)));
     int R = spFun.RanInt(Lo[loc], Hi[loc]);                          // return a random integer between max & min
     out.println("PERT_LOC: " + VarName[loc] + " \t"  + R);           //   and write the line to the PERT.DAT file
     if (pertTool.verifyPerts)
            out.println("VER_PERT: " +"..\\..\\" + pertTool.pertList);
     out.close();
     return R;
 } // setLocTest


 //-------------------------------------------------------------------------
 //  Returns the integer index for the variable with the specified name.
 public int findName(String N)  {
     int RV = -1;                                  // return a -1 if name is not found
     for (int i =0; i<NumVars; i++)  {
            if (N.equals(VarName[i])) RV=i;       // return index if name is equal
     }
     return RV;
 } // findName


 //-------------------------------------------------------------------------
 //  Routines to set or get local data.
 public int getNumVars()                 { return NumVars;         }
 public String getPertName(int ptr)      { return VarName[ptr];    }
 public boolean getDoIt(int ptr)         { return doIt[ptr];       }
 public void setDoIt(int ptr, boolean v) { doIt[ptr] = v;      }
 public boolean getKilled(int ptr)       { return killed[ptr];     }
 public int getKilledWith(int ptr)       { return killedWith[ptr]; }
 public void setKilled(int ptr, int TF)  { killed[ptr]=true;
                                          killedWith[ptr] = TF;   }
```

```
    //-------------------------------------------------------------------------
    //  Returns the total number of variables killed.
    public int getNumKilled()  {
         int RV = 0;
         for (int i = 0 ; i<NumVars ; i++)  {         // look at all of the variables
                if (killed[i]  && doIt[i]) RV++;      // needs to be both included and killed to count
         }
         return RV;
    } // getNumKilled


    //-------------------------------------------------------------------------
    //  Returns the total number of variables survived.
    public int getNumSurvived()  {
         int RV = 0;
         for (int i = 0 ; i<NumVars ; i++)  {         // look at all of the variables
                if (!killed[i] && doIt[i]) RV++;      // needs to be both included and not killed to count
         }
         return RV;
    } // getNumSurvived


    //-------------------------------------------------------------------------
    //  Returns the total number of valid perturable variables.
    public int getNumPerts()  {
         int RV = 0;
         for (int i = 0 ; i<NumVars ; i++)  {         // look at all of the variables
                if (doIt[i]) RV++;                    // needs to be included to count
         }
         return RV;
    } // getNumPerts


} // class managePERT
```

```
//-------------------------------------------------------
//   Richard Stutler
//   Specialized Functions for Java programs
//
//     Several functions are included here -----
//             ReadLine(prompt) : displays a prompt to the screen and
//                     then reads a line of data in from the keyboard.
//             FormatString(target,len) : right justifies the input string
//                     to a new string of length len
//             String NumerToString(num, len, cc) : first converts num to
//                     a string, then right justifies the string to length
//                     len, and then fills in the left blank values with
//                     character cc.
//             Random(Lo, Hi) : generates a random integer between Lo and Hi
//             MillsecToString(t) : converts a long time value t into a formatted
//                     string of the form hh:mm:ss.sss
//             WordWrap(Str, len) : takes Str and inserts a \r\n at the appropriate
//                     places to ensure displayed line length is less than len
//-------------------------------------------------------------------------

import java.io.*;
import java.lang.*;
import java.util.*;
import java.text.NumberFormat;

public class spFun  {

  //-------------------------------------------------------
  // Reads input from the keyboard and prompts the user for data requested.
  static String ReadLine(String prompt) throws IOException  {
        int c;
        if (prompt != "")
                System.out.print(prompt);             // display the prompt
        StringBuffer data = new StringBuffer(80);     // build the string buffer
        c = System.in.read();                         // input to char
        while ((c!=10)&&(c!=13)) {                    // 10 is newline,  13 is carrage return
                data.append((char)c);                 // append to string buffer
                c = System.in.read();
                }                                     // and continue until newlin
        while (System.in.available()>0)
                c = System.in.read();                 // flush input
        return (new String(data.toString()));         // convert char array to string
  } // ReadLine
```

```
//-------------------------------------------------------
// Justifies the given string to the specified length.
static String FormatString(String target,int len, char cc, char justify)    {
      StringBuffer data = new StringBuffer(target);               // create the string buffer
      if (justify == 'r')
            while(data.length() < len)
                  data.insert(0,cc);                            // add spaces to left until length
      if (justify == 'l')
            while(data.length() < len) data.insert(data.length(),cc);
      return (data.toString());                                   // and return as string
} // FormatString


//-------------------------------------------------------
// Converts number to string and right justifies the string
//    to the specified length with leading characters as
//    provided. (usually either ' ' or '0')
static String NumberToString(int num,int len, char cc)  {
      StringBuffer str = new StringBuffer(Integer.toString(num));  // create the string buffer
      while(str.length() < len) str.insert(0,cc);                 // add char to left until length
      return (str.toString());                                    // and return as string
} // NumerToString


//-------------------------------------------------------
// Generates random number between Lo and Hi.
static int RanInt(int Lo, int Hi) {
      int x = (int)(Math.random()*(Hi-Lo)) + Lo;                  // generate random number
      return(x);                                                  // and return it
} // Random


//-------------------------------------------------------
// Generates random Boolean
static boolean RanBool() {
      int x = (int)(Math.random()*(2));                           // generate random number
      if (x==0) return (true); else return(false);
} // RanBool
```

```
//---------------------------------------------------------
// Generates a formatted string of the given milliseconds
static String MillsecToString(long t) {
      int ms = (int)(t % 1000);            // compute millisecs
      t = t / 1000;                        //  and update given time
      int s = (int)(t % 60);               // compute sec
      t = t / 60;                          //  and update given time
      int m = (int)(t % 60);               // compute min
      t = t / 60;                          //  and update given time
      int h = (int)(t);                    // compute hours
      return (NumberToString(h,2, '0') + ":" + NumberToString(m,2, '0') +
              ":" + NumberToString(s,2, '0') + "." +
              NumberToString(ms,3, '0'));  // return string
} // MillsecToString


//---------------------------------------------------------
// Generates a string with line length < len (\r\n inserted)
//    Used as a word wrap function.
static String WordWrap(String Str, int len)  {
      String RetS = "";
      int x;
      while (Str.length() > len){          // loop while Str is longer than len
         x = Str.lastIndexOf(" ",len);     // find last space to left of len
         if (x != -1)  {
              RetS = RetS + Str.substring(0,x) + "\r\n";   // divide string
                                           //   (left half to RetS with cr)
              Str = Str.substring(x);               // and right half back to Str
              }                            // and continue until Str < len
         }
      return(RetS + Str);                  // Return both halfs
} // WordWrap
```

```
//-------------------------------------------------------
// Executes a Dos Command
//   puts the execution window in a minimum mode so that it only shows on the process line
static String DosCom(String S)  {
      int c;
      int exitVal=0;
      StringBuffer RET = new StringBuffer(100000);
      BufferedReader br;
      try  {
         Process proc = Runtime.getRuntime().exec("cmd.exe /c start /min /wait " + S);
         exitVal = proc.waitFor();
         if(exitVal==0)  {
              br = new BufferedReader(new InputStreamReader(proc.getInputStream()));
         }
         else  {
              br = new BufferedReader(new InputStreamReader(proc.getErrorStream()));
         }
         while ( (c = br.read()) != -1){
              RET.append((char)c);
              }
         br.close();
         }
      catch (Throwable t)  {
              t.printStackTrace();
              }
      return(RET.toString());
}  // DosCom


//-------------------------------------------------------
// Generates a formatted string for a double number with the specified precision
public static String Sformat( double value, int minDigits, int maxDigits ) {
      NumberFormat numFormat = NumberFormat.getInstance(); // Number format class to format the values
      numFormat.setMinimumFractionDigits( minDigits );
      numFormat.setMaximumFractionDigits( maxDigits );
      return numFormat.format( value );
} // Sformat
```

```
  //-------------------------------------------------------
  // Copy from one file to another
  public static int Fcopy(String from, String to)  throws IOException   {
        String line;
        int RV = 0;
        File test = new File(from);
        if (!test.exists()) {                              // if from file doesn't exist return -1
               System.out.println("ERROR : Attempt to copy non-existance file : " + from);
                RV = -1;
        }
        else  {                                            // only do this if file exists
               BufferedReader in = new BufferedReader(new FileReader(from));
               PrintWriter out = new PrintWriter(new BufferedWriter(new FileWriter(to)));
               line = in.readLine();                       // read a line in
               while (line != null) {                      // loop until EOF
                      out.println(line);                   // write the line
                      line = in.readLine();                // read next line in
                      }
               in.close();
               out.close();
               RV = 1;                                     // everything is ok so return a 1
        }
        return RV;
  } // Fcopy

} // class spFun
```

APPENDIX D

Test set conversion comparison

This appendix provides a representative sample of the test sets that were used for the testing. For comparison purposes it presents the test sets in the form used for testing the JAVA prototype and also in the form used for testing in the Tomahawk lab. In particular the following tests are presented in both the Java prototype format and the actual Tomahawk system format:

- Derived Test Set 6 – Test 0
- Derived Test Set 6 – Test 15
- Fone's Test– Test 0
- Fone's Test – Test 33

NOTE THIS APPENDIX IS WITHHELD DUE TO CLASSIFICATION

CONTACT THE AUTHOR TO REQUEST THIS INFORMATION.

Derived Test File 6.00 – As used by JAVA Prototype

```
# test file : 6.0

# ****** GLOBAL VALUES ******
preferLauncher : tubes
preferRedundantRole : readySpare
torpedoTubes : TT1 TT2
block3PrepTime : 127
block4PrepTime : 59

# ****** MISSILES ******
missile :   0 :   TT1 : CII     : faultedDSMAC faultedSDL            : 7 : 20 : 20051026
missile :   1 :   TT2 : DII     : faultedGPS DSMAC faultedSDL        : 9 : 23 : 2005425
missile :   7 :  CLS8 : C3      : faultedGPS faultedSDL              : 1 : 17 : 2005119
missile :   8 :  CLS9 : C8      : DSMAC                              : 9 : 18 : 2005326
missile :  11 : CLS12 : TACTOM  : GPS DSMAC                          : 2 : 28 : 2005 77
missile :  12 : CLS13 : DII     : GPS DSMAC SDL                      : 1 : 27 : 20051024
missile :  13 : CLS14 : C9      : GPS DSMAC SDL                      : 1 : 29 : 200599
missile :  16 :   UA1 : C1      : GPS DSMAC faultedSDL               : 9 : 16  : 2005225
missile :  28 :   LA1 : C2      : GPS faultedDSMAC                   : 9 : 19 : 20051019

# ****** TASKS ******
task :   1 : C2 1 C9 2 C5 3 C1 4     : GPS DSMAC SDL : 25 19  49 : execute : callForFire
task :   2 : C3 1 C1 2 C6 3 C6 4     : DSMAC         : 13 26  39 : execute : callForFire
task :   3 : TACTOM 1                :               : 23 61  10 : launch  : backup
task :   4 : C4 1 C1 2 C6 3          : GPS           : 15  2  24 : launch  : callForFire
task :   5 : C6 1 C4 2 C9 3 C7 4     : GPS DSMAC SDL : 16 48  49 : execute : backup
task :   6 : TACTOM 1                : SDL           : 19 88  94 : launch  : pool
task :   7 : DI 1 C1 2 C3 3 DII 4    : GPS SDL       : 12 74  91 : launch  : readySpare
task :   8 : TACTOM 1                : DSMAC SDL     :  9 10  50 : execute : callForFire
task :   9 : TACTOM 1                :               : 11 43  48 : launch  : readySpare
task :  10 : C4 1 C6 2               : SDL           :  3 75  24 : execute : pool
task :  11 : C4 1 DI 2 C9 3          : DSMAC SDL     : 26  3 125 : execute : primary
task :  12 : TACTOM 1                : SDL           : 29 12  46 : launch  : backup
task :  13 : C9 1                    : GPS DSMAC     : 23 54  80 : launch  : callForFire
task :  14 : C6 1 DII 2 C9 3         : SDL           : 13 57  51 : launch  : readySpare
task :  15 : C4 1 C4 2 C6 3          : GPS DSMAC     : 25 84 101 : launch  : pool
task :  16 : TACTOM 1                : SDL           :  3  2  67 : launch  : pool
task :  17 : CIII 1 CII 2 C4 3 C4 4  : DSMAC         : 22 22  92 : execute : callForFire
task :  18 : C3 1 DII 2 C5 3 DI 4    : DSMAC         :  2  7 113 : launch  : pool
task :  19 : TACTOM 1                : DSMAC SDL     :  4 24  80 : execute : primary
task :  20 : TACTOM 1                : GPS SDL       :  8  9  73 : execute : pool
task :  21 : C1 1 C9 2 C7 3          : SDL           : 28 26  55 : launch  : primary
task :  22 : C2 1 C1 2               : GPS DSMAC     : 19 45  47 : launch  : pool

# ****** MANUAL ALLOCATIONS ******
```

Derived Test File 00 – As used by Tomahawk System

```
--------------------------------------------------------------------------------
--  TS6/TF.txt.000000
--------------------------------------------------------------------------------
TaskingRolePriority    ReadySpare
Bk3PrepTime            1
Bk4PrepTime            1
ReloadsExcluded        true
AlgorithmTimeLimit     10
LauncherPreference      LauncherPreferenceTorpedoTube
--------------------------------------------------------------------------------
-- Define all the possible XMIDs. This includes the missile type, the xmid, the
-- missile version (Bk2, Bk3, or Bk4) and the XMID usage (Include, Exclude, Sparingly).
-- The only part of this section that should be touched is the XMID usage column.
--------------------------------------------------------------------------------
LAC_C  C220 4100 9298 95B8 0000 Bk2 Include
LAC_C  C220 4100 9201 9521 0000 Bk2 Include
LAC_C  C220 4100 9A01 9D21 0000 Bk2 Include
LAC_C  C220 4100 9A05 9D25 0000 Bk3 Include
LAC_C  C220 4100 BA01 BD21 0000 Bk2 Include
LAC_C  C220 4100 BA05 BD25 0000 Bk3 Include
LAC_C  C220 4100 9A45 9D65 0000 Bk3 Include
RLAC_C C220 2110 9298 75C8 0000 Bk2 Include
RLAC_C C220 2110 9201 7531 0000 Bk2 Include
RLAC_C C220 2110 9A01 7D31 0000 Bk2 Include
RLAC_C C220 2110 9A05 7D35 0000 Bk3 Include
XLAC_C C220 2108 9298 75C0 0000 Bk2 Include
XLAC_C C220 2108 9201 7529 0000 Bk2 Include
XLAC_C C220 2108 9A01 7D29 0000 Bk2 Include
XLAC_C C220 2108 9A05 7D2D 0000 Bk3 Include
XLAC_C C220 4108 9298 95C0 0000 Bk2 Include
XLAC_C C220 4108 9201 9529 0000 Bk2 Include
XLAC_C C220 4108 9A01 9D29 0000 Bk2 Include
XLAC_C C220 4108 9A05 9D2D 0000 Bk3 Include
LAC_D  C210 4080 9298 9528 0000 Bk2 Include
LAC_D  C210 4080 9201 9491 0000 Bk2 Include
LAC_D  C210 4080 9A01 9C91 0000 Bk2 Include
LAC_D  C210 4080 9A05 9C95 0000 Bk3 Include
LAC_D  C210 4080 BA01 BC91 0000 Bk2 Include
LAC_D  C210 4080 BA05 BC95 0000 Bk3 Include
LAC_D  C210 4080 9A45 9CD5 0000 Bk3 Include
LAC_D  C210 4100 9298 95A8 0000 Bk2 Include
LAC_D  C210 4100 9201 9511 0000 Bk2 Include
LAC_D  C210 4100 9A01 9D11 0000 Bk2 Include
LAC_D  C210 4100 9A05 9D15 0000 Bk3 Include
LAC_D  C210 4100 BA01 BD11 0000 Bk2 Include
LAC_D  C210 4100 BA05 BD15 0000 Bk3 Include
LAC_D  C210 4100 9A45 9D55 0000 Bk3 Include
XLAC_D C210 4188 9298 9630 0000 Bk2 Include
XLAC_D C210 4188 9201 9599 0000 Bk2 Include
XLAC_D C210 4188 9A01 9D99 0000 Bk2 Include
XLAC_D C210 4188 9A05 9D9D 0000 Bk3 Include
XLAC_D C210 2188 9298 7630 0000 Bk2 Include
XLAC_D C210 2188 9201 7599 0000 Bk2 Include
XLAC_D C210 2188 9A01 7D99 0000 Bk2 Include
XLAC_D C210 2188 9A05 7D9D 0000 Bk3 Include
XLAC_D C210 2208 9298 76B0 0000 Bk2 Include
XLAC_D C210 2208 9201 7619 0000 Bk2 Include
XLAC_D C210 2208 9A01 7E19 0000 Bk2 Include
XLAC_D C210 2208 9A05 7E1D 0000 Bk3 Include
LAC_E  C308 4080 0C01 2021 2FAA Bk4 Include
LAC_E  C308 4080 2C01 2021 4FAA Bk4 Include
XLAC_E C308 2088 0C01 2021 0FB2 Bk4 Include
XLAC_E C308 2088 2C01 2021 2FB2 Bk4 Include
XLAC_E C308 4088 0C01 2021 2FB2 Bk4 Include
XLAC_E C308 4088 2C01 2021 4FB2 Bk4 Include
```

```
-------------------------------------------------------------------------------
-- The missile sequence number, sdl required, bdii required, TLD, mission id, use state
-- (Execute, LaunchPlan, HoldFire), task role (Primary, ReadySpare, Backup, CallForFire,
-- Pooled), [times in minutes] launch time earliest, launch time planned, launch time
-- latest, missile type, optional specific tasking (tail number, xmid, or cell id)
-------------------------------------------------------------------------------
MSN 1  true  false TL001 048-001-0100 Execute    CallForFire @+25 @+19 @+49  LAC_C
MSN 2  false false TL001 048-001-0101 Execute    CallForFire @+13 @+26 @+39  LAC_C
MSN 3  false false TL001 048-001-0102 LaunchPlan Backup      @+23 @+61 @+10  LAC_E
MSN 4  false false TL001 048-001-0103 LaunchPlan CallForFire @+15 @+2  @+24  XLAC_C
MSN 5  true  false TL001 048-001-0104 Execute    Backup      @+16 @+48 @+49  LAC_D
MSN 6  true  false TL001 048-001-0105 LaunchPlan Pooled      @+19 @+88 @+94  LAC_E
MSN 7  true  false TL001 048-001-0106 LaunchPlan ReadySpare  @+12 @+74 @+91  XLAC_D
MSN 8  true  false TL001 048-001-0107 Execute    CallForFire @+9  @+10 @+50  LAC_E
MSN 9  false false TL001 048-001-0108 LaunchPlan ReadySpare  @+11 @+43 @+48  LAC_E
MSN 10 true  false TL001 048-001-0109 Execute    Pooled      @+3  @+75 @+24  XLAC_C
MSN 11 true  false TL001 048-001-0110 Execute    Primary     @+26 @+3  @+125 XLAC_C
MSN 12 true  false TL001 048-001-0111 LaunchPlan Backup      @+29 @+12 @+46  LAC_E
MSN 13 false false TL001 048-001-0112 LaunchPlan CallForFire @+23 @+54 @+80  XLAC_D
MSN 14 true  false TL001 048-001-0113 LaunchPlan ReadySpare  @+13 @+57 @+51  LAC_D
MSN 15 false false TL001 048-001-0114 LaunchPlan Pooled      @+25 @+84 @+101 XLAC_C
MSN 16 true  false TL001 048-001-0115 LaunchPlan Pooled      @+3  @+2  @+67  LAC_E
MSN 17 false false TL001 048-001-0116 Execute    CallForFire @+22 @+22 @+92  LAC_C
MSN 18 false false TL001 048-001-0117 LaunchPlan Pooled      @+2  @+7  @+113 LAC_C
MSN 19 true  false TL001 048-001-0118 Execute    Primary     @+4  @+24 @+80  LAC_E
MSN 20 true  false TL001 048-001-0119 Execute    Pooled      @+8  @+9  @+73  LAC_E
MSN 21 true  false TL001 048-001-0120 LaunchPlan Primary     @+28 @+26 @+55  LAC_C
MSN 22 false false TL001 048-001-0121 LaunchPlan Pooled      @+19 @+45 @+47  LAC_C

-------------------------------------------------------------------------------
-- The missile inventory. This includes the cell id, xmid, dsmac capability, gps
-- capability, sdl capability, missile status (available, inop, reload, or selected),
-- and optionally, the plan the missile is paired to (e.g., P1) and the missile sequence
-- number.
-------------------------------------------------------------------------------
TT1    C210 4100 9A45 9D55 0000      false  false  false  Available
TT2    C210 2208 9A05 7E1D 0000      true   false  false  Available
VT8    C220 4100 9298 95B8 0000      false  false  false  Available
VT9    C210 4100 9A05 9D15 0000      true   false  false  Available
VT12   C308 4080 0C01 2021 2FAA      true   true   false  Available
VT13   C210 2208 9A05 7E1D 0000      true   true   true   Available
VT14   C210 4188 9A05 9D9D 0000      true   true   true   Available
A1     C220 4100 9A05 9D25 0000      true   true   false  Available
B1     C220 4100 BA05 BD25 0000      false  true   false  Available

-------------------------------------------------------------------------------
-- Define the missions. This includes mission id, dsmac required, gps
-- required, and the xmids (separated by commas) comprising the m3 list.
-------------------------------------------------------------------------------
048-001-0100 true  true  C220 4100 BA05 BD25 0000 , C210 4188 9A05 9D9D 0000 ,
                         C220 4108 9A05 9D2D 0000 , C220 4100 9A05 9D25 0000
048-001-0101 true  false C220 4100 9298 95B8 0000 , C220 4100 9A05 9D25 0000 ,
                         C210 4080 9A05 9C95 0000 , C210 4080 9A05 9C95 0000
048-001-0102 false false C308 4080 0C01 2021 2FAA
048-001-0103 false true  C220 2108 9A05 7D2D 0000 , C220 4100 9A05 9D25 0000 ,
                         C210 4080 9A05 9C95 0000
048-001-0104 true  true  C210 4080 9A05 9C95 0000 , C220 2108 9A05 7D2D 0000 ,
                         C210 4188 9A05 9D9D 0000 , C210 4080 9A45 9CD5 0000
048-001-0105 false false C308 4080 0C01 2021 2FAA
048-001-0106 false true  C210 2188 9A05 7D9D 0000 , C220 4100 9A05 9D25 0000 ,
                         C220 4100 9298 95B8 0000 , C210 2208 9A05 7E1D 0000
048-001-0107 true  false C308 4080 0C01 2021 2FAA
048-001-0108 false false C308 4080 0C01 2021 2FAA
048-001-0109 false false C220 2108 9A05 7D2D 0000 , C210 4080 9A05 9C95 0000
048-001-0110 true  false C220 2108 9A05 7D2D 0000 , C210 2188 9A05 7D9D 0000 ,
                         C210 4188 9A05 9D9D 0000
```

```
048-001-0111 false false C308 4080 0C01 2021 2FAA
048-001-0112 true  true  C210 4188 9A05 9D9D 0000
048-001-0113 false false C210 4080 9A05 9C95 0000 , C210 2208 9A05 7E1D 0000 ,
                         C210 4188 9A05 9D9D 0000
048-001-0114 true  true  C220 2108 9A05 7D2D 0000 , C220 2108 9A05 7D2D 0000 ,
                         C210 4080 9A05 9C95 0000
048-001-0115 false false C308 4080 0C01 2021 2FAA
048-001-0116 true  false C220 4100 9A45 9D65 0000 , C210 410 0 9A45 9D55 0000 ,
                         C220 2108 9A05 7D2D 0000 , C220 2108 9A05 7D2D 0000
048-001-0117 true  false C220 4100 9298 95B8 0000 , C210 2208 9A05 7E1D 0000 ,
                         C220 4108 9A05 9D2D 0000 , C210 2188 9A05 7D9D 0000
048-001-0118 true  false C308 4080 0C01 2021 2FAA
048-001-0119 false true  C308 4080 0C01 2021 2FAA
048-001-0120 false false C220 4100 9A05 9D25 0000 , C210 4188 9A05 9D9D 0000 ,
                         C210 4080 9A45 9CD5 0000
048-001-0121 true  true  C220 4100 BA05 BD25 0000 , C220 4100 9A05 9 D25 0000
```

Derived Test File 15 – As used by JAVA Prototype

```
# test file : 6.15

# ****** GLOBAL VALUES ******
preferLauncher : canisters
preferRedundantRole : backup
torpedoTubes : TT1 TT4

# ****** MISSILES ******
missile :  0 :   TT1 : TACTOM       : DSMAC                    : 6 : 24 : 20051111
missile :  3 :   TT4 : C1           : faultedGPS faultedDSMAC  : 5 : 24 : 200549
missile :  6 :  CLS7 : DII          : faultedGPS               : 8 : 27 : 2005821
missile :  9 : CLS10 : C6           :                          : 4 : 20 : 20051012
missile : 10 : CLS11 : C9           : DSMAC SDL                : 8 : 25 : 2005127
missile : 12 : CLS13 : NON_TOMAHAWK : faultedSDL               : 9 : 15 : 200588
missile : 13 : CLS14 : C5           : faultedDSMAC faultedSDL  : 7 : 17 : 2005418
missile : 14 : CLS15 : C9           : faultedDSMAC faultedSDL  : 7 : 21 : 2005822
missile : 16 :   UA1 : C4           : DSMAC                    : 8 : 21 : 200569
missile : 28 :   LA1 : C7           : faultedGPS               : 1 :  17 : 20051227

# ****** TASKS ******
task :   1 : TACTOM 1               : GPS DSMAC SDL : 17 24  55 : execute : pool
task :   2 : C8 1                   : GPS DSMAC SDL : 23 68  34 : launch  : callForFire
task :   3 : C3 1 C4 2              : GPS DSMAC SDL : 26 30  66 : launch  : primary
task :   4 : C6 1 C8 2 CII 3 C1 4   : GPS            :  6 89  74 : execute : primary
task :   5 : CIII 1 C3 2            : GPS SDL        :  0 32 116 : execute : pool
task :   6 : C8 1 C2 2 C6 3         : GPS SDL        : 12 55  21 : execute : callForFire
task :   7 : C3 1 C5 2 C7 3 C7 4    : DSMAC          : 13 56  77 : execute : callForFire
task :   8 : CIII 1                 :                :  3 42  30 : launch  : backup
task :   9 : TACTOM 1               :                : 15 38 111 : execute : readySpare
task :  10 : TACTOM 1               :                : 28 41  33 : execute : pool
task :  11 : C3 1 DII 2 CII 3       : GPS            :  8 36 121 : launch  : callForFire
task :  12 : C3 1 C4 2              : DSMAC SDL      :  3 59  86 : execute : pool
task :  13 : DI 1                   : GPS DSMAC      : 26 65  13 : launch  : primary
task :  14 : C8 1 C4 2              : GPS SDL        :  5 38 120 : execute : callForFire
task :  15 : TACTOM 1               : GPS DSMAC      :  3 53  79 : execute : backup
task :  16 : TACTOM 1               : GPS DSMAC SDL  :  5 16  55 : launch  : primary
task :  17 : C1 1 C5 2 C6 3         : DSMAC SDL      : 15 43  12 : execute : pool

# ****** MANUAL ALLOCATIONS ******
```

Derived Test File 15 – As used by Tomahawk System

```
--------------------------------------------------------------------------------
--  TS6/TF.txt.000015
--------------------------------------------------------------------------------
TaskingRolePriority    Backup
Bk3PrepTime            1
Bk4PrepTime            1
ReloadsExcluded        true
AlgorithmTimeLimit     10
LauncherPreference      LauncherPreferenceVerticalTube
--------------------------------------------------------------------------------
-- Define all the possible XMIDs. This includes the missile type, the xmid, the
-- missile version (Bk2, Bk3, or Bk4) and the XMID usage (Include, Exclude, Sparingly).
-- The only part of this section that should be touched is the XMID usage column.
--------------------------------------------------------------------------------
LAC_C  C220 4100 9298 95B8 0000 Bk2 Include
LAC_C  C220 4100 9201 9521 0000 Bk2 Include
LAC_C  C220 4100 9A01 9D21 0000 Bk2 Include
LAC_C  C220 4100 9A05 9D25 0000 Bk3 Include
LAC_C  C220 4100 BA01 BD21 0000 Bk2 Include
LAC_C  C220 4100 BA05 BD25 0000 Bk3 Include
LAC_C  C220 4100 9A45 9D65 0000 Bk3 Include
RLAC_C C220 2110 9298 75C8 0000 Bk2 Include
RLAC_C C220 2110 9201 7531 0000 Bk2 Include
RLAC_C C220 2110 9A01 7D31 0000 Bk2 Include
RLAC_C C220 2110 9A05 7D35 0000 Bk3 Include
XLAC_C C220 2108 9298 75C0 0000 Bk2 Include
XLAC_C C220 2108 9201 7529 0000 Bk2 Include
XLAC_C C220 2108 9A01 7D29 0000 Bk2 Include
XLAC_C C220 2108 9A05 7D2D 0000 Bk3 Include
XLAC_C C220 4108 9298 95C0 0000 Bk2 Include
XLAC_C C220 4108 9201 9529 0000 Bk2 Include
XLAC_C C220 4108 9A01 9D29 0000 Bk2 Include
XLAC_C C220 4108 9A05 9D2D 0000 Bk3 Include
LAC_D  C210 4080 9298 9528 0000 Bk2 Include
LAC_D  C210 4080 9201 9491 0000 Bk2 Include
LAC_D  C210 4080 9A01 9C91 0000 Bk2 Include
LAC_D  C210 4080 9A05 9C95 0000 Bk3 Include
LAC_D  C210 4080 BA01 BC91 0000 Bk2 Include
LAC_D  C210 4080 BA05 BC95 0000 Bk3 Include
LAC_D  C210 4080 9A45 9CD5 0000 Bk3 Include
LAC_D  C210 4100 9298 95A8 0000 Bk2 Include
LAC_D  C210 4100 9201 9511 0000 Bk2 Include
LAC_D  C210 4100 9A01 9D11 0000 Bk2 Include
LAC_D  C210 4100 9A05 9D15 0000 Bk3 Include
LAC_D  C210 4100 BA01 BD11 0000 Bk2 Include
LAC_D  C210 4100 BA05 BD15 0000 Bk3 Include
LAC_D  C210 4100 9A45 9D55 0000 Bk3 Include
XLAC_D C210 4188 9298 9630 0000 Bk2 Include
XLAC_D C210 4188 9201 9599 0000 Bk2 Include
XLAC_D C210 4188 9A01 9D99 0000 Bk2 Include
XLAC_D C210 4188 9A05 9D9D 0000 Bk3 Include
XLAC_D C210 2188 9298 7630 0000 Bk2 Include
XLAC_D C210 2188 9201 7599 0000 Bk2 Include
XLAC_D C210 2188 9A01 7D99 0000 Bk2 Include
XLAC_D C210 2188 9A05 7D9D 0000 Bk3 Include
XLAC_D C210 2208 9298 76B0 0000 Bk2 Include
XLAC_D C210 2208 9201 7619 0000 Bk2 Include
XLAC_D C210 2208 9A01 7E19 0000 Bk2 Include
XLAC_D C210 2208 9A05 7E1D 0000 Bk3 Include
LAC_E  C308 4080 0C01 2021 2FAA Bk4 Include
LAC_E  C308 4080 2C01 2021 4FAA Bk4 Include
XLAC_E C308 2088 0C01 2021 0FB2 Bk4 Include
XLAC_E C308 2088 2C01 2021 2FB2 Bk4 Include
XLAC_E C308 4088 0C01 2021 2FB2 Bk4 Include
XLAC_E C308 4088 2C01 2021 4FB2 Bk4 Include
```

```
-------------------------------------------------------------------------------
-- The missile sequence number, sdl required, bdii required, TLD, mission id, use state
-- (Execute, LaunchPlan, HoldFire), task role (Primary, ReadySpare, Backup, CallForFire,
-- Pooled), [times in minutes] launch time earliest, launch time planned, launch time
-- latest, missile type, optional specific tasking (tail number, xmid, or cell id)
-------------------------------------------------------------------------------
MSN 1  true  false TL001 048-001-0100 Execute    Pooled      @+17 @+24 @+55  LAC_E
MSN 2  true  false TL001 048-001-0101 LaunchPlan CallForFire @+23 @+68 @+34  LAC_D
MSN 3  true  false TL001 048-001-0102 LaunchPlan Primary     @+26 @+30 @+66  LAC_C
MSN 4  false false TL001 048-001-0103 Execute    Primary     @+6  @+89 @+74  LAC_D
MSN 5  true  false TL001 048-001-0104 Execute    Pooled      @+0  @+32 @+116 LAC_C
MSN 6  true  false TL001 048-001-0105 Execute    CallForFire @+12 @+55 @+21  LAC_D
MSN 7  false false TL001 048-001-0106 Execute    CallForFire @+13 @+56 @+77  LAC_C
MSN 8  false false TL001 048-001-0107 LaunchPlan Backup      @+3  @+42 @+30  LAC_C
MSN 9  false false TL001 048-001-0108 Execute    ReadySpare  @+15 @+38 @+111 LAC_E
MSN 10 false false TL001 048-001-0109 Execute    Pooled      @+28 @+41 @+33  LAC_E
MSN 11 false false TL001 048-001-0110 LaunchPlan CallForFire @+8  @+36 @+121 LAC_C
MSN 12 true  false TL001 048-001-0111 Execute    Pooled      @+3  @+59 @+86  LAC_C
MSN 13 false false TL001 048-001-0112 LaunchPlan Primary     @+26 @+65 @+13  XLAC_D
MSN 14 true  false TL001 048-001-0113 Execute    CallForFire @+5  @+38 @+120 LAC_D
MSN 15 false false TL001 048-001-0114 Execute    Backup      @+3  @+53 @+79  LAC_E
MSN 16 true  false TL001 048-001-0115 LaunchPlan Primary     @+5  @+16 @+55  LAC_E
MSN 17 true  false TL001 048-001-0116 Execute    Pooled      @+15 @+43 @+12  LAC_C

-------------------------------------------------------------------------------
-- The missile inventory. This includes the cell id, xmid, dsmac capability, gps
-- capability, sdl capability, missile status (available, inop, reload, or selected),
-- and optionally, the plan the missile is paired to (e.g., P1) and the missile sequence
-- number.
-------------------------------------------------------------------------------
TT1    C308 4080 0C01 2021 2FAA      true   false  false  Available
TT4    C220 4100 9A05 9D25 0000      false  false  false  Available
VT7    C210 2208 9A05 7E1D 0000      false  false  false  Available
VT10   C210 4080 9A05 9C95 0000      false  false  false  Available
VT11   C210 4188 9A05 9D9D 0000      true   false  true   Available
VT13   C220 2110 9A05 7D35 0000      false  false  false  Available
VT14   C220 4108 9A05 9D2D 0000      false  false  false  Available
VT15   C210 4188 9A05 9D9D 0000      false  false  false  Available
A1     C220 2108 9A05 7D2D 0000      true   false  false  Available
B1     C210 4080 9A45 9CD5 0000      false  false  false  Available

-------------------------------------------------------------------------------
-- Define the missions. This includes mission id, dsmac requir ed, gps
-- required, and the xmids (separated by commas) comprising the m3 list.
-------------------------------------------------------------------------------
048-001-0100 true  true  C308 4080 0C01 2021 2FAA
048-001-0101 true  true  C210 4100 9A05 9D15 0000
048-001-0102 true  true  C220 4100 9298 95B8 0000 , C220 2108 9A05 7D2D 0000
048-001-0103 false true  C210 4080 9A05 9C95 0000 , C210 4100 9A05 9D15 0000 ,
                         C210 4100 9A45 9D55 0000 , C220 4100 9A05 9D25 0000
048-001-0104 false true  C220 4100 9A45 9D65 0000 , C220 4100 9298 95B8 0000
048-001-0105 false true  C210 4100 9A05 9D15 0000 , C220 4100 BA05 BD25 0000 ,
               C210 4080 9A05 9C95 0000
048-001-0106 true  false C220 4100 9298 95B8 0000 , C220 4108 9A05 9D2D 0000 ,
                         C210 4080 9A45 9CD5 0000 , C210 4080 9A45 9CD5 0000
048-001-0107 false false C220 4100 9A45 9D65 0000
048-001-0108 false false C308 4080 0C01 2021 2FAA
048-001-0109 false false C308 4080 0C01 2021 2FAA
048-001-0110 false true  C220 4100 9298 95B8 0000 , C210 2208 9A05 7E1D 0000 ,
                         C210 4100 9A45 9D55 0000
048-001-0111 true  false C220 4100 9298 95B8 0000 , C220 2108 9A05 7D2D 0000
048-001-0112 true  true  C210 2188 9A05 7D9D 0000
048-001-0113 false true  C210 4100 9A05 9D15 0000 , C220 2108 9A05 7D2D  0000
048-001-0114 true  true  C308 4080 0C01 2021 2FAA
048-001-0115 true  true  C308 4080 0C01 2021 2FAA
048-001-0116 true  false C220 4100 9A05 9D25 0000 , C220 4108 9A05 9D2D 0000 ,
                          C210 4080 9A05 9C95 0000
```

FONES Test File 00 – As used by JAVA Prototype

```
#FONES Test File 2.1

#########  GLOBAL VALUES

preferLauncher : tubes
preferRedundantRole : readySpare
block3prepTime : 40
block4prepTime : 15
torpedoTubes : TT1 TT2 TT3 TT4

#########  MISSILE INVENTORY
missile : 17 : LA1   : C3 : DSMAC GPS SDL : 3 : 20 : 20050428

#########  TASKS
task :  1 : C3 1 C1 2 : DSMAC GPS : 160 165 170 : execute : primary
task :  2 : C3 1 C1 2 : DSMAC GPS : 260 265 270 : execute : primary
task :  3 : C3 1 C1 2 : DSMAC GPS : 360 365 370 : execute : readySpare
task :  4 : C3 1 C1 2 : DSMAC GPS : 460 465 470 : execute : backup
task :  5 : C3 1 C1 2 : DSMAC GPS : 560 565 570 : launch  : primary
task :  6 : C3 1 C1 2 : DSMAC GPS : 660 665 670 : launch  : primary
task :  7 : C3 1 C1 2 : DSMAC GPS : 760 765 770 : launch   : readySpare
task :  8 : C3 1 C1 2 : DSMAC GPS : 860 865 870 : launch  : backup
```

FONES Test File 00 – As used by Tomahawk System

```
--------------------------------------------------------------------------------
--  FonesTS/TF.txt.000000
--------------------------------------------------------------------------------
TaskingRolePriority    ReadySpare
Bk3PrepTime            1
Bk4PrepTime            1
ReloadsExcluded        true
AlgorithmTimeLimit     10
LauncherPreference      LauncherPreferenceTorpedoTube
--------------------------------------------------------------------------------
-- Define all the possible XMIDs. This includes the missile type, the xmid, the
-- missile version (Bk2, Bk3, or Bk4) and the XMID usage (Include, Exclude, Sparingly).
-- The only part of this section that should be touched is the XMID usage column.
--------------------------------------------------------------------------------
LAC_C  C220 4100 9298 95B8 0000 Bk2 Include
LAC_C  C220 4100 9201 9521 0000 Bk2 Include
LAC_C  C220 4100 9A01 9D21 0000 Bk2 Include
LAC_C  C220 4100 9A05 9D25 0000 Bk3 Include
LAC_C  C220 4100 BA01 BD21 0000 Bk2 Include
LAC_C  C220 4100 BA05 BD25 0000 Bk3 Include
LAC_C  C220 4100 9A45 9D65 0000 Bk3 Include
RLAC_C C220 2110 9298 75C8 0000 Bk2 Include
RLAC_C C220 2110 9201 7531 0000 Bk2 Include
RLAC_C C220 2110 9A01 7D31 0000 Bk2 Include
RLAC_C C220 2110 9A05 7D35 0000 Bk3 Include
XLAC_C C220 2108 9298 75C0 0000 Bk2 Include
XLAC_C C220 2108 9201 7529 0000 Bk2 Include
XLAC_C C220 2108 9A01 7D29 0000 Bk2 Include
XLAC_C C220 2108 9A05 7D2D 0000 Bk3 Include
XLAC_C C220 4108 9298 95C0 0000 Bk2 Include
XLAC_C C220 4108 9201 9529 0000 Bk2 Include
XLAC_C C220 4108 9A01 9D29 0000 Bk2 Include
XLAC_C C220 4108 9A05 9D2D 0000 Bk3 Include
LAC_D  C210 4080 9298 9528 0000 Bk2 Include
LAC_D  C210 4080 9201 9491 0000 Bk2 Include
LAC_D  C210 4080 9A01 9C91 0000 Bk2 Include
LAC_D  C210 4080 9A05 9C95 0000 Bk3 Include
LAC_D  C210 4080 BA01 BC91 0000 Bk2 Include
LAC_D  C210 4080 BA05 BC95 0000 Bk3 Include
LAC_D  C210 4080 9A45 9CD5 0000 Bk3 Include
LAC_D  C210 4100 9298 95A8 0000 Bk2 Include
LAC_D  C210 4100 9201 9511 0000 Bk2 Include
LAC_D  C210 4100 9A01 9D11 0000 Bk2 Include
LAC_D  C210 4100 9A05 9D15 0000 Bk3 Include
LAC_D  C210 4100 BA01 BD11 0000 Bk2 Include
LAC_D  C210 4100 BA05 BD15 0000 Bk3 Include
LAC_D  C210 4100 9A45 9D55 0000 Bk3 Include
XLAC_D C210 4188 9298 9630 0000 Bk2 Include
XLAC_D C210 4188 9201 9599 0000 Bk2 Include
XLAC_D C210 4188 9A01 9D99 0000 Bk2 Include
XLAC_D C210 4188 9A05 9D9D 0000 Bk3 Include
XLAC_D C210 2188 9298 7630 0000 Bk2 Include
XLAC_D C210 2188 9201 7599 0000 Bk2 Include
XLAC_D C210 2188 9A01 7D99 0000 Bk2 Include
XLAC_D C210 2188 9A05 7D9D 0000 Bk3 Include
XLAC_D C210 2208 9298 76B0 0000 Bk2 Include
XLAC_D C210 2208 9201 7619 0000 Bk2 Include
XLAC_D C210 2208 9A01 7E19 0000 Bk2 Include
XLAC_D C210 2208 9A05 7E1D 0000 Bk3 Include
LAC_E  C308 4080 0C01 2021 2FAA Bk4 Include
LAC_E  C308 4080 2C01 2021 4FAA Bk4 Include
XLAC_E C308 2088 0C01 2021 0FB2 Bk4 Include
XLAC_E C308 2088 2C01 2021 2FB2 Bk4 Include
XLAC_E C308 4088 0C01 2021 2FB2 Bk4 Include
XLAC_E C308 4088 2C01 2021 4FB2 Bk4 Include
```

```
-------------------------------------------------------------------------------
-- The missile sequence number, sdl required, bdii required, TLD, mission id, use state
-- (Execute, LaunchPlan, HoldFire), task role (Primary, ReadySpare, Backup, CallForFire,
-- Pooled), [times in minutes] launch time earliest, launch time planned, launch time
-- latest, missile type, optional specific tasking (tail number, xmid, or cell  id)
-------------------------------------------------------------------------------
MSN 1 false false TL001 048-001-0100 Execute    Primary    @+160 @+165 @+170 LAC_C
MSN 2 false false TL001 048-001-0101 Execute    Primary    @+260 @+265 @+270 LAC_C
MSN 3 false false TL001 048-001-0102 Execute    ReadySpare @+360 @+365 @+370 LAC_C
MSN 4 false false TL001 048-001-0103 Execute    Backup     @+460 @+465 @+470 LAC_C
MSN 5 false false TL001 048-001-0104 LaunchPlan Primary    @+560 @+565 @+570 LAC_C
MSN 6 false false TL001 048-001-0105 LaunchPlan Primary    @+660 @+665 @+670 LAC_C
MSN 7 false false TL001 048-001-0106 LaunchPlan ReadySpare @+760 @+765 @+770 LAC_C
MSN 8 false false TL001 048-001-0107 LaunchPlan Backup     @+860 @+865 @+870 LAC_C

-------------------------------------------------------------------------------
-- The missile inventory. This includes the cell id, xmid, dsmac capability, gps
-- capability, sdl capability, missile status (available, inop, reload, or selected),
-- and optionally, the plan the missile is paired to (e.g., P1) and the missile sequence
-- number.
-------------------------------------------------------------------------------
B1      C220 4100 9298 95B8 0000      true    true    true    Available

-------------------------------------------------------------------------------
-- Define the missions. This includes mission id, dsmac required, gps
-- required, and the xmids (separated by commas) comprising the m3 list.
-------------------------------------------------------------------------------
048-001-0100 true true C220 4100 9298 95B8 0000 , C220 4100 9A05 9D25 0000
048-001-0101 true true C220 4100 9298 95B8 0000 , C220 4100 9A05 9D25 0000
048-001-0102 true true C220 4100 9298 95B8 0000 , C220 4100 9A05 9D25 0000
048-001-0103 true true C220 4100 9298 95B8 0000 , C220 4100 9A05 9D25 0000
048-001-0104 true true C220 4100 9298 95B8 0000 , C220 4100 9A05 9D25 0000
048-001-0105 true true C220 4100 9298 95B8 0000 , C220 4100 9A05 9D25 0000
048-001-0106 true true C220 4100 9298 95B8 0000 , C220 4100 9A05 9D25 0000
048-001-0107 true true C220 4100 9298 95B8 0000 , C220 4100 9A05 9D25 0000
```

FONES Test File 33 – As used by JAVA Prototype

```
#Fones Test File 2.34

#########  GLOBAL VALUES

preferLauncher : tubes
preferRedundantRole : readySpare
block3prepTime : 40
block4prepTime : 15
torpedoTubes : TT1 TT2 TT3 TT4

#########  MISSILE INVENTORY
missile : 21 : LA1 : C3 : DSMAC GPS SDL : 3 : 20 : 20050428
missile : 22 : LA1 : C3 : DSMAC GPS SDL : 3 : 20 : 20050428
missile : 23 : UA1 : C3 : DSMAC GPS SDL : 3 : 20 : 20050428
missile : 24 : UA1 : C3 : DSMAC GPS SDL : 3 : 20 : 20050428

#########  TASKS
task :  1 : C3 1 : DSMAC GPS : 160  165  170 : execute : primary
task :  2 : C3 1 : DSMAC GPS : 160  165  170 : execute : primary
task :  3 : C3 1 : DSMAC GPS : 160  165  170 : execute : primary
task :  4 : C3 1 : DSMAC GPS : 160  165  170 : execute : primary
task :  5 : C3 1 : DSMAC GPS : 160  165  170 : execute : primary
```

FONES Test File 33 – As used by Tomahawk System

```
--------------------------------------------------------------------------------
--  FonesTS/TF.txt.000033
--------------------------------------------------------------------------------
TaskingRolePriority    ReadySpare
Bk3PrepTime            1
Bk4PrepTime            1
ReloadsExcluded        true
AlgorithmTimeLimit     10
LauncherPreference     LauncherPreferenceTorpedoTube
--------------------------------------------------------------------------------
-- Define all the possible XMIDs. This includes the missile type, the xmid, the
-- missile version (Bk2, Bk3, or Bk4) and the XMID usage (Include, Exclude, Sparingly).
-- The only part of this section that should be touched is the XMID usage column.
--------------------------------------------------------------------------------
LAC_C  C220 4100 9298 95B8 0000 Bk2 Include
LAC_C  C220 4100 9201 9521 0000 Bk2 Include
LAC_C  C220 4100 9A01 9D21 0000 Bk2 Include
LAC_C  C220 4100 9A05 9D25 0000 Bk3 Include
LAC_C  C220 4100 BA01 BD21 0000 Bk2 Include
LAC_C  C220 4100 BA05 BD25 0000 Bk3 Include
LAC_C  C220 4100 9A45 9D65 0000 Bk3 Include
RLAC_C C220 2110 9298 75C8 0000 Bk2 Include
RLAC_C C220 2110 9201 7531 0000 Bk2 Include
RLAC_C C220 2110 9A01 7D31 0000 Bk2 Include
RLAC_C C220 2110 9A05 7D35 0000 Bk3 Include
XLAC_C C220 2108 9298 75C0 0000 Bk2 Include
XLAC_C C220 2108 9201 7529 0000 Bk2 Include
XLAC_C C220 2108 9A01 7D29 0000 Bk2 Include
XLAC_C C220 2108 9A05 7D2D 0000 Bk3 Include
XLAC_C C220 4108 9298 95C0 0000 Bk2 Include
XLAC_C C220 4108 9201 9529 0000 Bk2 Include
XLAC_C C220 4108 9A01 9D29 0000 Bk2 Include
XLAC_C C220 4108 9A05 9D2D 0000 Bk3 Include
LAC_D  C210 4080 9298 9528 0000 Bk2 Include
LAC_D  C210 4080 9201 9491 0000 Bk2 Include
LAC_D  C210 4080 9A01 9C91 0000 Bk2 Include
LAC_D  C210 4080 9A05 9C95 0000 Bk3 Include
LAC_D  C210 4080 BA01 BC91 0000 Bk2 Include
LAC_D  C210 4080 BA05 BC95 0000 Bk3 Include
LAC_D  C210 4080 9A45 9CD5 0000 Bk3 Include
LAC_D  C210 4100 9298 95A8 0000 Bk2 Include
LAC_D  C210 4100 9201 9511 0000 Bk2 Include
LAC_D  C210 4100 9A01 9D11 0000 Bk2 Include
LAC_D  C210 4100 9A05 9D15 0000 Bk3 Include
LAC_D  C210 4100 BA01 BD11 0000 Bk2 Include
LAC_D  C210 4100 BA05 BD15 0000 Bk3 Include
LAC_D  C210 4100 9A45 9D55 0000 Bk3 Include
XLAC_D C210 4188 9298 9630 0000 Bk2 Include
XLAC_D C210 4188 9201 9599 0000 Bk2 Include
XLAC_D C210 4188 9A01 9D99 0000 Bk2 Include
XLAC_D C210 4188 9A05 9D9D 0000 Bk3 Include
XLAC_D C210 2188 9298 7630 0000 Bk2 Include
XLAC_D C210 2188 9201 7599 0000 Bk2 Include
XLAC_D C210 2188 9A01 7D99 0000 Bk2 Include
XLAC_D C210 2188 9A05 7D9D 0000 Bk3 Include
XLAC_D C210 2208 9298 76B0 0000 Bk2 Include
XLAC_D C210 2208 9201 7619 0000 Bk2 Include
XLAC_D C210 2208 9A01 7E19 0000 Bk2 Include
XLAC_D C210 2208 9A05 7E1D 0000 Bk3 Include
LAC_E  C308 4080 0C01 2021 2FAA Bk4 Include
LAC_E  C308 4080 2C01 2021 4FAA Bk4 Include
XLAC_E C308 2088 0C01 2021 0FB2 Bk4 Include
XLAC_E C308 2088 2C01 2021 2FB2 Bk4 Include
XLAC_E C308 4088 0C01 2021 2FB2 Bk4 Include
XLAC_E C308 4088 2C01 2021 4FB2 Bk4 Include
```

```
-------------------------------------------------------------------------------
-- The missile sequence number, sdl required, bdii required, TLD, mission id, use state
-- (Execute, LaunchPlan, HoldFire), task role (Primary, ReadySpare, Backup, CallForFire,
-- Pooled), [times in minutes] launch time earliest, launch time planned, la unch time
-- latest, missile type, optional specific tasking (tail number, xmid, or cell id)
-------------------------------------------------------------------------------
MSN 1 false false TL001 048-001-0100 Execute Primary @+160 @+165 @+170 LAC_C
MSN 2 false false TL001 048-001-0101 Execute Primary @+160 @+165 @+170 LAC_C
MSN 3 false false TL001 048-001-0102 Execute Primary @+160 @+165 @+170 LAC_C
MSN 4 false false TL001 048-001-0103 Execute Primary @+160 @+165 @+170 LAC_C
MSN 5 false false TL001 048-001-0104 Execute Primary @+160 @+165 @+170 LAC_C

-------------------------------------------------------------------------------
-- The missile inventory. This includes the cell id, xmid, dsmac capability, gps
-- capability, sdl capability, missile status (available, inop, reload, or selected),
-- and optionally, the plan the missile is paired to (e.g., P1) and the missile sequence
-- number.
-------------------------------------------------------------------------------
B1      C220 4100 9298 95B8 0000      true    true    true    Available
B1      C220 4100 9298 95B8 0000      true    true    true    Available
A1      C220 4100 9298 95B8 0000      true    true    true    Available
A1      C220 4100 9298 95B8 0000      true    true    true    Available

-------------------------------------------------------------------------- -----
-- Define the missions. This includes mission id, dsmac required, gps
-- required, and the xmids (separated by commas) comprising the m3 list.
-------------------------------------------------------------------------------
048-001-0100 true true C220 4100 9298 95B8 0000
048-001-0101 true true C220 4100 9298 95B8 0000
048-001-0102 true true C220 4100 9298 95B8 0000
048-001-0103 true true C220 4100 9298 95B8 0000
048-001-0104 true true C220 4100 9298 95B8 0000
```

APPENDIX E

Tomahawk Validation Testing

This appendix includes comparison of the validation analysis for the derived test set 5.6 and for the test set developed by Kristen Fones [MF03]. It compares the results of each when run against the prototype and the actual Tomahawk deployed code

NOTE THIS APPENDIX IS WITHHELD DUE TO CLASSIFICATION

CONTACT THE AUTHOR TO REQUEST THIS INFORMATION.

COMPARISON OF DERIVED TESTSET

Test File	Prototype Results	Tomahawk Code Results
0	Performed 5 allocations Allocation[task=2,launcher=TT1,missile=16,manual=false,chosen=false] Allocation[task=9,launcher=CLS12,missile=11,manual=false,chosen=false] Allocation[task=11,launcher=CLS14,missile=13,manual=false,chosen=false] Allocation[task=14,launcher=CLS13,missile=12,manual=false,chosen=false] Allocation[task=18,launcher=TT2,missile=1,manual=false,chosen=false]	Performed 4 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute CallForFire A1 15:32:13 15:56:13 048-001-00100 00007 LaunchPlan ReadySpare VT13 15:19:13 16:38:13 048-001-00106 00009 LaunchPlan ReadySpare VT12 15:18:13 15:55:13 048-001-00108 00013 LaunchPlan CallForFire VT14 15:30:13 16:27:13 048-001-00112
1	Performed 4 allocations Allocation[task=14,launcher=CLS12,missile=11,manual=false,chosen=false] Allocation[task=25,launcher=CLS8,missile=7,manual=false,chosen=false] Allocation[task=33,launcher=TT4,missile=28,manual=false,chosen=false] Allocation[task=47,launcher=CLS7,missile=6,manual=false,chosen=false]	Performed 3 allocations MSN Use Role Alloc Earliest Latest Mission ID 00014 Execute ReadySpare VT12 15:18:16 16:02:16 048-001-00113 00033 Execute CallForFire VT7 15:16:16 15:27:16 048-001-00132 00051 LaunchPlan Pooled VT8 15:17:16 15:54:16 048-001-00150
2	Performed 7 allocations Allocation[task=12,launcher=CLS14,missile=13,manual=false,chosen=false] Allocation[task=13,launcher=TT2,missile=16,manual=false,chosen=false] Allocation[task=14,launcher=CLS12,missile=11,manual=false,chosen=false] Allocation[task=15,launcher=TT4,missile=3,manual=false,chosen=false] Allocation[task=18,launcher=CLS8,missile=7,manual=false,chosen=false] Allocation[task=23,launcher=CLS16,missile=15,manual=false,chosen=false] Allocation[task=26,launcher=CLS13,missile=12,manual=false,chosen=false]	Performed 6 allocations MSN Use Role Alloc Earliest Latest Mission ID 00005 LaunchPlan CallForFire TT4 15:19:21 16:48:21 048-001-00104 00007 LaunchPlan ReadySpare TT2 15:29:21 16:51:21 048-001-00106 00009 Execute Pooled VT16 15:15:21 17:03:21 048-001-00108 00014 Execute Primary VT12 15:12:21 15:49:21 048-001-00113 00018 Execute CallForFire VT8 15:25:21 15:55:21 048-001-00117 00023 Execute Primary VT14 15:20:21 16:42:21 048-001-00122
3	Error	Performed 7 allocations MSN Use Role Alloc Earliest Latest Mission ID 00013 Execute Pooled VT9 15:11:24 15:21:24 048-001-00112 00026 Execute Primary VT12 15:36:24 15:40:24 048-001-00125 00027 LaunchPlan CallForFire TT1 15:35:24 16:37:24 048-001-00126 00033 LaunchPlan Primary TT2 15:22:24 17:05:24 048-001-00132 00044 Execute ReadySpare TT4 15:26:24 16:46:24 048-001-00143 00055 LaunchPlan CallForFire VT6 15:31:24 17:08:24 048-001-00154 00062 Execute ReadySpare VT5 15:29:24 15:43:24 048-001-00161
4	Performed 5 allocations Allocation[task=11,launcher=CLS13,missile=12,manual=false,chosen=false] Allocation[task=15,launcher=CLS14,missile=13,manual=false,chosen=false] Allocation[task=26,launcher=CLS11,missile=10,manual=false,chosen=false] Allocation[task=30,launcher=TT1,missile=16,manual=false,chosen=false] Allocation[task=6,launcher=TT2,missile=0,manual=false,chosen=false]	Performed 5 allocations MSN Use Role Alloc Earliest Latest Mission ID 00006 LaunchPlan CallForFire TT1 15:31:29 16:41:29 048-001-00105 00011 Execute Primary VT13 15:08:29 15:31:29 048-001-00110 00015 Execute Backup VT14 15:25:29 15:09:29 048-001-00114 00026 Execute Backup VT11 15:27:29 16:59:29 048-001-00125 00029 Execute Backup A1 15:25:29 17:03:29 048-001-00128
5	Performed 4 allocations Allocation[task=11,launcher=TT2,missile=1,manual=true,chosen=true] Allocation[task=17,launcher=TT4,missile=3,manual=false,chosen=false] Allocation[task=19,launcher=CLS11,missile=10,manual=false,chosen=false] Allocation[task=25,launcher=CLS7,missile=6,manual=false,chosen=false]	Performed 7 allocations MSN Use Role Alloc Earliest Latest Mission ID 00006 Execute Primary TT4 15:28:33 15:49:33 048-001-00105 00008 Execute ReadySpare VT9 15:25:33 16:27:33 048-001-00107 00015 LaunchPlan Backup B1 15:13:33 17:04:33 048-001-00114 00019 LaunchPlan ReadySpare VT11 15:24:33 15:39:33 048-001-00118 00025 Execute Pooled VT7 15:22:33 15:19:33 048-001-00124 00027 Execute ReadySpare VT6 15:14:33 17:04:33 048-001-00126 00028 LaunchPlan Primary TT2 15:19:33 16:59:33 048-001-00127

Test File	Prototype Results	Tomahawk Code Results
6	Performed 4 allocations Allocation[task=15,launcher=TT4,missile=3,manual=false,chosen=false] Allocation[task=21,launcher=CLS11,missile=10,manual=false,chosen=false] Allocation[task=36,launcher=CLS6,missile=5,manual=false,chosen=false] Allocation[task=40,launcher=TT3,missile=2,manual=false,chosen=false]	Performed 5 allocations MSN Use Role Alloc Earliest Latest Mission ID 00012 Execute Backup VT11 15:31:37 16:30:37 048-001-00111 00021 Execute Primary VT16 15:34:37 16:41:37 048-001-00120 00036 LaunchPlan Pooled VT5 15:30:37 15:53:37 048-001-00135 00037 Execute ReadySpare TT4 15:07:37 16:44:37 048-001-00136 00040 Execute Backup TT3 15:29:37 17:06:37 048-001-00139
7	Performed 5 allocations Allocation[task=32,launcher=CLS11,missile=10,manual=false,chosen=false] Allocation[task=37,launcher=CLS5,missile=4,manual=false,chosen=false] Allocation[task=43,launcher=CLS15,missile=14,manual=false,chosen=false] Allocation[task=45,launcher=CLS8,missile=7,manual=false,chosen=false] Allocation[task=53,launcher=CLS6,missile=5,manual=false,chosen=false]	Performed 8 allocations MSN Use Role Alloc Earliest Latest Mission ID 00032 Execute Backup B1 15:12:41 16:35:41 048-001-00131 00035 LaunchPlan CallForFire VT6 15:23:41 15:42:41 048-001-00134 00037 Execute Pooled VT5 15:14:41 16:02:41 048-001-00136 00043 Execute CallForFire VT15 15:11:41 15:11:41 048-001-00142 00044 Execute Primary VT8 15:13:41 15:15:41 048-001-00143 00053 Execute Pooled VT11 15:29:41 16:38:41 048-001-00152 00064 Execute Primary VT13 15:23:41 16:12:41 048-001-00163 00074 Execute CallForFire VT9 15:30:41 16:00:41 048-001-00173
8	Performed 6 allocations Allocation[task=29,launcher=TT4,missile=3,manual=false,chosen=false] Allocation[task=36,launcher=TT1,missile=0,manual=false,chosen=false] Allocation[task=57,launcher=CLS14,missile=13,manual=false,chosen=false] Allocation[task=60,launcher=TT1,missile=16,manual=false,chosen=false] Allocation[task=78,launcher=CLS6,missile=5,manual=false,chosen=false] Allocation[task=8,launcher=TT2,missile=28,manual=false,chosen=false]	Performed 6 allocations MSN Use Role Alloc Earliest Latest Mission ID 00029 LaunchPlan Pooled VT14 15:34:46 17:12:46 048-001-00128 00036 Execute ReadySpare TT1 15:28:46 15:33:46 048-001-00135 00047 Execute Primary A1 15:18:46 16:24:46 048-001-00146 00057 Execute Backup VT9 15:17:46 15:09:46 048-001-00156 00066 LaunchPlan Primary TT2 15:16:46 15:21:46 048-001-00165 00078 Execute Primary VT6 15:24:46 16:43:46 048-001-00177
9	Performed 6 allocations Allocation[task=43,launcher=CLS11,missile=10,manual=false,chosen=false] Allocation[task=45,launcher=CLS16,missile=15,manual=false,chosen=false] Allocation[task=49,launcher=CLS13,missile=12,manual=false,chosen=false] Allocation[task=50,launcher=CLS5,missile=4,manual=false,chosen=false] Allocation[task=61,launcher=CLS15,missile=14,manual=false,chosen=false] Allocation[task=9,launcher=CLS9,missile=8,manual=false,chosen=false]	Performed 4 allocations MSN Use Role Alloc Earliest Latest Mission ID 00009 Execute Pooled VT9 15:17:52 16:08:52 048-001-00108 00050 LaunchPlan Pooled VT5 15:11:52 15:41:52 048-001-00149 00057 LaunchPlan Backup VT13 15:20:52 17:02:52 048-001-00156 00061 Execute CallForFire VT15 15:36:52 16:50:52 048-001-00160
10	Performed 6 allocations Allocation[task=13,launcher=CLS12,missile=11,manual=false,chosen=false] Allocation[task=14,launcher=TT4,missile=2,manual=false,chosen=false] Allocation[task=15,launcher=CLS14,missile=13,manual=false,chosen=false] Allocation[task=16,launcher=TT3,missile=28,manual=false,chosen=false] Allocation[task=18,launcher=CLS6,missile=5,manual=false,chosen=false] Allocation[task=4,launcher=CLS7,missile=6,manual=false,chosen=false]	Performed 7 allocations MSN Use Role Alloc Earliest Latest Mission ID 00004 LaunchPlan Backup VT7 15:35:57 16:13:57 048-001-00103 00009 Execute Primary VT6 15:20:57 15:27:57 048-001-00108 00013 LaunchPlan Backup VT12 15:09:57 15:56:57 048-001-00112 00014 Execute Pooled TT3 15:07:57 16:31:57 048-001-00113 00015 Execute Pooled VT14 15:33:57 16:31:57 048-001-00114 00016 LaunchPlan CallForFire B1 15:19:57 16:09:57 048-001-00115 00019 Execute CallForFire VT10 15:08:57 16:31:57 048-001-00118
11	Performed 6 allocations Allocation[task=14,launcher=CLS6,missile=5,manual=false,chosen=false] Allocation[task=23,launcher=TT3,missile=16,manual=false,chosen=false] Allocation[task=29,launcher=TT4,missile=2,manual=false,chosen=false] Allocation[task=3,launcher=CLS10,missile=9,manual=false,chosen=false] Allocation[task=41,launcher=CLS12,missile=11,manual=false,chosen=false] Allocation[task=52,launcher=CLS15,missile=14,manual=false,chosen=false]	Performed 7 allocations MSN Use Role Alloc Earliest Latest Mission ID 00003 Execute CallForFire VT7 15:22:00 16:21:00 048-001-00102 00013 LaunchPlan Primary TT3 15:22:00 16:09:00 048-001-00112 00021 LaunchPlan ReadySpare VT8 15:19:00 16:36:00 048-001-00120 00023 Execute CallForFire A1 15:19:00 16:26:00 048-001-00122 00052 LaunchPlan Pooled VT15 15:37:00 16:55:00 048-001-00151 00058 LaunchPlan Pooled VT12 15:30:00 16:25:00 048-001-00157 00063 LaunchPlan Pooled VT5 15:36:00 16:42:00 048-001-00162

Test File	Prototype Results	Tomahawk Code Results
12	Performed 11 allocations Allocation[task=1,launcher=CLS13,missile=12,manual=false,chosen=false] Allocation[task=12,launcher=TT1,missile=28,manual=false,chosen=false] Allocation[task=27,launcher=CLS8,missile=7,manual=false,chosen=false] Allocation[task=29,launcher=CLS11,missile=10,manual=false,chosen=false] Allocation[task=56,launcher=CLS15,missile=14,manual=false,chosen=false] Allocation[task=66,launcher=CLS12,missile=11,manual=false,chosen=false] Allocation[task=69,launcher=CLS6,missile=5,manual=false,chosen=false] Allocation[task=72,launcher=CLS9,missile=8,manual=false,chosen=false] Allocation[task=73,launcher=CLS7,missile=6,manual=false,chosen=false] Allocation[task=77,launcher=TT4,missile=3,manual=false,chosen=false] Allocation[task=81,launcher=TT3,missile=16,manual=false,chosen=false]	Performed 9 allocations MSN Use Role Alloc Earliest Latest Mission ID 00029 Execute Backup VT8 15:35:05 17:17:05 048-001-00128 00036 LaunchPlan Primary VT11 15:10:05 15:42:05 048-001-00135 00058 LaunchPlan Backup B1 15:34:05 16:24:05 048-001-00157 00066 LaunchPlan CallForFire VT12 15:24:05 15:34:05 048-001-00165 00069 Execute Primary VT6 15:22:05 16:41:05 048-001-00168 00077 Execute CallForFire TT4 15:27:05 15:10:05 048-001-00176 00080 Execute Backup VT9 15:20:05 16:02:05 048-001-00179 00081 LaunchPlan Backup VT7 15:37:05 15:38:05 048-001-00180 00089 LaunchPlan ReadySpare TT1 15:33:05 15:14:05 048-001-00188
13	Performed 9 allocations Allocation[task=12,launcher=CLS8,missile=7,manual=false,chosen=false] Allocation[task=2,launcher=CLS14,missile=13,manual=false,chosen=false] Allocation[task=3,launcher=CLS7,missile=6,manual=false,chosen=false] Allocation[task=45,launcher=TT4,missile=3,manual=false,chosen=false] Allocation[task=54,launcher=TT3,missile=1,manual=false,chosen=false] Allocation[task=6,launcher=CLS16,missile=15,manual=false,chosen=false] Allocation[task=63,launcher=TT2,missile=2,manual=false,chosen=false] Allocation[task=73,launcher=CLS13,missile=12,manual=false,chosen=false] Allocation[task=74,launcher=TT4,missile=28,manual=false,chosen=false]	Performed 8 allocations MSN Use Role Alloc Earliest Latest Mission ID 00002 LaunchPlan CallForFire TT2 15:31:12 17:16:12 048-001-00101 00012 Execute CallForFire VT8 15:19:12 15:13:12 048-001-00111 00025 Execute ReadySpare TT4 15:09:12 16:39:12 048-001-00124 00033 Execute ReadySpare VT9 15:33:12 15:50:12 048-001-00132 00040 LaunchPlan ReadySpare VT13 15:17:12 15:15:12 048-001-00139 00047 LaunchPlan CallForFire VT16 15:13:12 17:02:12 048-001-00146 00063 Execute CallForFire TT3 15:12:12 15:28:12 048-001-00162 00083 LaunchPlan CallForFire VT7 15:15:12 16:58:12 048-001-00182
14	Performed 8 allocations Allocation[task=2,launcher=CLS10,missile=9,manual=false,chosen=false] Allocation[task=27,launcher=CLS9,missile=8,manual=false,chosen=false] Allocation[task=28,launcher=TT3,missile=1,manual=false,chosen=false] Allocation[task=39,launcher=CLS5,missile=4,manual=false,chosen=false] Allocation[task=42,launcher=TT4,missile=3,manual=false,chosen=false] Allocation[task=44,launcher=CLS12,missile=11,manual=false,chosen=false] Allocation[task=63,launcher=TT2,missile=2,manual=false,chosen=false] Allocation[task=7,launcher=CLS14,missile=13,manual=false,chosen=false]	Performed 9 allocations MSN Use Role Alloc Earliest Latest Mission ID 00004 Execute ReadySpare TT4 15:27:18 17:03:18 048-001-00103 00007 Execute CallForFire VT14 15:24:18 16:46:18 048-001-00106 00018 Execute Backup TT3 15:14:18 16:31:18 048-001-00117 00027 Execute Backup VT9 15:21:18 16:02:18 048-001-00126 00028 LaunchPlan CallForFire TT2 15:30:18 15:40:18 048-001-00127 00039 Execute CallForFire VT5 15:27:18 16:54:18 048-001-00138 00042 Execute CallForFire VT11 15:18:18 16:54:18 048-001-00141 00044 Execute Primary VT12 15:21:18 16:33:18 048-001-00143 00055 Execute CallForFire VT10 15:14:18 17:00:18 048-001-00154
15	Performed 1 allocations Allocation[task=9,launcher=TT1,missile=0,manual=false,chosen=false]	Performed 1 allocations MSN Use Role Alloc Earliest Latest Mission ID 00009 Execute ReadySpare TT1 15:23:24 16:59:24 048-001-00108
16	Performed 5 allocations Allocation[task=13,launcher=TT1,missile=0,manual=false,chosen=false] Allocation[task=17,launcher=CLS16,missile=15,manual=false,chosen=false] Allocation[task=22,launcher=CLS13,missile=12,manual=false,chosen=false] Allocation[task=37,launcher=CLS10,missile=9,manual=false,chosen=false] Allocation[task=43,launcher=CLS5,missile=4,manual=false,chosen=false]	Performed 4 allocations MSN Use Role Alloc Earliest Latest Mission ID 00013 Execute CallForFire TT1 15:34:26 15:23:26 048-001-00112 00017 Execute CallForFire VT16 15:14:26 16:15:26 048-001-00116 00023 LaunchPlan Primary VT10 15:09:26 16:58:26 048-001-00122 00043 Execute Primary VT5 15:24:26 17:11:26 048-001-00142
17	Performed 5 allocations Allocation[task=20,launcher=CLS7,missile=6,manual=false,chosen=false] Allocation[task=48,launcher=TT3,missile=2,manual=false,chosen=false] Allocation[task=76,launcher=TT3,missile=16,manual=false,chosen=false] Allocation[task=84,launcher=TT4,missile=3,manual=false,chosen=false] Allocation[task=97,launcher=CLS11,missile=10,manual=false,chosen=false]	Performed 6 allocations MSN Use Role Alloc Earliest Latest Mission ID 00048 Execute Primary VT9 15:34:31 15:51:31 048-001-00147 00076 Execute ReadySpare VT7 15:19:31 15:15:31 048-001-00175 00084 Execute Primary TT4 15:08:31 16:51:31 048-001-00183 00092 Execute Backup VT15 15:08:31 16:43:31 048-001-00191 00096 Execute CallForFire TT3 15:12:31 15:27:31 048-001-00195 00097 Execute Backup VT11 15:22:31 16:38:31 048-001-00196

Test File	Prototype Results	Tomahawk Code Results
18	Performed 5 allocations Allocation[task=17,launcher=TT3,missile=0,manual=false,chosen=false] Allocation[task=27,launcher=CLS11,missile=10,manual=false,chosen=false] Allocation[task=42,launcher=CLS5,missile=4,manual=false,chosen=false] Allocation[task=53,launcher=TT1,missile=2,manual=false,chosen=false] Allocation[task=91,launcher=CLS9,missile=8,manual=false,chosen=false]	Performed 5 allocations MSN Use Role Alloc Earliest Latest Mission ID 00009 Execute ReadySpare VT11 15:33:38 16:57:38 048-001-00108 00027 Execute Backup VT9 15:09:38 17:03:38 048-001-00126 00042 Execute Primary VT5 15:13:38 16:04:38 048-001-00141 00053 Execute Backup TT3 15:25:38 17:08:38 048-001-00152 00067 Execute Primary TT1 15:30:38 15:21:38 048-001-00166
19	Performed 5 allocations Allocation[task=10,launcher=CLS14,missile=13,manual=false,chosen=false] Allocation[task=11,launcher=CLS6,missile=5,manual=true,chosen=true] Allocation[task=14,launcher=CLS5,missile=4,manual=false,chosen=false] Allocation[task=17,launcher=CLS13,missile=12,manual=false,chosen=false] Allocation[task=4,launcher=CLS7,missile=6,manual=false,chosen=false]	Performed 4 allocations MSN Use Role Alloc Earliest Latest Mission ID 00002 LaunchPlan ReadySpare VT8 15:28:44 15:57:44 048-001-00101 00004 LaunchPlan CallForFire VT7 15:16:44 16:03:44 048-001-00103 00010 Execute CallForFire VT14 15:21:44 15:41:44 048-001-00109 00017 LaunchPlan Primary VT10 15:17:44 15:33:44 048-001-00116
20	Performed 7 allocations Allocation[task=13,launcher=CLS9,missile=8,manual=false,chosen=false] Allocation[task=21,launcher=CLS6,missile=5,manual=false,chosen=false] Allocation[task=47,launcher=CLS12,missile=11,manual=false,chosen=false] Allocation[task=62,launcher=CLS13,missile=12,manual=false,chosen=false] Allocation[task=67,launcher=CLS15,missile=14,manual=false,chosen=false] Allocation[task=69,launcher=CLS7,missile=6,manual=false,chosen=false] Allocation[task=74,launcher=CLS5,missile=4,manual=false,chosen=false]	Performed 7 allocations MSN Use Role Alloc Earliest Latest Mission ID 00014 Execute Pooled B1 15:36:47 16:10:47 048-001-00113 00047 Execute Primary VT12 15:08:47 16:06:47 048-001-00146 00052 LaunchPlan CallForFire A1 15:12:47 16:00:47 048-001-00151 00062 Execute Pooled VT5 15:20:47 15:13:47 048-001-00161 00067 Execute Backup VT15 15:32:47 15:37:47 048-001-00166 00068 Execute CallForFire VT6 15:12:47 15:51:47 048-001-00167 00069 Execute Backup VT7 15:32:47 16:40:47 048-001-00168
21	Performed 6 allocations Allocation[task=11,launcher=TT4,missile=3,manual=false,chosen=false] Allocation[task=17,launcher=CLS13,missile=12,manual=false,chosen=false] Allocation[task=24,launcher=CLS16,missile=15,manual=false,chosen=false] Allocation[task=43,launcher=TT3,missile=2,manual=false,chosen=false] Allocation[task=6,launcher=TT2,missile=28,manual=false,chosen=false] Allocation[task=63,launcher=CLS8,missile=7,manual=false,chosen=false]	Performed 6 allocations MSN Use Role Alloc Earliest Latest Mission ID 00011 Execute CallForFire TT4 15:36:53 15:26:53 048-001-00110 00023 LaunchPlan ReadySpare B1 15:36:53 16:12:53 048-001-00122 00024 Execute Backup VT16 15:36:53 15:39:53 048-001-00123 00031 Execute Pooled TT3 15:37:53 16:10:53 048-001-00130 00039 Execute Pooled VT13 15:16:53 17:00:53 048-001-00138 00063 LaunchPlan CallForFire VT8 15:14:53 15:38:53 048-001-00162
22	Performed 9 allocations Allocation[task=12,launcher=CLS5,missile=4,manual=false,chosen=false] Allocation[task=14,launcher=TT1,missile=0,manual=false,chosen=false] Allocation[task=36,launcher=TT3,missile=1,manual=false,chosen=false] Allocation[task=60,launcher=CLS6,missile=5,manual=false,chosen=false] Allocation[task=62,launcher=CLS9,missile=8,manual=false,chosen=false] Allocation[task=69,launcher=CLS11,missile=10,manual=false,chosen=false] Allocation[task=73,launcher=CLS16,missile=15,manual=false,chosen=false] Allocation[task=78,launcher=CLS10,missile=9,manual=false,chosen=false] Allocation[task=81,launcher=TT2,missile=2,manual=false,chosen=false]	Performed 6 allocations MSN Use Role Alloc Earliest Latest Mission ID 00007 Execute ReadySpare TT3 15:27:57 16:43:57 048-001-00106 00014 Execute Primary TT1 15:25:57 15:10:57 048-001-00113 00060 Execute CallForFire VT11 15:36:57 15:11:57 048-001-00159 00062 Execute CallForFire VT9 15:37:57 15:22:57 048-001-00161 00069 Execute Primary VT6 15:30:57 15:53:57 048-001-00168 00078 Execute CallForFire VT10 15:22:57 15:29:57 048-001-00177

Test File	Prototype Results	Tomahawk Code Results
23	Performed 10 allocations Allocation[task=1,launcher=TT4,missile=3,manual=false,chosen=false] Allocation[task=32,launcher=CLS16,missile=15,manual=false,chosen=false] Allocation[task=33,launcher=TT1,missile=0,manual=false,chosen=false] Allocation[task=45,launcher=CLS10,missile=9,manual=false,chosen=false] Allocation[task=47,launcher=CLS12,missile=11,manual=false,chosen=false] Allocation[task=49,launcher=TT2,missile=28,manual=false,chosen=false] Allocation[task=63,launcher=CLS9,missile=8,manual=false,chosen=false] Allocation[task=64,launcher=CLS11,missile=10,manual=false,chosen=false] Allocation[task=67,launcher=CLS13,missile=12,manual=false,chosen=false] Allocation[task=74,launcher=TT4,missile=1,manual=false,chosen=false]	Performed 11 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Pooled TT4 15:32:03 16:03:03 048-001-00100 00032 Execute Backup VT16 15:32:03 16:06:03 048-001-00131 00033 Execute CallForFire VT8 15:14:03 15:12:03 048-001-00132 00043 Execute Primary VT12 15:28:03 15:47:03 048-001-00142 00045 Execute CallForFire VT10 15:16:03 15:23:03 048-001-00144 00049 Execute ReadySpare B1 15:15:03 16:55:03 048-001-00148 00063 Execute Pooled VT9 15:25:03 16:57:03 048-001-00162 00064 Execute Pooled VT11 15:29:03 16:42:03 048-001-00163 00067 Execute ReadySpare VT13 15:28:03 15:20:03 048-001-00166 00071 LaunchPlan Primary TT1 15:38:03 15:49:03 048-001-00170 00074 Execute CallForFire TT2 15:12:03 15:11:03 048-001-00173
24	Performed 8 allocations Allocation[task=15,launcher=CLS14,missile=13,manual=false,chosen=false] Allocation[task=18,launcher=CLS9,missile=8,manual=false,chosen=false] Allocation[task=19,launcher=TT3,missile=0,manual=false,chosen=false] Allocation[task=27,launcher=TT1,missile=2,manual=false,chosen=false] Allocation[task=4,launcher=CLS10,missile=9,manual=false,chosen=false] Allocation[task=49,launcher=CLS13,missile=12,manual=false,chosen=false] Allocation[task=6,launcher=CLS6,missile=5,manual=false,chosen=false] Allocation[task=9,launcher=TT4,missile=3,manual=false,chosen=false]	Performed 8 allocations MSN Use Role Alloc Earliest Latest Mission ID 00004 LaunchPlan Primary VT10 15:20:09 16:53:09 048-001-00103 00006 Execute Backup VT6 15:21:09 15:28:09 048-001-00105 00009 Execute CallForFire TT4 15:29:09 16:34:09 048-001-00108 00015 Execute CallForFire VT14 15:30:09 15:48:09 048-001-00114 00027 Execute CallForFire VT9 15:10:09 15:52:09 048-001-00126 00036 Execute CallForFire VT13 15:34:09 16:58:09 048-001-00135 00049 Execute CallForFire TT1 15:17:09 17:01:09 048-001-00148 00051 LaunchPlan Primary TT3 15:15:09 15:38:09 048-001-00150
25	Performed 10 allocations Allocation[task=1,launcher=TT1,missile=16,manual=false,chosen=false] Allocation[task=17,launcher=CLS5,missile=4,manual=false,chosen=false] Allocation[task=20,launcher=CLS7,missile=6,manual=false,chosen=false] Allocation[task=23,launcher=CLS8,missile=7,manual=false,chosen=false] Allocation[task=24,launcher=CLS15,missile=14,manual=false,chosen=false] Allocation[task=39,launcher=CLS10,missile=9,manual=false,chosen=false] Allocation[task=5,launcher=TT2,missile=1,manual=false,chosen=false] Allocation[task=53,launcher=TT1,missile=0,manual=false,chosen=false] Allocation[task=57,launcher=CLS11,missile=10,manual=false,chosen=false] Allocation[task=63,launcher=CLS12,missile=11,manual=false,chosen=false]	Performed 10 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Backup TT2 15:09:14 15:17:14 048-001-00100 00005 LaunchPlan Primary A1 15:28:14 16:31:14 048-001-00104 00020 Execute CallForFire VT7 15:36:14 17:12:14 048-001-00119 00023 LaunchPlan CallForFire VT8 15:16:14 15:55:14 048-001-00122 00024 Execute Backup VT15 15:14:14 16:27:14 048-001-00123 00034 Execute Pooled VT11 15:14:14 15:12:14 048-001-00133 00039 LaunchPlan CallForFire VT10 15:24:14 15:49:14 048-001-00138 00045 LaunchPlan Primary VT5 15:14:14 15:24:14 048-001-00144 00053 LaunchPlan Pooled TT1 15:38:14 16:31:14 048-001-00152 00063 Execute ReadySpare VT12 15:18:14 15:32:14 048-001-00162
26	Performed 6 allocations Allocation[task=11,launcher=TT4,missile=3,manual=false,chosen=false] Allocation[task=37,launcher=CLS10,missile=9,manual=false,chosen=false] Allocation[task=46,launcher=CLS6,missile=5,manual=false,chosen=false] Allocation[task=5,launcher=CLS9,missile=8,manual=false,chosen=false] Allocation[task=56,launcher=TT1,missile=0,manual=false,chosen=false] Allocation[task=63,launcher=CLS11,missile=10,manual=false,chosen=false]	Performed 4 allocations MSN Use Role Alloc Earliest Latest Mission ID 00011 Execute Primary TT4 15:35:19 16:03:19 048-001-00110 00046 Execute ReadySpare VT6 15:38:19 16:21:19 048-001-00145 00056 Execute Primary TT1 15:37:19 15:38:19 048-001-00155 00063 Execute Backup VT10 15:12:19 16:59:19 048-001-00162

Test File	Prototype Results	Tomahawk Code Results
27	Performed 9 allocations Allocation[task=1,launcher=CLS5,missile=4,manual=false,chosen=false] Allocation[task=17,launcher=CLS15,missile=14,manual=false,chosen=false] Allocation[task=21,launcher=CLS9,missile=8,manual=false,chosen=false] Allocation[task=22,launcher=CLS8,missile=7,manual=false,chosen=false] Allocation[task=27,launcher=CLS14,missile=13,manual=false,chosen=false] Allocation[task=32,launcher=CLS7,missile=6,manual=false,chosen=false] Allocation[task=46,launcher=CLS11,missile=10,manual=false,chosen=false] Allocation[task=74,launcher=CLS6,missile=5,manual=false,chosen=false] Allocation[task=82,launcher=CLS13,missile=12,manual=false,chosen=false]	Performed 11 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute CallForFire VT5 15:10:24 15:14:24 048-001-00100 00006 Execute Backup VT15 15:36:24 15:57:24 048-001-00105 00017 Execute CallForFire A1 15:30:24 17:03:24 048-001-00116 00021 Execute Primary VT7 15:29:24 16:25:24 048-001-00120 00045 LaunchPlan Primary VT9 15:26:24 17:13:24 048-001-00144 00051 Execute Pooled VT8 15:34:24 17:12:24 048-001-00150 00061 Execute ReadySpare VT14 15:21:24 15:59:24 048-001-00160 00065 Execute Primary VT11 15:29:24 16:31:24 048-001-00164 00067 Execute Primary B1 15:17:24 16:19:24 048-001-00166 00074 Execute Primary VT6 15:29:24 16:19:24 048-001-00173 00082 Execute Pooled VT13 15:21:24 16:20:24 048-001-00181
28	Performed 3 allocations Allocation[task=1,launcher=TT1,missile=16,manual=false,chosen=false] Allocation[task=13,launcher=TT4,missile=0,manual=false,chosen=false] Allocation[task=17,launcher=TT2,missile=2,manual=false,chosen=false]	Performed 2 allocations MSN Use Role Alloc Earliest Latest Mission ID 00010 Execute ReadySpare A1 15:25:30 16:46:30 048-001-00109 00014 Execute Primary VT6 15:17:30 15:41:30 048-001-00113
29	Performed 2 allocations Allocation[task=1,launcher=CLS14,missile=13,manual=false,chosen=false] Allocation[task=9,launcher=TT4,missile=2,manual=false,chosen=false]	Performed 1 allocations MSN Use Role Alloc Earliest Latest Mission ID 00009 Execute ReadySpare VT12 15:16:33 16:27:33 048-001-00108

COMPARISON OF FONES TESTSET

Test File	Prototype Results	Tomahawk Code Results
0	Performed 1 allocations Allocation[task=1,launcher=TT2,missile=17,manual=false,chosen=false]	Performed 1 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Primary B1 17:42:50 17:52:50 048-001-00100
1	Performed 1 allocations Allocation[task=3,launcher=TT2,missile=17,manual=false,chosen=false]	Performed 1 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Primary B1 17:42:57 17:52:57 048-001-00100
2	Performed 2 allocations Allocation[task=1,launcher=TT2,missile=18,manual=false,chosen=false] Allocation[task=2,launcher=CLS6,missile=17,manual=false,chosen=false]	Performed 2 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Primary B1 17:43:05 17:53:05 048-001-00100 00002 Execute ReadySpare VT6 17:43:05 17:53:05 048-001-00101
3	Performed 2 allocations Allocation[task=1,launcher=CLS6,missile=17,manual=false,chosen=false] Allocation[task=2,launcher=TT2,missile=18,manual=false,chosen=false]	Performed 2 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Primary VT6 17:43:10 17:53:10 048-001-00100 00002 Execute ReadySpare B1 17:43:10 17:53:10 048-001-00101
4	Performed 2 allocations Allocation[task=1,launcher=TT2,missile=17,manual=false,chosen=false] Allocation[task=2,launcher=TT2,missile=18,manual=false,chosen=false]	Performed 1 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Primary B1 17:43:16 17:53:16 048-001-00100
5	Performed 2 allocations Allocation[task=1,launcher=TT1,missile=23,manual=false,chosen=false] Allocation[task=2,launcher=TT2,missile=21,manual=false,chosen=false]	Performed 1 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Primary B1 17:43:20 17:53:20 048-001-00100
6	Performed 2 allocations Allocation[task=1,launcher=TT2,missile=21,manual=false,chosen=false] Allocation[task=2,launcher=TT1,missile=23,manual=false,chosen=false]	Performed 1 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Primary B1 17:43:25 17:53:25 048-001-00100
7	Performed 2 allocations Allocation[task=1,launcher=CLS9,missile=21,manual=false,chosen=false] Allocation[task=2,launcher=CLS10,missile=23,manual=false,chosen=false]	Performed 2 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Primary VT9 17:43:29 17:53:29 048-001-00100 00002 Execute Backup VT10 19:23:29 19:33:29 048-001-00101
8	Performed 1 allocations Allocation[task=1,launcher=CLS9,missile=21,manual=false,chosen=false]	Performed 1 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Primary VT9 17:43:31 17:53:31 048-001-00100
9	Performed 1 allocations Allocation[task=1,launcher=CLS9,missile=21,manual=false,chosen=false]	Performed 1 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Primary VT9 17:43:34 17:53:34 048-001-00100
10	Performed 1 allocations Allocation[task=1,launcher=CLS9,missile=21,manual=false,chosen=false]	Performed 1 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Backup VT9 17:43:38 17:53:38 048-001-00100
11	Performed 1 allocations Allocation[task=1,launcher=CLS9,missile=21,manual=false,chosen=false]	Performed 1 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 LaunchPlan Primary VT9 17:43:42 17:53:42 048-001-00100
12	Performed 1 allocations Allocation[task=1,launcher=CLS9,missile=21,manual=false,chosen=false]	Performed 1 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 LaunchPlan ReadySpare VT9 17:43:46 17:53:46 048-001-00100
13	Performed 2 allocations Allocation[task=3,launcher=CLS5,missile=21,manual=false,chosen=false] Allocation[task=4,launcher=CLS6,missile=22,manual=false,chosen=false]	Performed 2 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Primary VT5 21:03:50 21:13:50 048-001-00100 00002 Execute Primary VT6 21:03:50 21:13:50 048-001-00101

Test File	Prototype Results	Tomahawk Code Results
14	Performed 2 allocations Allocation[task=3,launcher=CLS6,missile=22,manual=false,chosen=false] Allocation[task=4,launcher=CLS5,missile=21,manual=false,chosen=false]	Performed 2 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Primary VT5 21:03:52 21:13:52 048-001-00100 00002 Execute Primary VT6 21:03:52 21:13:52 048-001-00101
15	Performed 2 allocations Allocation[task=3,launcher=CLS5,missile=21,manual=false,chosen=false] Allocation[task=4,launcher=CLS6,missile=22,manual=false,chosen=false]	Performed 2 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Backup VT5 21:03:54 21:13:54 048-001-00100 00002 Execute Backup VT6 21:03:54 21:13:54 048-001-00101
16	Performed 2 allocations Allocation[task=3,launcher=CLS6,missile=22,manual=false,chosen=false] Allocation[task=4,launcher=CLS5,missile=21,manual=false,chosen=false]	Performed 2 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute ReadySpare VT5 21:03:56 21:13:56 048-001-00100 00002 Execute ReadySpare VT6 21:03:56 21:13:56 048-001-00101
17	Performed 1 allocations Allocation[task=1,launcher=CLS5,missile=17,manual=false,chosen=false]	Performed 1 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Primary VT5 17:43:59 17:53:59 048-001-00100
18	Performed 1 allocations Allocation[task=1,launcher=CLS16,missile=21,manual=false,chosen=false]	Performed 1 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Primary VT16 17:44:01 17:54:01 048-001-00100
19	Performed 1 allocations Allocation[task=1,launcher=CLS16,missile=21,manual=false,chosen=false]	Performed 1 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Backup VT16 17:44:05 17:54:05 048-001-00100
20	Performed 1 allocations Allocation[task=1,launcher=CLS16,missile=21,manual=false,chosen=false]	Performed 1 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 LaunchPlan Primary VT16 17:44:09 17:54:09 048-001-00100
21	Performed 1 allocations Allocation[task=1,launcher=CLS16,missile=21,manual=false,chosen=false]	Performed 1 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 LaunchPlan ReadySpare VT16 17:44:13 17:54:13 048-001-00100
22	Performed 2 allocations Allocation[task=1,launcher=CLS6,missile=22,manual=false,chosen=false] Allocation[task=2,launcher=CLS5,missile=21,manual=false,chosen=false]	Performed 2 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Primary VT5 21:04:17 21:14:17 048-001-00100 00002 Execute Primary VT6 21:04:17 21:14:17 048-001-00101
23	Performed 2 allocations Allocation[task=1,launcher=CLS5,missile=21,manual=false,chosen=false] Allocation[task=2,launcher=CLS6,missile=22,manual=false,chosen=false]	Performed 2 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Primary VT5 21:04:19 21:14:19 048-001-00100 00002 Execute Primary VT6 21:04:19 21:14:19 048-001-00101
24	Performed 2 allocations Allocation[task=1,launcher=CLS6,missile=22,manual=false,chosen=false] Allocation[task=2,launcher=CLS5,missile=21,manual=false,chosen=false]	Performed 2 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Backup VT5 21:04:22 21:14:22 048-001-00100 00002 Execute Backup VT6 21:04:22 21:14:22 048-001-00101
25	Performed 2 allocations Allocation[task=1,launcher=CLS5,missile=21,manual=false,chosen=false] Allocation[task=2,launcher=CLS6,missile=22,manual=false,chosen=false]	Performed 2 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute ReadySpare VT5 21:04:24 21:14:24 048-001-00100 00002 Execute ReadySpare VT6 21:04:24 21:14:24 048-001-00101
26	Performed 1 allocations Allocation[task=1,launcher=TT1,missile=21,manual=false,chosen=false]	Performed 1 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Primary TT1 17:44:27 17:54:27 048-001-00100
27	Performed 1 allocations Allocation[task=1,launcher=TT1,missile=22,manual=false,chosen=false]	Performed 1 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Primary A1 17:44:29 17:54:29 048-001-00100

Test File	Prototype Results	Tomahawk Code Results
28	Performed 1 allocations Allocation[task=1,launcher=CLS5,missile=21,manual=false,chosen=false]	Performed 1 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Primary VT5 17:44:31 17:54:31 048-001-00100
29	Performed 1 allocations Allocation[task=1,launcher=TT2,missile=21,manual=false,chosen=false]	Performed 1 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Primary B1 17:44:34 17:54:34 048-001-00100
30	Performed 2 allocations Allocation[task=1,launcher=TT1,missile=22,manual=false,chosen=false] Allocation[task=2,launcher=TT2,missile=21,manual=false,chosen=false]	Performed 2 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Primary B1 17:44:36 17:54:36 048-001-00100 00002 Execute Primary A1 17:44:36 17:54:36 048-001-00101
31	Performed 2 allocations Allocation[task=1,launcher=TT1,missile=22,manual=false,chosen=false] Allocation[task=2,launcher=TT2,missile=21,manual=false,chosen=false]	Performed 2 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Primary B1 17:44:38 17:54:38 048-001-00100 00002 Execute Primary A1 17:44:38 17:54:38 048-001-00101
32	Performed 1 allocations Allocation[task=1,launcher=TT2,missile=21,manual=false,chosen=false]	Performed 1 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Primary B1 17:44:41 17:54:41 048-001-00100
33	Performed 4 allocations Allocation[task=2,launcher=TT4,missile=22,manual=false,chosen=false] Allocation[task=3,launcher=TT2,missile=21,manual=false,chosen=false] Allocation[task=4,launcher=TT3,missile=23,manual=false,chosen=false] Allocation[task=5,launcher=TT1,missile=24,manual=false,chosen=false]	Performed 2 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Primary B1 17:44:43 17:54:43 048-001-00100 00002 Execute Primary A1 17:44:43 17:54:43 048-001-00101
34	Performed 1 allocations Allocation[task=1,launcher=TT1,missile=23,manual=false,chosen=false]	Performed 1 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Primary A1 17:44:46 17:54:46 048-001-00100
35	Performed 1 allocations Allocation[task=1,launcher=TT2,missile=21,manual=false,chosen=false]	Performed 1 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Primary B1 17:44:48 17:54:48 048-001-00100
36	Performed 0 allocations	Performed 0 allocations
37	Performed 1 allocations Allocation[task=1,launcher=CLS7,missile=27,manual=false,chosen=false]	Performed 1 allocations MSN Use Role Alloc Earliest Latest Mission ID 00001 Execute Primary VT5 17:44:53 17:54:53 048-001-00100

VITA

Richard Stutler was born on November 15, 1953 in Lansing, Michigan. He received his Bachelor of Science in Electrical Engineering from Virginia Polytechnic Institute and State University in 1975. He has been an employee of the Naval Surface Warfare Center in Dahlgren, Virginia since 1977. During this time he has been involved with several research efforts to prototype sophisticated digital elements before inclusion on Navy Ships. He has also served as the Division Head for the Tomahawk Mission Planning Division and more recently as the Deputy Department Head for the Strategic & Weapon Control Systems Department. This department is tasked with the development of weapon control software for several major systems in the Navy.

Figure 5 Richard Stutler