



# **NAVAL POSTGRADUATE SCHOOL**

**MONTEREY, CALIFORNIA**

## **THESIS**

### **AUTOMATED BEHAVIOR PROPERTY VERIFICATION TOOL**

by

John K. Leo

September 2008

Thesis Advisor:  
Second Reader:

Mikhail Auguston  
Man-Tak Shing

**Approved for public release; distribution is unlimited.**

THIS PAGE INTENTIONALLY LEFT BLANK

<b>REPORT DOCUMENTATION PAGE</b>			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
<b>1. AGENCY USE ONLY (Leave blank)</b>		<b>2. REPORT DATE</b> September 2008	<b>3. REPORT TYPE AND DATES COVERED</b> Master's Thesis	
<b>4. TITLE AND SUBTITLE</b> Automated Behavior Property Verification Tool			<b>5. FUNDING NUMBERS</b>	
<b>6. AUTHOR(S)</b> Leo, John K.				
<b>7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)</b> Naval Postgraduate School Monterey, CA 93943-5000			<b>8. PERFORMING ORGANIZATION REPORT NUMBER</b>	
<b>9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES)</b> N/A			<b>10. SPONSORING/MONITORING AGENCY REPORT NUMBER</b>	
<b>11. SUPPLEMENTARY NOTES</b> The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
<b>12a. DISTRIBUTION / AVAILABILITY STATEMENT</b> Approved for public release; distribution is unlimited.			<b>12b. DISTRIBUTION CODE</b>	
<b>13. ABSTRACT (maximum 200 words)</b> <p>Computer generated forces (CGF) simulations have entities as actors in their simulation. A type of CGF in which the entities have limited autonomy is semi-automated forces (SAF). The SAF system for this thesis research is OneSAF, a near real-time SAF that offers raw data collection of the entities in a particular simulation scenario. The data collection files vary in size from 500 kilobytes to larger than four gigabytes.</p> <p>Entity behavior property verification (BPV) is an integral part of SAF simulation software testing. The purpose for this research is to provide immediate feedback to the system user/developer as to what an entity had done in a scenario. From the simulation point of view, it provides answers to questions like "Did the entity route shortest distance to destination?" From the developer's point of interest, the BPV can provide insight to flaws in the model, such as a vehicle crossing a river where a bridge does not exist.</p> <p>Automated BPV (ABPV) takes one step further by minimizing "hard coding" of tools that process collection files. ABPV allows portability of the product of this thesis to other systems. ABPV Tools (ABPVT) of this thesis is designed to run in Linux and Windows and will be included in future distributions of OneSAF as an intricate part of the testing suite.</p>				
<b>14. SUBJECT TERMS</b> Entity Behavior Verification, Computer Simulation Verification, Analysis of Real-Time Simulation, OneSAF			<b>15. NUMBER OF PAGES</b> 157	
			<b>16. PRICE CODE</b>	
<b>17. SECURITY CLASSIFICATION OF REPORT</b> Unclassified	<b>18. SECURITY CLASSIFICATION OF THIS PAGE</b> Unclassified	<b>19. SECURITY CLASSIFICATION OF ABSTRACT</b> Unclassified	<b>20. LIMITATION OF ABSTRACT</b> UU	

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)  
Prescribed by ANSI Std. Z39-18

THIS PAGE INTENTIONALLY LEFT BLANK

**Approved for public release; distribution is unlimited.**

**AUTOMATED BEHAVIOR PROPERTY VERIFICATION TOOL**

John K. Leo  
Lieutenant, United States Navy  
B.S., Austin Peay State University, 2002

Submitted in partial fulfillment of the  
requirements for the degree of

**MASTER OF SCIENCE IN COMPUTER SCIENCE**

from the

**NAVAL POSTGRADUATE SCHOOL  
September 2008**

Author: John K. Leo

Approved by: Mikhail Auguston  
Thesis Advisor

Man-Tak Shing  
Second Reader

Peter J. Denning  
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

## **ABSTRACT**

Computer generated forces (CGF) simulations have entities as actors in their simulation. A type of CGF in which the entities have limited autonomy is semi-automated forces (SAF). The SAF system for this thesis research is OneSAF, a near real-time SAF that offers raw data collection of the entities in a particular simulation scenario. The data collection files vary in size from 500 kilobytes to larger than four gigabytes.

Entity behavior property verification (BPV) is an integral part of SAF simulation software testing. The purpose for this research is to provide immediate feedback to the system user/developer as to what an entity had done in a scenario. From the simulation point of view, it provides answers to questions like “Did the entity route shortest distance to destination?” From the developer’s point of interest, the BPV can provide insight to flaws in the model, such as a vehicle crossing a river where a bridge does not exist.

Automated BPV (ABPV) takes one step further by minimizing “hard coding” of tools that process collection files. ABPV allows portability of the product of this thesis to other systems. ABPV Tools (ABPVT) of this thesis is designed to run in Linux and Windows and will be included in future distributions of OneSAF as an intricate part of the testing suite.

THIS PAGE INTENTIONALLY LEFT BLANK



## TABLE OF CONTENTS

<b>I.</b>	<b>INTRODUCTION.....</b>	<b>1</b>
<b>A.</b>	<b>COMPUTER SIMULATION BASICS.....</b>	<b>1</b>
<b>B.</b>	<b>BEHAVIOR VERIFICATION.....</b>	<b>1</b>
<b>C.</b>	<b>ONESAF OBJECTIVE SYSTEM (OOS).....</b>	<b>2</b>
<b>D.</b>	<b>THESIS ENVIRONMENT AND CONDITIONS.....</b>	<b>3</b>
<b>E.</b>	<b>THE PROBLEM SPACE.....</b>	<b>5</b>
<b>F.</b>	<b>PURPOSE OF STUDY.....</b>	<b>7</b>
<b>II.</b>	<b>MOVE TACTICALLY (MT) SCENARIO .....</b>	<b>9</b>
<b>A.</b>	<b>SCENARIO OVERVIEW.....</b>	<b>9</b>
<b>B.</b>	<b>PHASE I.....</b>	<b>9</b>
<b>C.</b>	<b>PHASE II.....</b>	<b>10</b>
<b>D.</b>	<b>PHASE III.....</b>	<b>13</b>
<b>III.</b>	<b>EMPLACE CONTROLLED MINEFIELD (ECM) SCENARIO.....</b>	<b>19</b>
<b>A.</b>	<b>SCENARIO OVERVIEW.....</b>	<b>19</b>
<b>B.</b>	<b>PHASE I.....</b>	<b>20</b>
<b>C.</b>	<b>PHASE II.....</b>	<b>29</b>
<b>D.</b>	<b>PHASE III.....</b>	<b>38</b>
<b>IV.</b>	<b>CONCLUSION .....</b>	<b>43</b>
<b>A.</b>	<b>SOFTWARE TESTING.....</b>	<b>43</b>
<b>B.</b>	<b>POSITIVES .....</b>	<b>44</b>
<b>C.</b>	<b>NEGATIVES.....</b>	<b>46</b>
<b>D.</b>	<b>FUTURE WORK.....</b>	<b>48</b>
	<b>LIST OF REFERENCES.....</b>	<b>51</b>
	<b>APPENDIX A MOVE TACTICALLY (MT) SCENARIO RUBY SCRIPT.....</b>	<b>53</b>
	<b>APPENDIX B PRESCRIPT.....</b>	<b>61</b>
	<b>APPENDIX C POSTSCRIPT.....</b>	<b>65</b>
	<b>APPENDIX D MOVE TACTICALLY (MT) SAMPLE RAW DATA FILES .....</b>	<b>71</b>
	<b>APPENDIX E MOVE TACTICALLY (MT) SAMPLE REPORTS .....</b>	<b>75</b>
	<b>APPENDIX F MOVE TACTICALLY (MT) PRESENTATION REPORT .....</b>	<b>79</b>
	<b>APPENDIX G AUTOMATED BEHAVIOR PROPERTY TEST (ABPT) DESIGN DIAGRAM.....</b>	<b>91</b>
	<b>APPENDIX H SAMPLE ONESAF ENTITY IN A SCENARIO FILE.....</b>	<b>95</b>
	<b>APPENDIX I SAMPLE ONESAF DATA COLLECTION FILE.....</b>	<b>99</b>
	<b>APPENDIX J TRAC-MONTEREY VERIFICATION PROCESS METHODOLOGY .....</b>	<b>103</b>

**APPENDIX K    ONESAF USERS CONFERENCE ORLANDO FLORIDA**  
**PRESENTATION.....119**  
**INITIAL DISTRIBUTION LIST .....141**

## LIST OF FIGURES

Figure 1	SciTE display showing malformed OneSAF data file.....	11
Figure 2	MT Entity data set.....	15
Figure 3	Emplace Controlled Minefield Scenario Layout. ....	19
Figure 4	Entity Locations. ....	20
Figure 5	An example output of Prescript. ....	25
Figure 6	Entity A's initial movement.....	27
Figure 7	Entity B's initial movement. ....	27
Figure 8	Entity HMMWV's initial movement. ....	28
Figure 9	Example output of Prescript #2.....	33
Figure 10	Entity A's movement. ....	34
Figure 11	Entity B's movement. ....	34
Figure 12	Entity HMMWV's movement. ....	35
Figure 13	Entity A, B, and HUMMWV.....	35
Figure 14	A, B, C - showing HMMWV crossing river.....	37
Figure 15	GUI Interface Design Diagram.....	38
Figure 16	GUI prototype. ....	40
Figure 17	Automated testing suite results. ....	42
Figure 18	Common error message (OneSAF screen capture).....	42

THIS PAGE INTENTIONALLY LEFT BLANK

## **EXECUTIVE SUMMARY**

Chapter I consists of an introduction to the basic ideas of entity based computer simulation and concept of verifying entity behavior through software testing and data mining as well as brief introduction to some of the background issues motivating this study.

Chapter II provides further details of entity behavior verification by applying key concepts to a particular scenario. This section provides lessons learned and in depth look at the problems of entity behavior verification as a whole in an environment where documentation is not provided, or simply does not exist.

Chapter III builds on the knowledge gained from Chapter II, provides prototype development and further exploration of applying the lessons learned from Section II on a more advanced scenario.

Chapter IV provides a summary of the work involved as well as errors discovered in the OneSAF simulation. Chapter IV provides information on where to continue the work provided from this thesis.

Lastly, the products of this thesis are enclosed in the appendixes starting on page 53. Appendixes F, G, J, and K are the reports, presentations and products submitted to OneSAF Verification and Validation (OV&V) from TRADOC Analysis Center Monterey (TRAC-Monterey) that formally presented the work of this thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

## **ACKNOWLEDGMENTS**

I would like to thank Dr. Auguston for his enthusiasm, guidance, inspiration, wisdom and vision for this project. This project had many twists and turns, and often appeared to come to a standstill. His support and encouragement had direct impact on the outcome of this thesis.

I would also like to thank Dr. Shing for his time, invaluable insight, and guidance in producing a polished, finished product.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author, and not necessarily views of OneSAF, TRAC-Monterey, or Computer Science Department at NPS.

THIS PAGE INTENTIONALLY LEFT BLANK



# **I. INTRODUCTION**

## **A. COMPUTER SIMULATION BASICS**

In general terms, a computer simulation is a computer program that simulates the behavior, action, or characteristic of another system. Computer simulations vary from one to another depending upon their purpose, target, and audience. Some common examples of computer simulations are: flight simulators (Microsof® Flight Simulator®), an automated colony of life forms (SimCity®), and weather forecasting (The Weather Channel®).

Flight simulators are classified as virtual simulators in that the input source for guidance and direction is a human interface; humans are the primary actors in these types of simulations and verify when something is correct or not. Automated colonies and weather forecasting simulations process data provided by an external input and produce an output dependant on the inputs; the system is the actor and does not have a feedback system to know if the current status is correct or not.

Particular types of simulation where the entities and the system are actors in the simulation are referred to as constructive simulations. In Semi-Automated Forces (SAF), the actors have some level of autonomy. While the system user generally plans missions and other high level refinements, a SAF entity has basic knowledge of the current mission and performs limited tasks like simple route re-planning when faced with an obstacle.

## **B. BEHAVIOR VERIFICATION**

Software testing is a major component of Software Verification. Software verification tries to answer the question: Are we building the product right?<sup>1</sup> Entity behavior verification extends the definition of verification by trying to answer: Did the entity do what was expected? For example, an entity is tasked to relocate from location x to some location y. In the absence of physical obstacles between point x and point y, the

---

<sup>1</sup> Barry Boehm., "Verifying and Validating Software Requirements and Design Specifications," In *IEEE Software* 1 January 1984, 75-88.

entity is expected to move in a relative straight line from x to y. This is a simple scenario, yet it does have numerous factors to consider such as: time of day, elapsed time, weight, terrain, weather, and condition of the entity (tired, hungry, dehydrated, wounded and etc.).

Since OneSAF system is a real-time military simulation, there are elements of entity behaviors that must reflect real-time, real-world constraints. For example, some real world physical constraints of human entities have are strength, endurance, and speed. The simulation would not reflect real world situation if a soldier was able to carry half his weight for any distance without fatigue setting in, or an aircraft achieving maximum elevation and maximum speed from the ground instantaneously. Every different type of entity has a set of behaviors that should be sensitive to the simulation environment. The behavior of an entity in a particular scenario must be verified

### **C. ONESAF OBJECTIVE SYSTEM (OOS)**

#### **What is OneSAF?**

SAF stands for Semi-Automated Forces. There are many computer generated forces (CGF) simulations. These are often referred to as constructive simulations. CGFs model and simulate combat entities and systems. These entities and systems are actors in the simulation. (In contrast with constructive simulations, the humans are actors in the virtual simulations.) A SAF is a CGF in which the entities have some level of autonomy. For instance SAF entities often react to contact, can do some limited route re-planning when faced with an obstacle, can choose some actions based on their knowledge of the current situation, etc. SAF entities are semi autonomous, because they generally require human operators to do holistic planning, provide goals for goal-directed behaviors, etc.<sup>2</sup>

#### **What is composability?**

Composability is a design philosophy and implementation methodology of OneSAF that enables users to rapidly tailor the simulation to meet the needs of a specific

---

<sup>2</sup> OneSAF Restricted Site  
[http://www.onesaf.net/community/index.php?option=com\\_content&task=category&sectionid=5&id=18&Itemid=36#7](http://www.onesaf.net/community/index.php?option=com_content&task=category&sectionid=5&id=18&Itemid=36#7). Accessed May 5, 2008.

simulation exercise or experiment. The toolbox analogy is useful in explaining composability. When a person wants to fix a light switch, he or she doesn't generally drag the entire work bench from the garage to the site of the repair. Instead, he or she takes the two or three tools from the bench and takes only those. When that person now wants to rebuild the master brake cylinder on his car, he takes a different set of tools from the bench. With OneSAF, the designer of the simulation exercise or experiment can build an instance (or composition) of OneSAF that has only those tools needed for that job. OneSAF also supports battlespace composition. The entity, unit, and behavior composer tools allow the user to modify the way the simulation operates – without recompiling any software.

Composability is enabled through the OneSAF product line architecture framework PLAF. Software is commonly developed in a horizontally layered architecture. In OneSAF, tools are arranged in vertically organized product lines. The interactions between modules in these product lines are controlled through hundreds of defined application programmer interfaces (API's) and data interchange formats (DIFs). This modular architecture enables developers outside of the OneSAF team to develop new modules to replace ones built by Team OneSAF. As long as these new modules comply with the architecture and use the correct API's and DIFs, a user can build a composition that includes their own module rather than one of ours. We feel that our open architecture, open-source software, composability, and modular design will create numerous business opportunities for industry to build replacement modules or new modules with functionality we haven't yet considered.<sup>3</sup>

#### **D. THESIS ENVIRONMENT AND CONDITIONS**

OneSAF's Verification and Validation (OV&V) group had subcontracted entity behavior verification to a small Army organization TRADOC Analysis Center Monterey

---

<sup>3</sup> One SAF Public Site  
[http://www.onesaf.net/community/index.php?option=com\\_content&task=category&sectionid=5&id=18&Itemid=36#9](http://www.onesaf.net/community/index.php?option=com_content&task=category&sectionid=5&id=18&Itemid=36#9). Accessed May 5, 2008.

(TRAC-Monterey) located at Naval Postgraduate School (NPS) as an independent (from OneSAF development) verifying authority. TRAC-Monterey requested the assistance of the Computer Science Department of NPS.

TRAC-Monterey's responsibility was to verify OneSAF's entity behavior and report all findings to OV&V. However, in order to verify entity behavior, scenarios were required. OV&V did not provide the scenarios they wanted to test (an issue further discussed in subsequent chapters). TRAC-Monterey tasks quickly expanded to include scenario design, execute scenario testing, and report all developments to OV&V on a scheduled weekly basis. TRAC-Monterey had been working on the entity verification before the work on this thesis began. They spent most of their resources on building test machines with OneSAF Objective System installed. Then they spent additional months on scenario design and execution. Once the scenario executed properly, they designed a methodology to verify entity behavior. By their developed methodology, TRAC-Monterey was able to satisfy OV&V requirements for three of the 51 scenarios listed according to the TRAC-Monterey Verification Process Methodology Briefing (shown in Appendix J TRAC-Monterey Verification Process Methodology <sup>4</sup>). The major pitfall for TRAC-Monterey's verification methodology was stated on page six of the brief. It states: "Cannot use the Data Collection Specification Tool [DCST]; therefore, quantitative data taken from the Status Window." Translation: "Cannot use the tool provided due to lack of documentation and developer support. Current verification methodology cannot verify entity behavior due to lack of necessary entity data. Can only verify what is observed on screen while the scenario is executed. All quantitative status reported in the Status Window will be reported." Proper entity behavior verification was not possible under current conditions and as slide seven states, "Assumptions. Testing a representative sample of scenarios for each composite behavior is sufficient to evaluate behavior performance."

TRAC-Monterey lacked the appropriate resources to analyze the OneSAF system in order to draw more accurate conclusions on entity behaviors, "Constraints. We do not

---

<sup>4</sup> TRAC-Monterey Verification Process Methodology Briefing, OOS\_Verification\_Monterey.ppt, slide#1, accessed 31 July 2008.

have enough resources (primarily manpower) to verify all 51 composite behaviors... Limitations. Difficulty in collecting output data will affect the accuracy of our results.”<sup>5</sup> Shortly after this brief was delivered to OV&V, TRAC-Monterey representatives requested assistance from the CS Department to derive a workable solution that was not based on the screen outputs of OneSAF simulation. The work conducted on entity verification pertaining to this thesis started thereafter, around late July of 2007 and continued through early March of 2008, when the results of entity behavior verification was presented at the 2008 OneSAF Users Conference (see Appendix K OneSAF Users Conference Orlando Florida Presentation).

While the environment for this work is done with OneSAF Objective System, the intended application system is for any independent data producing software system. The work of this thesis is strictly independent of OneSAF system, OneSAF organization, TRAC-Monterey and associated affiliates. However the work of this thesis directly benefited OneSAF and TRAC-Monterey.

## **E. THE PROBLEM SPACE**

OneSAF is a system of systems consisting of modular components written in Java programming language (estimated over three million lines of Java code) and C for some components. As of version 1.5, the installation package consisted of eight DVDs. OneSAF is intended to run on Microsoft® Windows® and various Linux platforms and included separate DVDs for either platform.<sup>6</sup> The following challenges were discovered and were overcome during the work of this thesis:

1. Lack of system documentation and minimal development support.
2. Instability of the OneSAF Objective System as a whole.
3. Lack of documentation on the Data Collection Module.
4. Data Collection Module instability.
5. Data Collection Module inconsistency.

---

<sup>5</sup> TRAC-Monterey Verification Process Methodology Briefing, OOS\_Verification\_Monterey.ppt, slide#7, last accessed 31 July 2008.

<sup>6</sup> OneSAF version 1.0 -1.4 installation instructions dictates Debian™ core, while version 1.5's instructions are written for Red Hat Linux. This thesis work uses Windows® and Debian™ Linux. Other Linux variations have not been tested in this thesis.

6. Lack of documentation of collected data units and conversions.
7. Inconsistencies of units, i.e., meters, kilometers, global coordinates.
8. Lack of documentation for data tags.
9. Lack of documentation to successfully run data collection.
10. Lack of documentation of data collection files.
11. Lack of parsing/viewing/analyzing tools for the collected data files.
12. Data collection files not adhering to XML schemas.
13. Linux installation was inconsistent.

The OneSAF system is not intuitive (ease of use) nor user friendly (lack of help and feedback). Initial assistance came from members of TRAC-Monterey, a sub-contract U.S. Army group located at Naval Postgraduate School. It was very apparent that OneSAF developers provided minimal guidance as to how to run the system. Nonetheless, a Windows® and a Linux machine had OneSAF version 1.1 installed. Version 1.2 was already on hand but had not been installed. OneSAF delivered version 1.4 (developmental edition) which offered more stable data collection functionality over previous versions. Few weeks after version 1.4 installation and initial testing, TRAC-Monterey received version 1.5. All data collection and analysis for Move Tactically scenario was performed on version 1.4 prior to receiving version 1.5.

After many system instability issues with version 1.1-1.4<sup>7</sup>, a fresh installation of version 1.5 was done in virtual machine environment provided by VMware Work Station 6.x. Had this not been done, a delay of estimated six months was foreseeable.<sup>8</sup>

Performance between the two operating systems (OS) was immediately noticeable. Every task performed in Windows® had a significant delay and lacked “robustness.” OneSAF required 1.2 GB of RAM in Windows®, while in Linux a *mere* 720 MB was sufficient. All work in this thesis was done on the Linux platform, and after successful debugging and testing, ported to the Windows® virtual machine (VM) where it was tested for compatibility.

---

<sup>7</sup> Version 1.3 was never received/installed.

<sup>8</sup> Delays not limited to the following: multiple re-installations, process of loading scenario- running scenario - verifying data files, creating scenarios, modifying scenarios, and etc.

## F. PURPOSE OF STUDY

The purpose of study for this thesis was to provide a solution - if one existed, to verifying entity behavior in a semi-autonomous computer simulation system. The computer simulation used for this study is OneSAF. OneSAF is real-time semi-autonomous system the U. S. Army currently uses in modeling warfare scenarios. As of the latest release version, the developers of OneSAF had not provided any documentation to the system, which ultimately impacted the amount research, testing, and data analysis for this thesis.

The question or problem this thesis wants to explore is: “How do we produce a behavior property verification tool (BPVT) for an unknown system environment?” The second problem is to find a way to automate such tool development (ABPVT). The work of this thesis is to produce tools that would enhance users and developers of data producing systems like OneSAF.

Additionally, the benefits of this study were to gain working knowledge and experience with a few of the industry leading tools and technologies such as XML, Ruby, Java, GNU Plot<sup>9</sup>, CGI, and real-time Java simulation system to list a few. As a direct result of the work for this thesis, a working entity behavior verification solution module for analyzing entity behavior data would be integrated into the testing components of future releases of OneSAF.

The ultimate application of this thesis is to provide one solution to unravel a large undocumented system in order to verify certain behavior characteristics for that system.

---

<sup>9</sup> **gnuplot** is copyrighted, but freely distributable. <http://www.gnuplot.info>, last accessed 21 July 2008.

THIS PAGE INTENTIONALLY LEFT BLANK



## II. MOVE TACTICALLY (MT) SCENARIO

### A. SCENARIO OVERVIEW

The first scenario analyzed was Move Tactically (MT) Scenario. MT is a simple scenario in which the mission of an entity (an M1 Abrams Tank), moves from a preset location to another. This simple mission had certain constraints, for example, the entity was specified not to exceed the maximum speed to travel. This scenario was chosen to study the OneSAF environment and to familiarize with the OneSAF interface. OneSAF system provided minimal or no documentation so all research and work was done in a systemic - trial and error approach.

MT scenario was selected for its simplicity. The idea behind selecting a simple scenario was to show two things: (1) Data collection in OneSAF was possible, and (2) Data files collected were *useable*<sup>10</sup>. If the two criteria were successfully met, then the following were intermediate goals for entity behavior verification: speed, distance to target, and deviation distance from shortest path to destination. The final piece for MT was to present the analysis in presentable reports, summaries and diagrams.

Phase I describes the environment for testing OneSAF on two physical machines. It details the challenges of an undocumented system and manipulating the system to achieve certain tasks. In a way, it is similar to “black-box” testing, differing only in the results. Phase II is the start of data mining. It covers how outputs were gathered and what output OneSAF actually provided. Phase III provides what was done with the gathered data and provides a summary of how MT scenario executed and what was done to provide the analysis in a presentable way.

### B. PHASE I

A selling point for OneSAF is that it has been developed to operate in two personal computer (PC) platforms, Microsof® Windows®, and Linux. TRAC-Monterey had two machines installed with OneSAF 1.1, one on a Debian Linux distribution and the

---

<sup>10</sup> Data files adhered to industry standard file format for either Windows ® or Linux for the type of file, for OneSAF, XML files.

other on Windows® XP Professional SP2. OneSAF developers did not provide the scenario files to TRAC-Monterey. The developers provided a list of scenarios they wanted to have verification testing but did not provide the scenarios, even upon TRAC-Monterey's multiple requests. The developers requesting the verification testing made it clear that they had not worked with, or seen the scenarios in operation, and that TRAC-Monterey's task was to create a working scenario, and verify the results utilizing the limited data collection capabilities of the OneSAF system. The challenges facing TRAC-Monterey were clear: familiarize with the OneSAF interface, create working scenarios, perform entity verification testing, and report the results of testing.

For nearly four weeks, TRAC-Monterey had coordinated with OneSAF developers to have MT scenario execute from start to finish. The scenario itself was not the main cause of difficulty, rather the learning curve to the OneSAF interface was rather steep. Even towards the end of the work on this thesis, many components of the OneSAF interface were still unclear. Thankfully, understanding the interface and being a proficient user of the OneSAF system were not part of the requirements given to TRAC-Monterey.

## **C. PHASE II**

The first data files produced by OneSAF were absolutely unusable. The size of the data files ranged from 600 Megabytes (MB) to four Gigabytes (GB). The data files contained detailed items that pertained to OneSAF, but not anything usable in terms of the mobile entity. The XML parsers available would not open these large XML data files. Each XML editor or browser would throw a parsing error message. Initial thoughts for parsing errors were caused by the file sizes, and for whatever the reason, the editor/browser program was not able to open the large XML data files. It turned out not to be the case; the data files were malformed. For unknown reasons, the data files were not closing properly in accordance with XML schemas - simply, they all lacked closing tags. Standard UNIX editor vi and the newer Vim<sup>11</sup> were able to open the files and simple text searches were possible. Text search is a great tool if one knows what to

---

<sup>11</sup> Vim, <http://www.vim.org/download.php>, last accessed 20 July 2008.

search for. However, since no documentation was provided on the data files, searching for any recognizable text pertaining to the mobile entity proved useless. An XML viewer was needed that would display malformed XML files of any size that OneSAF would create.

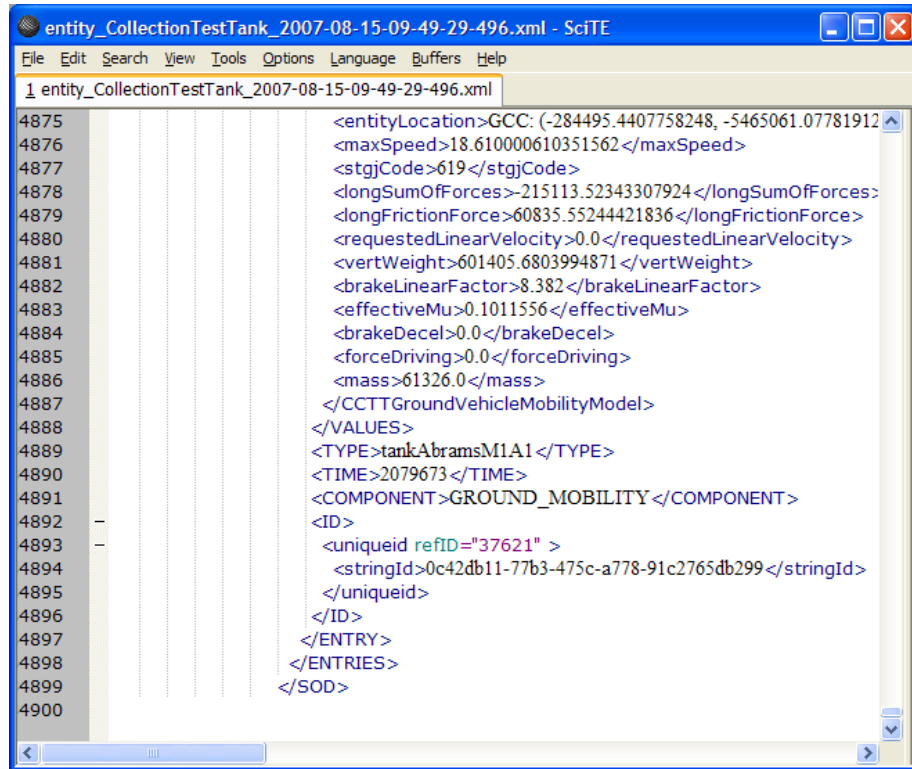


Figure 1 SciTE display showing malformed OneSAF data file.

Windows® version of Ruby<sup>12</sup> installation provides a programming text editor called SciTE<sup>13</sup>. SciTE proved useful for opening and browsing malformed XML data files. Like most XML viewers, SciTE grouped tags so that a single group can be expanded or collapsed; however, unlike most viewers, SciTE would still display unclosed tags. When an XML file is opened in SciTE, the opening tags are located along the left edge of the test area. A faint line extends from an opening tag to the closing tag, if one exists. If a closing tag does not exist, SciTE continues to tabulate additional opening

<sup>12</sup> Ruby: A Programmers Best Friend, <http://www.ruby-lang.org/en>, last accessed 20 July 2008.

<sup>13</sup> SciTE, <http://www.scintilla.org/SciTE.html>, last accessed 20 July 2008.

tags. At the end of the file, if there are missing closing tags, then the last entry will not a line to the left edge. Figure 1 is an example of one of the data files from OneSAF. It illustrates the malformed XML data file produced by OneSAF. The minus signs at line number 4892 and 4893 represents expanded block of tags between an opening tag and a closing tag.

After several weeks working with the large (greater than 600 MB) files, a pattern to the data presented in the XML file was discovered. Entity data was being collected in between intervals of additional OneSAF data. Sometimes the intervals were alternating, while most of the time, the intervals were more sporadic. While the early data files contained entity tags, often they did not have data within the entity tag sets. In another words, it almost seemed like a template, or a place holder. Nevertheless, this was a step in the right direction.

One of the previous efforts of TRAC-Monterey was creating automated scenarios for OneSAF. Automated scenario generation provide means of creating test cases to maximize systems testing more efficiently. Prior to auto-generated scenarios, a tester would have to create a scenario and then modify this newly created scenario manually each and every time a “what-if” was asked. Once modified, the tester would then run the scenario and the cycle of modify-and-run would be repeated until testing criteria would be satisfied. It turned out that a speed parameter of the auto generated scenarios did not match. For example, a maximum speed commanded for the entity was set for 24 kilometers per hour (km/hr). The scenario generator tried to put values like 24, 24.0, and 24.00 in between the speed tags like this: `<speed>24.0<\speed>`. However, when OneSAF creates a scenario, values like maximum speed are converted to meters per second. So, 24 km/hr is roughly 6.67 meters per second (m/s). But OneSAF scenario would not execute the auto-generated scenario unless the values inserted contained certain fixed decimal positions, which varied in value depending on which tag - value pair. In case of the maximum speed value, it was 12 decimal positions - 24 km/hr was represented as 6.666666666667 m/s. Discovering and allowing for this small detail produced consistent entity behavior data to be collected. This discovery is really trivial if documentation was available.

After executing MT scenario dozens of times, the last thing discovered was that the OneSAF data collector did not stop collection even after the scenario completed. The scenario time stamp would continue to be recorded along with the last data known. The data collector was disconnected from the OneSAF interface and therefore never receives the end of scenario message and to stop collection. This explains why the XML data files were not closed properly and why most XML viewers and browsers could not open the data files. In all previous tests prior to this discovery, it was assumed the scenario, and data collection would cease once the scenario ended. This discovery was presented to OneSAF developers at the end of the MT scenario verification testing.

MT scenario was executed 20 to 30 times producing data files from 200 Kilobyte (KB) to 35 MB. The data collection modules of OneSAF was turned off manually once the scenario finished preventing unnecessary large data files (explained towards the end of previous phase). The scenario was run multiple times because the results were never exact copies of another scenario execution. As stated in the introduction, the entities in OneSAF have limited autonomy. The limited autonomy provided for slight deviations of an entity's response. The differences were subtle and viewed from the total scenario, would not seem any different. However, when viewed from the level of entity's position in a given time, the exact locations would differ. After every scenario execution, the data files were compressed and copied to an analysis machine. A 25 MB XML data file was able to compress to 590 KB using a standard zip compression program.

#### **D. PHASE III**

At this point, an XML parser and an XML viewer were needed. While the files could have been processed manually, it would have taken far longer than necessary. If a free tool like SciTE was not available, then an XML viewer would have been necessary to build. As mentioned earlier, free XML viewers and parsers were available; however none of the ones tested would open the malformed OneSAF data collection files.

Using SciTE, the data files were opened and scanned for data that pertained to the entity. The data files contained less than 20 groups of XML tag blocks. An example of a data file is in Appendix I Sample OneSAF Data Collection FileBlocks in this case refers

to a group of tags marked with an opening tag to the closing tag. The good news observed from the data files were the fact that data was collected, that they were sequential, and that they were grouped together in blocks. The bad news is that the blocks were often nested seven or more levels deep and were found all throughout the data file without conforming to any certain pattern.

After assessing the data files, the second step was to parse the relevant data out of the large data files. Since the data files varied greatly in sizes, an intermediate data file was needed. The data files were sequential and the amount of data they contained was not exactly known. The parsing had to accommodate for the worst case, a four GB data file with 500 MB of useable entity data. A parser without writing to an intermediate file to a storage device for this size is possible, but may not be the most efficient.

This is where Ruby programming language became so indispensable. Parsing large XML data files in Ruby using the built-in regular expression capability was extremely fast. Regular expression allows pattern matching instead of an absolute exact match used in many string comparators. If a pattern matches, then action can be taken as to what to do with the matching values. During this data mining stage, when a pattern matches, the data is written to an intermediate file. Once the entire data file is parsed, the intermediate files are closed for further analysis.

The third step is to take a closer look at the parsed data to see if it contains *usable* entity data. It is one thing to have data, and another to have the right data needed for useful analysis. Unfortunately, parsing the data files is a necessary preparatory step. The parsed data was then inputted into Microsoft® Excel®. Graphs and plotting tools in Excel® allowed quick visualization of the raw parsed data. On page 14 of Appendix F Move Tactically (MT) Presentation Report, the diagrams generated from Excel® visually show that the data parsed from the data collection files are in fact contiguous data of an entity. Furthermore, an Excel® plot of the entity in MT scenario shows an actual path of the entity. Thus, preliminary data does show entity tracking throughout the scenario is possible. Now, it is only a question of what is available for collection, and depending on what is available, would dictate what can or cannot be verified. The following (Figure 2) is a block of entity tags from MT scenario.

```

<ENTRY refID="32" >
  <VALUES>
    <CCTTGroundVehicleMobilityModel refID="33" >
      <vehBin>highMobilityTracked</vehBin>
      <requestedLinearAcceleration>2.451675</requestedLinearAcceleration>
      <slope>-0.007060990631857278</slope>
      <brakeForce>0.0</brakeForce>
      <longWeight>0.0</longWeight>
      <currentSpeed>0.42843525442672326</currentSpeed>
      <linearAcceleration>0.0</linearAcceleration>
      <entityLocation>GCC: (-287361.86191023444, -5464905.17195063, 3265213.5139627373)</entityLocation>
      <maxSpeed>18.610000610351562</maxSpeed>
      <stgjCode>619</stgjCode>
      <longSumOfForces>87488.0034785146</longSumOfForces>
      <longFrictionForce>0.0</longFrictionForce>
      <requestedLinearVelocity>9.0</requestedLinearVelocity>
      <vertWeight>0.0</vertWeight>
      <brakeLinearFactor>8.382</brakeLinearFactor>
      <effectiveMu>0.1011556</effectiveMu>
      <brakeDecel>0.0</brakeDecel>
      <forceDriving>87488.0034785146</forceDriving>
      <mass>61326.0</mass>
    </CCTTGroundVehicleMobilityModel>
  </VALUES>
  <TYPE>tankAbramsM1A1</TYPE>
  <TIME>608</TIME>
  <COMPONENT>GROUND_MOBILITY</COMPONENT>
  <ID>
    <uniqueid refID="34" >
      <stringId>0c42db11-77b3-475c-a778-91c2765db299</stringId>
    </uniqueid>
  </ID>
</ENTRY>

```

Figure 2 MT Entity data set.

One piece of information that is not found in the data file is references to other objects in the scenario, or any reference to the mission the entity is assigned to carry out. MT scenario has an entity move from one location tactically to another, without any reference to the target location. The data collection module records information of the entity according to preset time intervals, such as every four milliseconds of simulation running time. According to OneSAF developers, the collection modules were add-on modules that were independent of the simulation system. This can be demonstrated by the fact that OneSAF system can run with or without the data collection modules activated. However, the separation of development of the collection modules has pros and cons.

On the pro side, the separate modular development allows for independence, meaning the data collection should not be able to interfere with the running system and vice versa. This also means that scenario information, like entity location, destination,

mission and so forth are not manipulated by the data collector. The data collector acts as an external entity with a sole purpose of just recording observed data. The separate role of the data collector is critical to verification process because in order to verify an entity behavior in a system, it has to be compared to an external source. Simply, the OneSAF system cannot verify what happens within itself. In order to verify internal behaviors of OneSAF's entities, a credible external recorder must be used.

On the con side, having a separated modular development introduces inconsistencies, anomalies and possible errors. Inconsistencies are like the example mentioned earlier about speed specified in the scenario in units of km/hr while speed is represented in units of m/s in the collection file. In Chapter III, section "B. Phase I," a more annoying inconsistency pertaining to an x, y, and z grid coordinate system is discussed. Anomalies are a bit harder to detect, however, on page 14 of Appendix F Move Tactically (MT) Presentation Report, and the second graph on the page (titled "Raw Slope Data" shows slope of a terrain changing +/- eight to nine meters in a span of few milliseconds. While this drastic change in slope is theoretically possible, the speed of the entity at these outlier points should reflect equally drastic changes as well. However, as the first diagram (titled "Raw Speed") on the same page show, the speed does not drastically change. A positive slope should slow a moving vehicle down and conversely, a negative slope should accelerate a vehicle. When these two diagrams are super imposed, they should complement each other. Since they do not, this is an example of an anomaly. An example of a possible error that can occur when modules are developed separately are their inability to communicate together. While it may not be detrimental to the system, it can be a source of frustration. Take for example the initial file sizes of the data collection files. The initial file sizes were in gigabytes and not in kilobytes or megabytes because the data collection modules did not stop collection upon completion of the scenario.

Since continuous entity behavior collection was possible, the fourth step is what the work of this thesis is all about: verify entity behaviors. As outlined in the Scenario Overview of this chapter, since data were collectable and useable, the intermediate goals of entity verification are entity speed, distance to target, and deviation



distance from shortest path to destination. As stated earlier, the data collection modules were independent of the scenario; therefore, the scenario information had to come from external to the data collection file. The first step of verifying entity behavior was to parse the scenario file. The scenario file was an XML file that OneSAF reads in order to create the simulation. The scenario file contains location coordinates of entities, commands and situation information particular to a scenario. Parsing the scenario file alone was a challenge particularly because of lack of documentation. The key to pairing the scenario to the data collection file were unique string identifications (ID) like “0c42db11-77b3-475c-a778-91c2765db299.” In scenarios where a single entity is being tracked this was an easier feat. In Appendix H Sample OneSAF Entity in a Scenario File, the string ID in the example can be mapped to the string ID in Appendix I Sample OneSAF Data Collection File.

The first step to verify entity behavior was to parse the scenario file for entity location, destination, and commanded speed. If a path for entity was specified in the scenario file, then that path would also be parsed. Second step was then to parse the data collection file. For each of the three behaviors, the Ruby script output a temporary data file. For speed behavior, the output consisted of three fields: time of the recording, current speed, and current slope. The expected values for the speed raw data file were zero m/s to the commanded speed. Obviously, a negative value or values greater than double the commanded value would not be expected. For distance to target, a calculated value derived from current location and destination was recorded with the current time. The distance was a simple calculation of the x and y coordinates of the current location and the destination’s x and y coordinates. The z coordinate was not used for this calculation. The expected value for the distance to target raw data file was a gradual decrease in distance over time. The last behavior to verify was the entity’s deviation distance from the shortest path from start to finish. The shortest path was calculated once and stored, and the distance from this line to the entity’s current location was calculated and recorded to the raw data file. Samples of the raw data files are located in Appendix D Move Tactically (MT) Sample Raw Data Files.

The raw data files alone were a success, however, TRAC-Monterey needed to present the work in a report to OneSAF developers. Once again, Ruby proved indispensable to the task. The output lines for the reports were stream lined within the entire script, from parsing the scenario file to the end of the data collection file. The reason for inline processing was because the data file would only be accessed once, and as stated earlier, since there was no definitive data file size. The Ruby script was modified to accommodate for four reports, one for each of the behaviors and a test summary. Examples of the reports are in Appendix E Move Tactically (MT) Sample Reports Adding the reports to the script had little or no impact on the script performance. In the Linux platform, a 25 MB data file took less than 40 seconds. In Windows®, every test took about twice as long.

The last step to finish the MT scenario was to create visualization of the data. The raw data files were imported into Microsoft® Excel® and graphs were generated. The graphs were an immediate accepted by the OneSAF developers and made a request to TRAC-Monterey to produce a prototype that would automate the verification tests with visualizations as part of the testing outputs. This last step eliminated the option using Excel® for the rest of the work for this thesis. Since OneSAF was developed for Windows® and Linux, another packaged tool that would run in both environments would be required. Another issue with Excel® was the fact that it didn't allow for automated scripts to generate visualizations. The report TRAC-Monterey turned in to OneSAF is in Appendix F Move Tactically (MT) Presentation Report.

The success from MT scenario generated new challenges that will be further discussed in the next chapter.

### III. EMPLACE CONTROLLED MINEFIELD (ECM) SCENARIO

#### A. SCENARIO OVERVIEW

Emplace Controlled Minefield (ECM) scenario is a scenario in which a mobile unit of two or more soldiers load up into an HMMWV, move to a weapons Conversion Cache where they convert a specified number of land mines in a kneeling or sitting posture. Once the mines are converted, the entities load the HMMWV with the converted mines and relocate to weapons Dump Cache where they offload the converted mines. From the Dump Cache, the entities randomly place the mines in a designated minefield area. Once all mines are set in the minefield, the mission is completed. As with the previous Move Tactically scenario, the work for ECM was broken down into three phases.

Phase I covers the first iteration through this scenario. It explores many of the obstacles encountered and what was done to overcome those obstacles. The Ruby scripts from the previous chapter proved inadequate for the data collected in ECM scenario. Phase II is where this scenario gets real interesting. Since the previous approach provided small gains, new scripts were developed to parse the data files. Data files collected are viewable and verification analysis was initialized. Figure 3 below provides the overlay to the EMC scenario and a labeled closer view is in Figure 4.

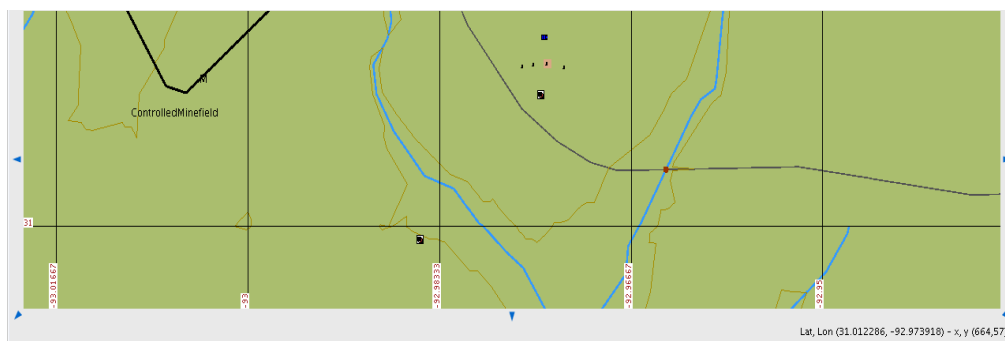


Figure 3 Emplace Controlled Minefield Scenario Layout<sup>14</sup>.

<sup>14</sup> Partial screen capture from OneSAF Objective System with ECM scenario loaded.

For the purpose of this thesis, Emplace Controlled Minefield (ECM) scenario consists of two soldiers (one highlighted - Entity “A”, the other to the immediate left of the highlighted entity - Entity “B”) and five mines. The soldiers are commanded to board the HMMWV (blue square along top edge), then proceed to the Conversion Cache (CC) located along bottom edge. Once at the CC, they are to dismount the HMMWV, convert five mines in the sitting position. Each mine takes two minutes to convert per person but allows division of labor such that two entities could work on one mine thus requiring half the time needed. So, with two entities working on five mines would require the entities to work in the sitting posture for five minutes. Once the mines are converted, the entities load the converted mines into the HMMWV’s cargo area, board the HMMWV, and proceed to the Dump Cache (DC). At the DC, they are to arm the mines and randomly place the mines in the minefield. For the purpose of this thesis, each and every step listed above must be verifiable in terms of location, elapsed time, weight of cargo, cargo (quantity), entity posture, and entity speed.

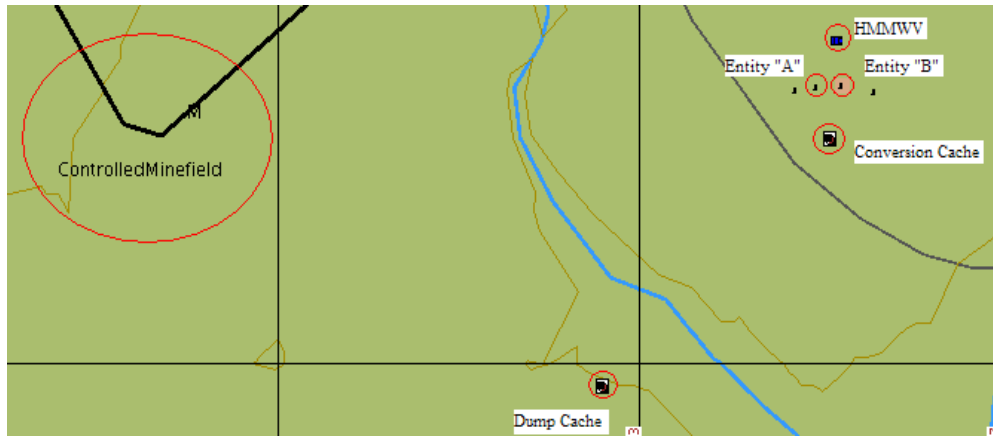


Figure 4 Entity Locations<sup>15</sup>.

## B. PHASE I

Initially, the ECM scenario was created and tested in OneSAF version 1.4. After several unsuccessful attempts to have ECM scenario execute on the 1.4 platform, the 1.4 platform was abandoned and the testing migrated to OneSAF 1.5. Most notable among the test results were things like no output to the data collection files, scenario failed to

---

<sup>15</sup> Partial screen capture from OneSAF Objective System with ECM scenario loaded.

terminate, collection files did not close properly, collection files were extremely large (greater than two Gigabytes [GB]), data on some entities would not be collected while others would - a random selection of which entity the collection module would collect on. Migration from OneSAF 1.4 to 1.5 was easier than previous installations of OneSAF on both Windows® and Linux platforms. However, the physical machines became more unstable and the decision was made to have both machines rebuilt from a clean formatted hard drive. Learning from the frustrations from Move Tactically scenario, a more practical solution was needed. The progress of this thesis and the efforts of TRAC-Monterey ceased until OneSAF 1.5 was stable.

After a week passed without a working OneSAF platform, focus for this thesis was redirected to constructing virtual machines (VM), one for Debian® Linux Version 4 and one for Windows® XP SP2. Both VMs had OneSAF 1.5 loaded and running in three days. The stable VM images were then archived using lossless compression software called WinRAR®. VMs proved indispensable for the work of this thesis in both saving time and proving means of testing several different configurations without concern for platform instability. With this new found success, all tests were conducted in VMs.

In OneSAF version 1.5, the data collection modules were more stable according to the developers. The developers did not define or clarify what they meant by “stable” because from data collection and testing point of view, all testing conducted on 1.5 were similar in results to the 1.4 version.

The first step was to run ECM scenario with data collection modules enabled. The scenario was executed twenty to thirty times and the data files saved. The only data files that were of use were the files that were greater than two Megabytes (MB) and less than 80 MB. Files that were less than two MB were files that were initialized for the scenario, but did not contain any entity data while the scenario was executed. Conversely, files that were greater than 80 MB contained fillers like heartbeat information of the scenario.

Data collection failure was defined as: When data collection was specified for a particular entity and data was not collected for that specified entity during any test

execution. During all test executions, never once a data file was created that was not specified. In another words, the only “randomness” of data collection files were from specified entities. If the entity was not specified, collection on an unspecified entity never occurred. From the initial 20 to 30 tests, only the data collection files from successful collection were used and analyzed for this thesis. As in the previous scenario, the XML data files had some structure but were hard to understand the pattern of the re-occurring tags. As a matter of fact, it can be said there were not any specific ordering or sequencing of the data sets. The work developed in the previous section provided a starting point but no useful data were obtained. It was apparent a new XML parser, analyzer, and data presentation were needed.

One familiar XML tag from the previous chapter was “entityLocation.” However, initial parsing on this known tag produced incomplete scenario data. The scenario produced 2-80 MB files for each entity the data collection was initialized for. Yet, the location of the entity was only reported from the initial location of each entity to that entity’s first stop in the scenario. For example, the soldiers’ locations were recorded from their initial location to the location of the HMMWV. The HMMWV’s location was logged until it reached the Conversion Cache (CC). Based on incomplete location data of the entities in ECM scenario, it was apparent the data collection files contained unusable and repeated information (continuous filler information polled every two millisecond). The data collection files were visually inspected to confirm this observation.<sup>16</sup>

The Data Collection Specification (DCS) files associated with this scenario were not collecting the entities behavior after the entity reached its first waypoint. In the case of the light infantry entities, the first waypoint was the location of the HMMWV. In the case of the HMMWV, it was the location of the Conversion Cache (CC). The tools developed from the previous section proved inadequate for the current scenario data files.

---

<sup>16</sup> A sample of the repeated data marked by time units in the appendix.

The following is a data sample derived from the data collection files used to generate the graphs below.

```
<ENTRY refID="95" >
  <VALUES>
    <ICMobilityModelMR refID="96" >
      <slope>4.3450884128308633E-4</slope>
      <timeResting>0.0</timeResting>
      <bin>normallyLoadedIC</bin>
      <entityLocation>GCC: (-283859.98297301884, -5463960.4266113555, 3267075.8437014534)</entityLocation>
      <entityType>ICFullyLoaded</entityType>
      <maxSpeed>6.0</maxSpeed>
      <stgjCode>619</stgjCode>
      <linearVelocity>1.3893916876222037</linearVelocity>
      <fuelStatus>3600.0</fuelStatus>
      <useEnergyEquations>>false</useEnergyEquations>
      <inFreeFall>>false</inFreeFall>
      <postureState>Standing</postureState>
      <timeMoving>0.0</timeMoving>
      <maxSustainableEnergyLevel>6900.0</maxSustainableEnergyLevel>
      <climbModeOn>>false</climbModeOn>
      <mPenaltyModifier>1.0</mPenaltyModifier>
    </ICMobilityModelMR>
  </VALUES>
  <TYPE>ICFullyLoaded</TYPE>
  <TIME>17884</TIME>
  <COMPONENT>GROUND_MOBILITY</COMPONENT>
  <ID>
    <uniqueid refID="97" >
      <stringId>3202a396-1885-47d5-a54c-ec37fe3f1c35</stringId>
    </uniqueid>
  </ID>
</ENTRY>
<ENTRY refID="98" >
  <VALUES>
    <SuppressionSpeedLimit refID="99" >
      <beingSuppressed>>false</beingSuppressed>
      <dayNight>Day</dayNight>
      <entityType>ICFullyLoaded</entityType>
      <maxSpeed>1.67</maxSpeed>
    </SuppressionSpeedLimit>
  </VALUES>
  <TYPE>ICFullyLoaded</TYPE>
  <TIME>17884</TIME>
  <COMPONENT>MOBILITY_CONTROLLER</COMPONENT>
  <ID>
    <uniqueid refID="100" >
      <stringId>3202a396-1885-47d5-a54c-ec37fe3f1c35</stringId>
    </uniqueid>
  </ID>
</ENTRY>
```

The following XML is what fills the bulk of the data file:

```
<ENTRY refID="7370" >
  <VALUES>
  </VALUES>
  <TYPE>null</TYPE>
  <TIME>179604</TIME>
  <COMPONENT>null</COMPONENT>
  <ID>
    <uniqueid refID="7371" >
      <stringId>3202a396-1885-47d5-a54c-ec37fe3f1c35</stringId>
    </uniqueid>
  </ID>
</ENTRY>
```

```

<ENTRY refID="7372" >
  <VALUES>
  </VALUES>
  <TYPE>null</TYPE>
  <TIME>179604</TIME>
  <COMPONENT>null</COMPONENT>
  <ID>
    <uniqueid refID="7373" >
      <stringId>3202a396-1885-47d5-a54c-ec37fe3f1c35</stringId>
    </uniqueid>
  </ID>
</ENTRY>

```

... (The rest of the data file is filled with this repeated entry.)

Initial view of the data looked promising to have the data required for analysis. The tools developed in the previous section were not able to parse the data from the ECM scenario because the data file format was very different. In order to dissect the data files, a Ruby script was needed to parse the data files. Similar to the scripts from the previous chapter, a script was needed to display the data format; in particular, the tags associated with the new data files. The understanding gained from Move Tactically (MT) scenario was that the format of the data files was supposedly constant. Viewing the data files from ECM, however, shows the data files to be very different. Thus a more dynamic script was needed to be able to parse any of the XML data files generated by OneSAF and present the format of a particular data file to the data analyzer. A Ruby script called Prescript (Appendix B Prescript) was created to do the task of parsing data files created by OneSAF. The purpose of this parser was to: (1) detect bad data files, (2) provide an entity data set tags if the data file was good. The performance of Prescript was amazing, even in Windows® platform. Almost instantaneously, the result of Prescript is returned. On command line execution, the Prescript receives the name of the XML data file. If the data file contains usable entity data, it returns the XML file name, the entity name and all associated tags with the tag data types. The tag data types are produced by Prescript and are informative information to users. The bulk of the work for Prescript came from the Ruby scripts from the previous section (MT). Ruby provides a powerful and extremely fast regular expression parsing capability. All XML data files from OneSAF were parsed within seconds, regardless of the size of the file. The following screen (Figure 5 shows the output of Prescript on an infantry entity of the ECM scenario.



```

entity_A_2_MechInf_Plt_ANTIARMOR_-ICT4_2008-05-17-24-12-37-517.cut.xml
  ICMobilityModelMR
    slope(float)
    timeResting(float)
    bin(string)
    entityLocation(float)
    entityType(string)
    maxSpeed(float)
    stgjCode(integer)
    linearVelocity(float)
    fuelStatus(float)
    useEnergyEquations(string)
    inFreeFall(string)
    postureState(string)
    timeMoving(float)
    maxSustainableEnergyLevel(float)
    climbModeOn(string)
    mPenaltyModifier(float)
    :

```

Figure 5 An example output of Prescript.

Prescript (Appendix B Prescript) takes the XML filename as the only argument from the command line. If useable entity data is encountered during the parsing of the data file, Prescript would output the above screen to the terminal. However, if unusable data is encountered, Prescript would report an error with the data file. During early preliminary tests, when Prescript reported an error with a data file, those data files were processed manually with Ruby's incorporated editor SciTE<sup>17</sup>. SciTE proved useful for OneSAF data files because none of the XML files were closed with proper XML closing tags. Much like HTML, proper XML has an opening tag and at some point later followed by a closing tag. For example, a start tag: <TAG> would have </TAG> to represent the end to this tag. OneSAF does not properly close their XML data output files which created a problem of viewing these files with many standard third party XML browser tools. Standard editor in Linux like *Vi* and *Vim* proved useful to viewing the malformed XML data files. However, it was tedious and painstaking slow because these tools only provided simple text searching capability, useful if one knows what to search for,

---

<sup>17</sup> SciTE Version 1.72 Jan 15 2007 by Neil Hodgson. Dec 1998-Jan 2007 <http://www.scintilla.org>

worthless if one did not. SciTE was able to group tags when tags had a closing tag, and revealed unclosed/missing tags. All data files generated by OneSAF were missing closing tags<sup>18</sup>.

The main work done by Prescript (Appendix B Prescript) was to identify a valid data set. One of the reasons why Ruby was selected for the scripting language was because of its ability to parse regular expressions swiftly. The longest matching set of expressions was returned, with their matching set of data types such as an integer, float or string.

Prescript provided a fast view of the tags that was found in the data files. Once the tags were identified, a method was needed to process the data contained in the tags. The next step after Prescript was to produce a way to parse useable data out of the large XML data files. Another Ruby script was created for this purpose. This script was called Postscript (Appendix C Postscript). In order to use Postscript, a minimum of three parameters were required: data filename, entity name, and a tag within the entity's set of tags produced by Prescript. When the data files contained data from multiple entities, the name of the entity was critical to distinguish one entity data set from another. Otherwise, tags within the entity set would conflict.

Postscript parses an entire data file for the entity name. Upon a match, the tags following the entity name are parsed for any of the matching tags provided from the command line. When a match for the parameter is found, the data is sent to a temporary text file. Once the entire data collection file is parsed, Postscript calls GNU Plot to plot the data in the temporary data text file. GNU Plot was chosen for this thesis because of performance and platform independence. Although the binary executable is different, the commands are the same whether on a Windows® or Linux. Furthermore, GNU Plot can be wrapped in a Java executable jar file for final deployment, more on deployment in Phase III. The following examples are of Postscript (Appendix C Postscript) processing

---

<sup>18</sup> As of OneSAF 1.5. Later releases had not been tested, however preliminary data view from 2.0 also shared the same characteristics as all previous releases.

three data collection files from OneSAF during initial ECM scenario testing. Postscript with “entityLocation” on the first set of data files from ECM scenario produced the following graphs:

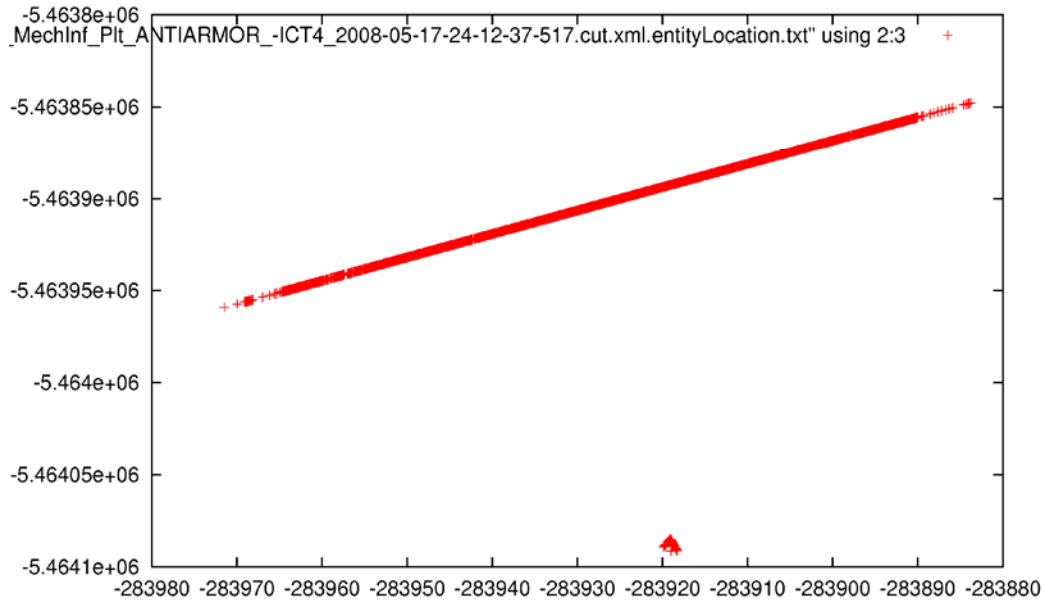


Figure 6 Entity A's initial movement.

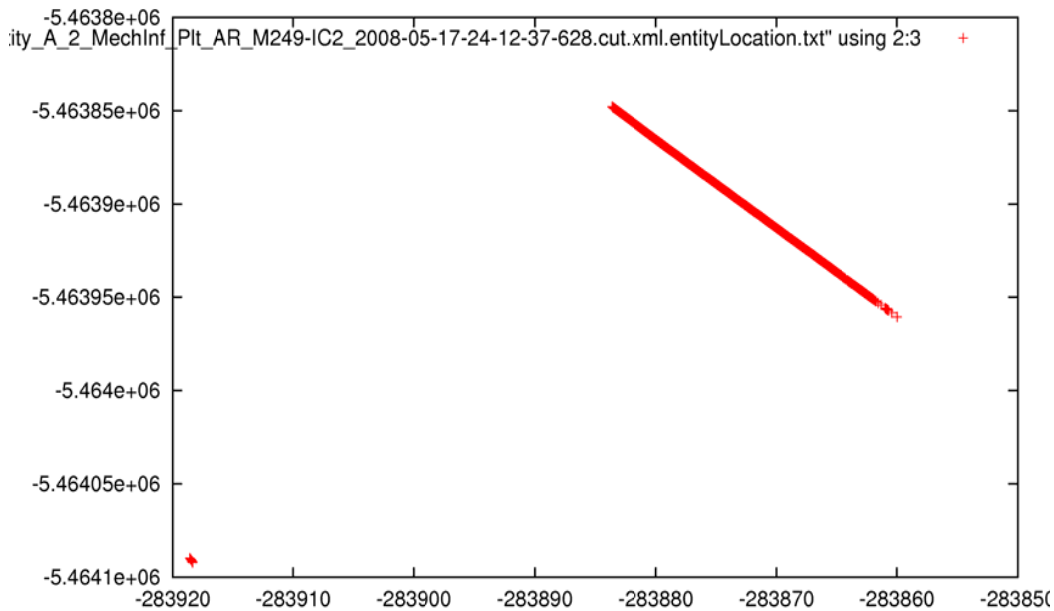


Figure 7 Entity B's initial movement.

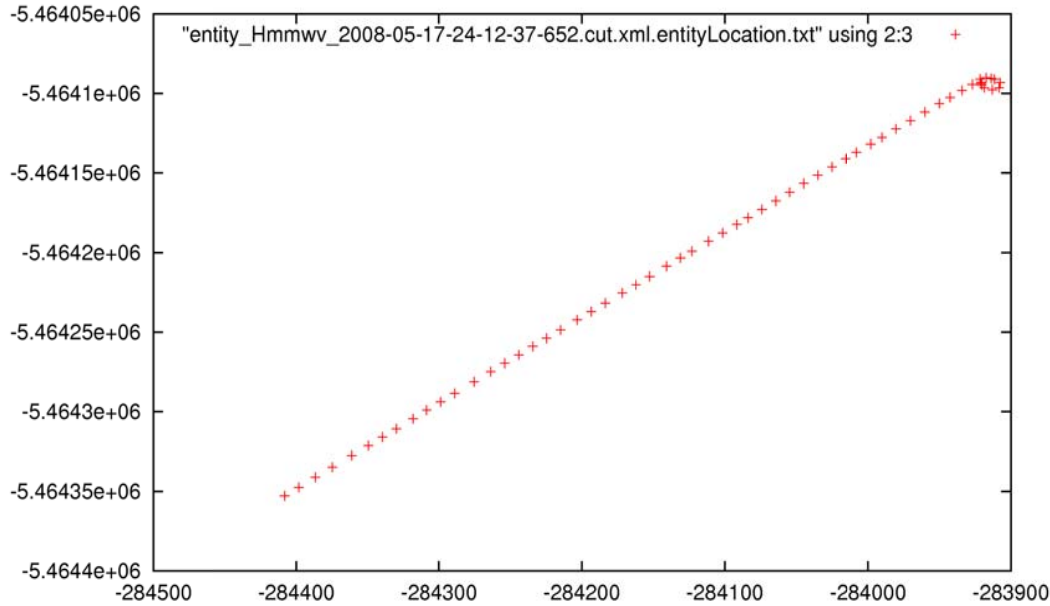


Figure 8 Entity HMMWV's initial movement.

The above figure plots (Figure 6, Figure 7, and Figure 8) are x and y coordinate plots of the three mobile entities in ECM scenario. The Y-axis is presumed latitude coordinate and the X-axis presumed longitude coordinate. The filename of the data file that generated the plots are printed in quotes at the top of each plot. While the coordinates really do not provide much information, when plotted they do provide a visual representation of the entity's movement, velocity and speed. The data for all of these plots were collected at four milliseconds intervals. The closer the points are to one another, represents slower velocity. Acceleration and deceleration are visually represented by the plots gradually distancing themselves either apart or closer together. Velocity and acceleration can be visually compared from the human entities in Figure 10 and Figure 11 alongside the motorized entity such in this case, the HMMWV in Figure 12. Acceleration and speed of the entities are characteristics of entity's behavior, which can be verified using the work of this thesis.

Entity A's movement (Figure 6) went from lower left to upper right, entity B's movement went from lower right to upper left, and the HMMWV's movement from upper right to lower left. The plots of the two entity soldiers also contained additional

plots around point (-5.4641e+06, -283920). This was assumed to be the point where the entities disembarked the HMMWV and walked to the Conversion Cache (CC). However, the end points plotted by the HMMWV did not correspond to locations near the end points of the soldiers. The only conclusion that can be made is that the data provided in the set were inconclusive. The same data was collected for the first 10-20 iterations of the scenario.

As stated earlier, OneSAF did not provide any documentation on their system, so the exact conversion from the scenario map to the x and y coordinates were not clear. The HMMWV's location, however, did match (Figure 8 of HMMWV tacking) at least until it reached its first destination, Conversion Cache (CC). From CC waypoint, entity location (and any other useful data) was not available in any of the data files.

The coordinates were also different from the simulation screen compared to the data output. For example, the latitude-longitude (lat/long) starting location for entity B according to the scenario diagram is approximately (31.01, -92.97), however, the data output reflects roughly (-283860, -5.46e06). The exact conversion of the data output to the simulation is not defined, since documentation was not available. What is clear is that a conversion is applied at some point. It is also of note to think the scenario could be replayed based on the data output files.

## **C. PHASE II**

Two issues became apparent: (1) the data collector provided an incomplete set of data and (2) the format of the data files were different from the data files from the previous scenario. The methods developed in Chapter II of this thesis would not be effective for the format of the current data collection files.

All data collection specification had been executed using a built-in tool of the MCT called Collect Analysis Data. This tool did not allow data customization to collect, only the specific entity in which to collect. However, one of the OneSAF Desktop applications did allow limited data collection specification called, Data Collection Specification Tool (DCST). This was discovered by trial and error, and luck.

Like most features of OneSAF, the learning curve to use DCST was quite steep and extremely non-intuitive. OneSAF simulation system operates by scenario files. A new scenario file must be named "Scenario.xml". When a scenario is modified OneSAF saves the modified scenario file as "ScenarioX.xml". The "X" represents sequential number starting with one. OneSAF will allow the latest scenario file to load (for example a scenario file like "Scenario23.xml"), and run the simulation. However, because the latest scenario file contains a number at the end of the scenario file name, the data collection would not collect data. This was caused by the DCST. DCST only recognized a scenario file named "Scenario.xml." Even though the OneSAF system by default saved updated and/or modified scenario files with a sequential number affixed to the end of the filename, the DCST would not recognize it. The DCST basically did the following: parsed the scenario file named "Scenario.xml" for all entities that collection could be specified. The operator then selected an entity and selected either general or AAR<sup>19</sup>. The DCST would then save the specification file. The specification files only work with a loaded scenario file named "Scenario.xml." In order to specify the latest scenario file, that file needed to be renamed as "Scenario.xml." Then, the operator would have to specify collection files using the DCST. Once new collection files are created, they had to be selected with the Collect Analysis Data (CAD). The CAD allows renaming of the actual data collection file output. This must be done for all entity data is to be collected. In order for the collection process to begin, the scenario must be saved. However, the newly saved scenario file now has a numeric value added to the end of the filename. The newly saved scenario file must be renamed as "Scenario.xml", and must be reloaded. This process was critical to the entire data collection process for OneSAF. Any missed step would result in failed collection. Even if all steps were followed, there was still a bit of randomness to what entity would be collected on. Luckily, about 1/5 of the tests had full data collection files.

---

<sup>19</sup> AAR - OneSAF does not specify what AAR is. AAR in this study was presumed to mean "After Action Review/Report." For the purpose of this study, the name did not contribute anything.

When it worked, the DCST provided a systemic collection of more data than the pre-built Collect Analysis Data (CAD) provided. The Prescript (Appendix B Prescript) from Chapter II was modified to accommodate the different data format that DCST AAR outputted.

DCST allowed two settings for data collection, general and AAR<sup>20</sup>. After trial and error, the general settings proved similar results as stated in previous section. However, the AAR collection provided more usable data. An example of what AAR collection provided is shown below.

```
<ENTRY refID="128" >
<VALUES>
<AAR_EntityData refID="129" >
<contaminant>null</contaminant>
<contaminationStatus>null</contaminationStatus>
<velocity>
<net.onesaf.core.services.data.dm.rdm.phys.Tuple3dStruct refID=" »130 » >
<z>1.1828997366419163</z>
<y>0.7149462845717757</y>
<x>-0.14730872069896997</x>
</net.onesaf.core.services.data.dm.rdm.phys.Tuple3dStruct>
</velocity>
<posture>16</posture>
<catastrophicKill>false</catastrophicKill>
<mobilityKill>false</mobilityKill>
<orientation>
<net.onesaf.core.services.data.dm.rdm.phys.Quat4dStruct refID="131" >
<z>-0.37976037115501066</z>
<w>0.31534386085032484</w>
<y>-0.639351867294955</y>
<x>-0.589550252065652</x>
</net.onesaf.core.services.data.dm.rdm.phys.Quat4dStruct>
</orientation>
<initialContaminationTime>0</initialContaminationTime>
<superiorID>
<uniqueid refID="132" >
<stringId>2d6f971a-2929-4750-8b4a-38da06253c0e</stringId>
</uniqueid>
</superiorID>
<entityType>ICFullyLoaded</entityType>
<contaminationConcentration>0.0</contaminationConcentration>
<modelName>IC, Loaded</modelName>
<damage>NO_KILL</damage>
<mounted>false</mounted>
<damageString>Healthy</damageString>
<communicationKill>false</communicationKill>
<incapacitatedKill>false</incapacitatedKill>
<firepowerKill>false</firepowerKill>
<unique_id>
<uniqueid refID="133" >
<stringId>3202a396-1885-47d5-a54c-ec37fe3f1c35</stringId>
</uniqueid>
</unique_id>
<affiliation>SUSPECT</affiliation>
<location>
```

---

<sup>20</sup> AAR - OneSAF does not specify what AAR is. AAR in this study was presumed to mean "After Action Review/Report.." For the purpose of this study, the name did not contribute anything.

```

<net.onesaf.core.services.data.dm.rdm.phys.Tuple3dStruct refID="134" >
  <z>3267083.302499887</z>
  <y>-5463955.922321388</y>
  <x>-283860.911732261</x>
</net.onesaf.core.services.data.dm.rdm.phys.Tuple3dStruct>
</location>
<name>A/2/MechInf_Plt:AR M249-IC2</name>
<parent>
  <encodableReference refID="0" />
</parent>
</AAR_EntityData>
</VALUES>
<TYPE>ICFullyLoaded</TYPE>
<TIME>9532</TIME>
<COMPONENT>null</COMPONENT>
<ID>
  <encodableReference refID="133" />
</ID>
</ENTRY>

```

The revised Prescript (Appendix B Prescript) output of AAR data collection file produced an output shown in Figure 9. The tags following the entity are listed as they appear in the data file. Unlike previous iterations, the new AAR Prescript output contained four tags with child tags. For example, entity's location from the previous data files was listed as "entityLocation" without any child tags. However, in the AAR data file, the same location data were listed as just "location" with x, y, and z child tags. To make matters a bit more complicated, three entity tags had x, y, and z child tags: velocity, orientation, and location. The Prescript from previous work would not have been able to distinguish between these three sets of child tags. Prescript had to be modified so that if an x, y, or z tag was specified, a default parent tag had to be specified. The default parent tag specified in Prescript was "location" since most of the entity verification centered on the entity location in the simulation. In order to specify child tags, the child tag must immediately follow the parent tag. For example, in order to specify velocity's x tag, the command parameters would be as follows: "<filename> <entity name> <velocity> <x>". The tags with child tags are identified differently from other tags by a set of parenthesis around the tag name followed by an asterisk. Likewise, the child tags are identifiable by an asterisk following the child tag data type.



```

aa_2008-02-29-17-45-36-81.xml
  AAR_EntityData
    contaminant(string)
    contaminationStatus(string)
    (velocity)*
    z(float)*
    y(float)*
    x(float)*
    posture(integer)
    catastrophicKill(string)
    mobilityKill(string)
    (orientation)*
    z(float)*
    w(float)*
    y(float)*
    x(float)*
    initialContaminationTime(integer)
    (superiorID)*
    stringId(integer)
    entityType(string)
    contaminationConcentration(float)
    modelName(string)
    damage(string)
    mounted(string)
    damageString(string)
    communicationKill(string)
    incapacitatedKill(string)
    firepowerKill(string)
    (unique_id)*
    stringId(integer)
    affiliation(string)
    (location)*
    z(float)*
    y(float)*
    x(float)*
    name(integer)
    (parent)*
    :

```

Figure 9 Example output of Prescript #2.

The following figures (Figure 10, Figure 11, Figure 12, and Figure 13) are the outputs of Postscript (Appendix C Postscript) on the DCST AAR data collection files. In all the figures, the entities are roughly located in the upper right quadrant of the plots. Their destination is to the lower left quadrant of the plots.

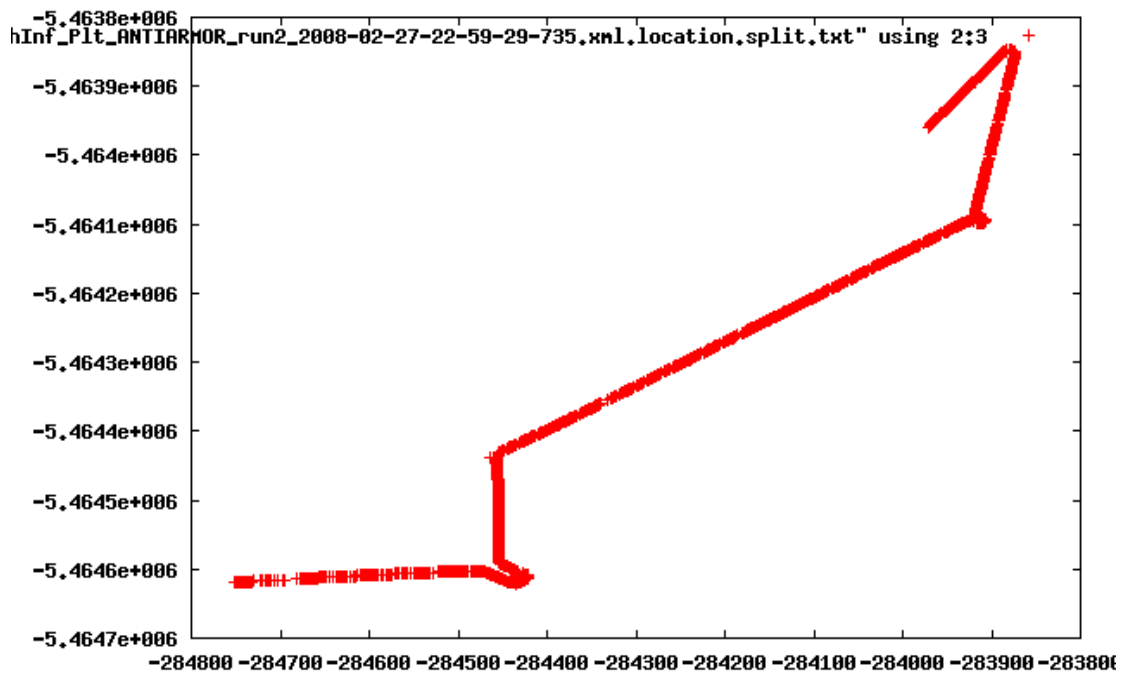


Figure 10 Entity A's movement.

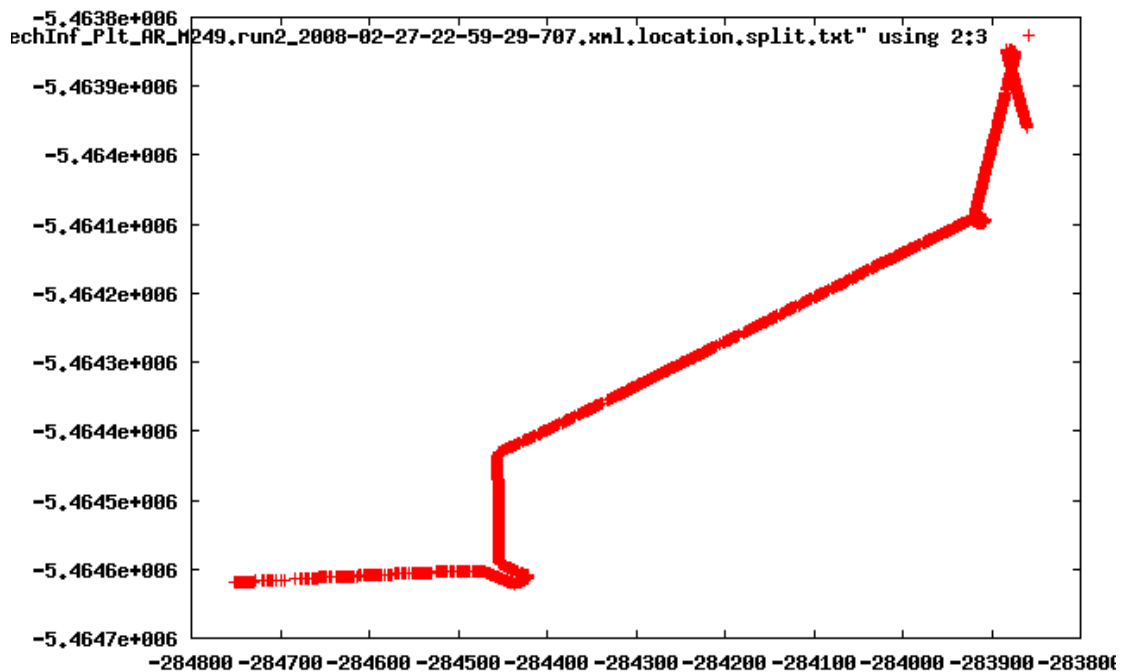


Figure 11 Entity B's movement.

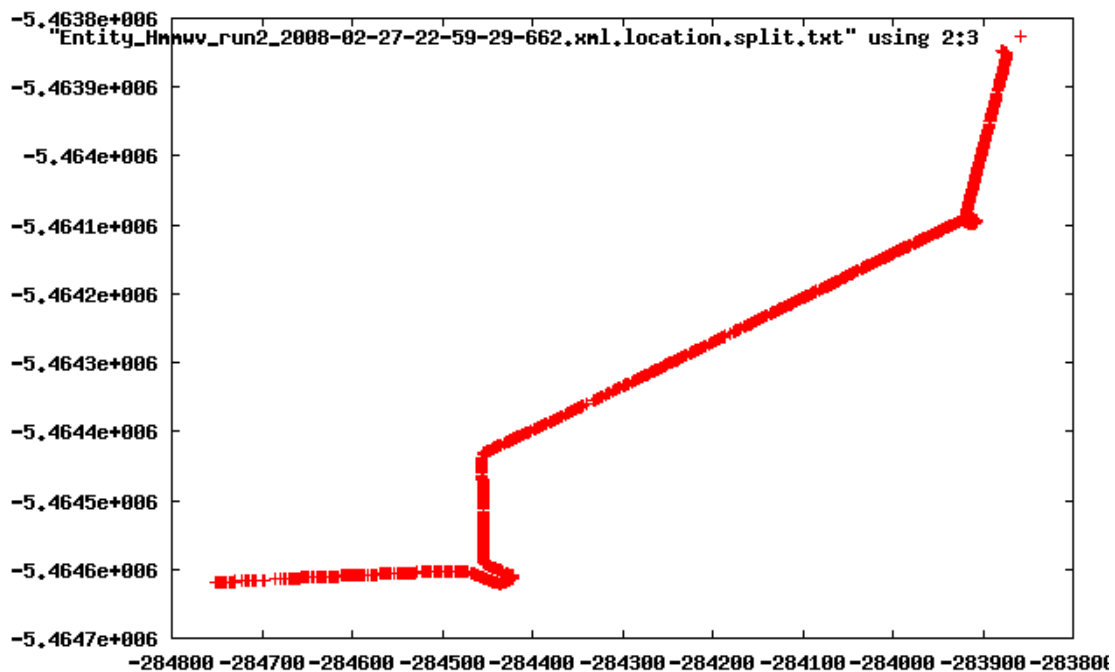


Figure 12 Entity HMMWV's movement.

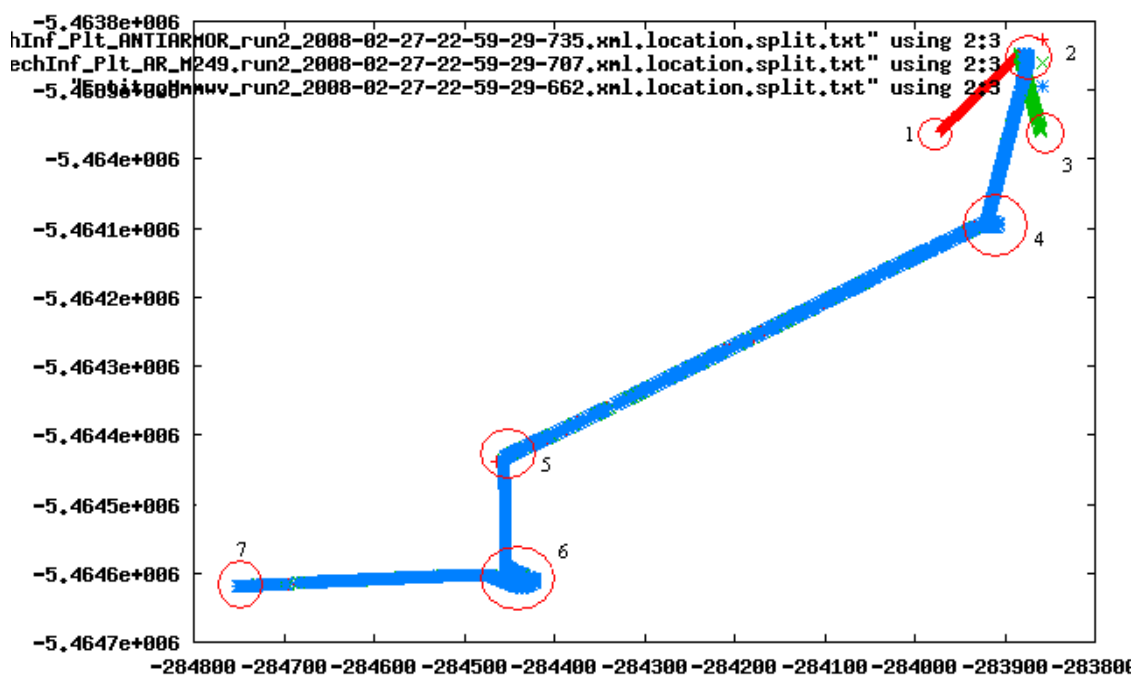


Figure 13 Entity A, B, and HUMMWV

Figure 13 is a composite of the three entities in one of the ECM scenarios. The composite image shows clearly that the path each entity traveled was what was expected. Points 1, 2 and 3 are the initial locations of the three entities. Figure 6 and Figure 7 shows what would be equivalent to a smaller scale of the entities initial movement from points 1 and 3 to point 2. Point 4 is the location of the Conversion Cache. Point 5 is where the HUMMWV approaches the river, and travels south in search of a suitable crossing location. In this particular simulation execution, the entities traveled south, while in other iterations of this simulation the entities traveled north. Ironically, when the entities travel north, they do not cross the river. Point 6 is the point where the HMMWV travels alongside the river bank, then crosses the river, then travels alongside the opposite river bank, then turns west to the Dump Cache (DC) located at point 7. This simulation clearly revealed a programming flaw in OneSAF's mapping system. The point where the HMMWV crossed the river is where two pieces of the map adjoin. The terrain has mouse event listener that highlight and identify the objects when a mouse click is detected, shown below in Figure 14C (note the images are not to scale). During the work of this thesis, programmatic "bugs" similar to this were discovered and reported to OneSAF developers.

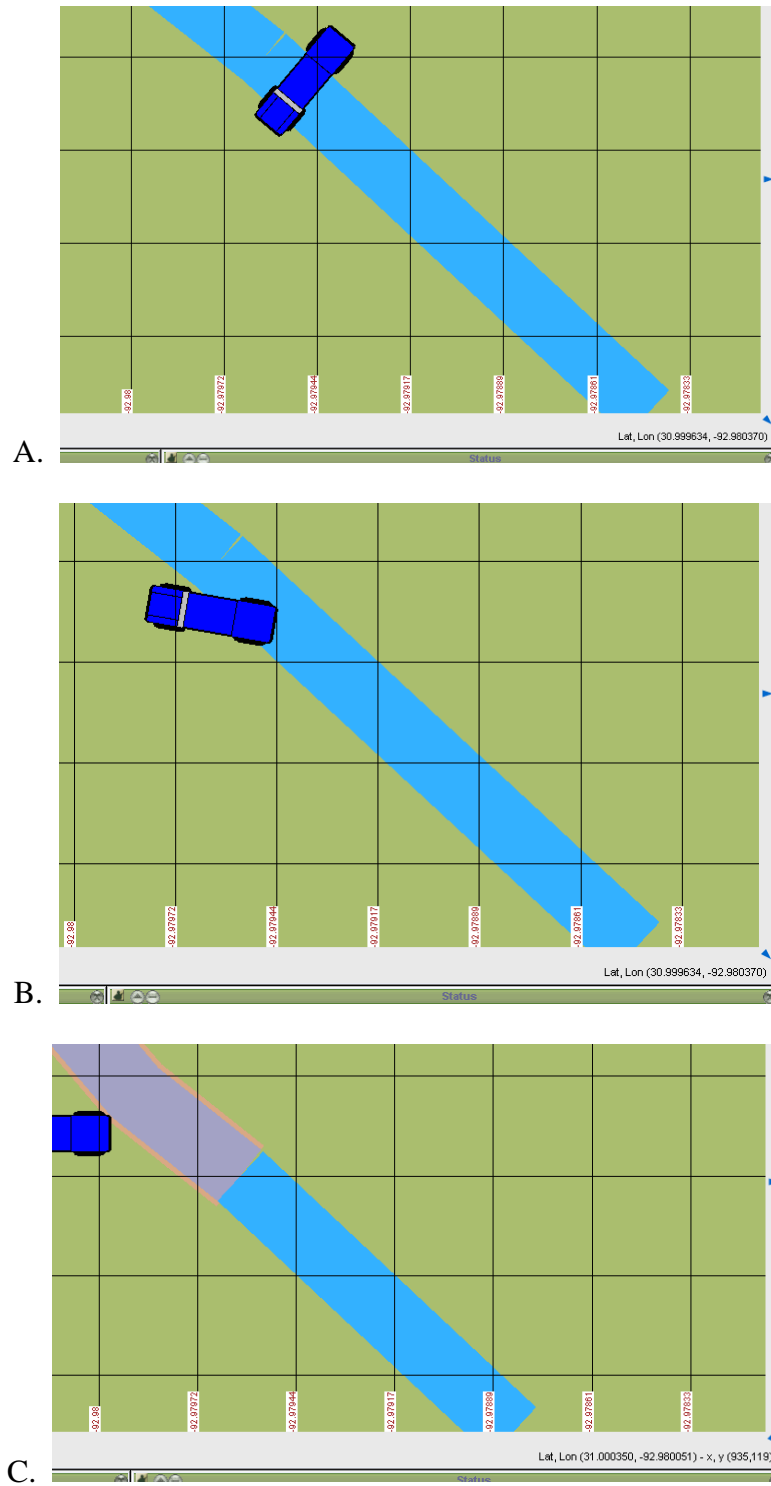


Figure 14 A, B, C - showing HMMWV crossing river.<sup>21</sup>

<sup>21</sup> OneSAF screen captures.

#### D. PHASE III

OneSAF developers requested a graphical user interface (GUI) that would allow OneSAF users and developers to use the tools developed in this thesis. While TRAC-Monterey was requested by OV&V for this task, the prototype for the verification GUI was done for this thesis. Since most of OneSAF was developed using Java, the logical solution for a verification GUI was also to create using Java. OV&V team provided results of MT scenario (from Chapter II of this thesis) to OneSAF developers. The developers were very pleased with the results but were not enthusiastic about using Ruby. Fortunately Ruby has been implemented for Java as JRuby, and available open source<sup>22</sup>.

### GUI Interface Design Diagram

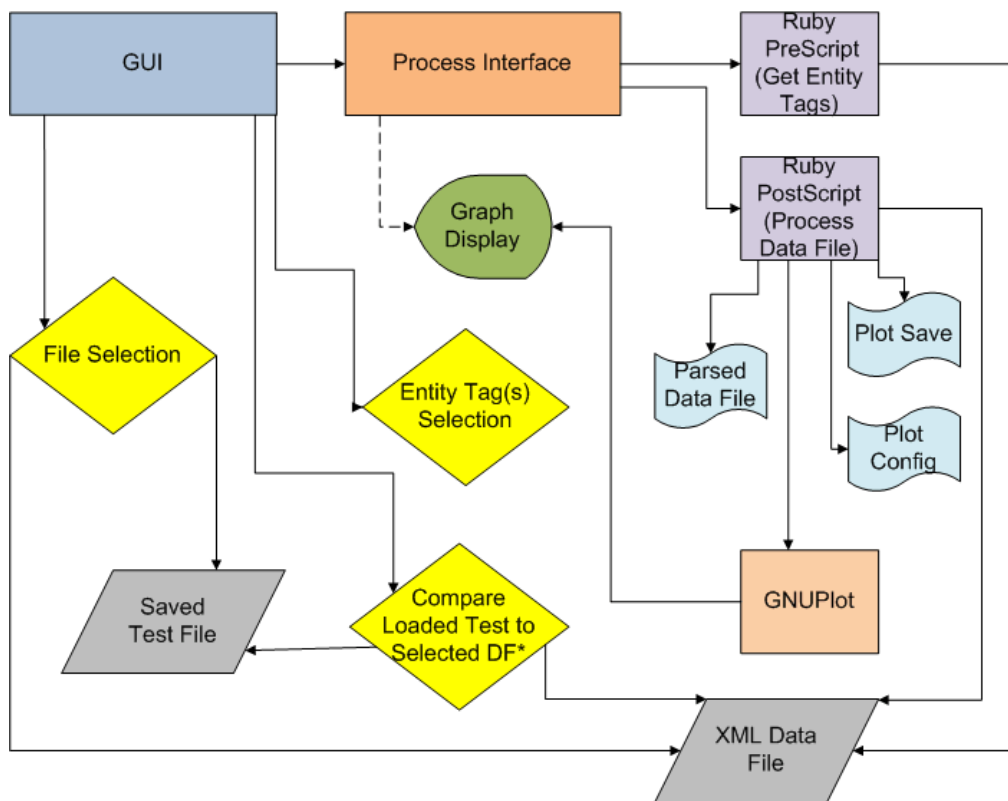


Figure 15 GUI Interface Design Diagram

<sup>22</sup> JRuby, <http://jruby.codehaus.org/>, Accessed 20 July 2008.

It is important to point out that the GUI development was not a focus of this thesis. The GUI provided from this thesis was strictly a prototype/demo of what an example of a solution may be like. The GUI provides an easier access to the Ruby scripts and is strictly independent of all processing. The GUI allows users to select a data collection file, select what tag(s) to verify, run the test, and run the Automated Test Suite (discussed later this section). It also allows users to open a plot from previous tests. The GUI provided a more attractive appearance to the ABVT. The main drawback to using the GUI verses via command line Ruby is the inability to have total control over the verification tests. This limitation would be addressed for future work in this area.

The basic functions of the GUI were: (1) select an OneSAF xml data file, (2) parse the data file for entity tags, (3) provide a way to select an entity tag for verification, and (4) provide visual output when completed. Figure 16 is the screen capture of the GUI prototype.

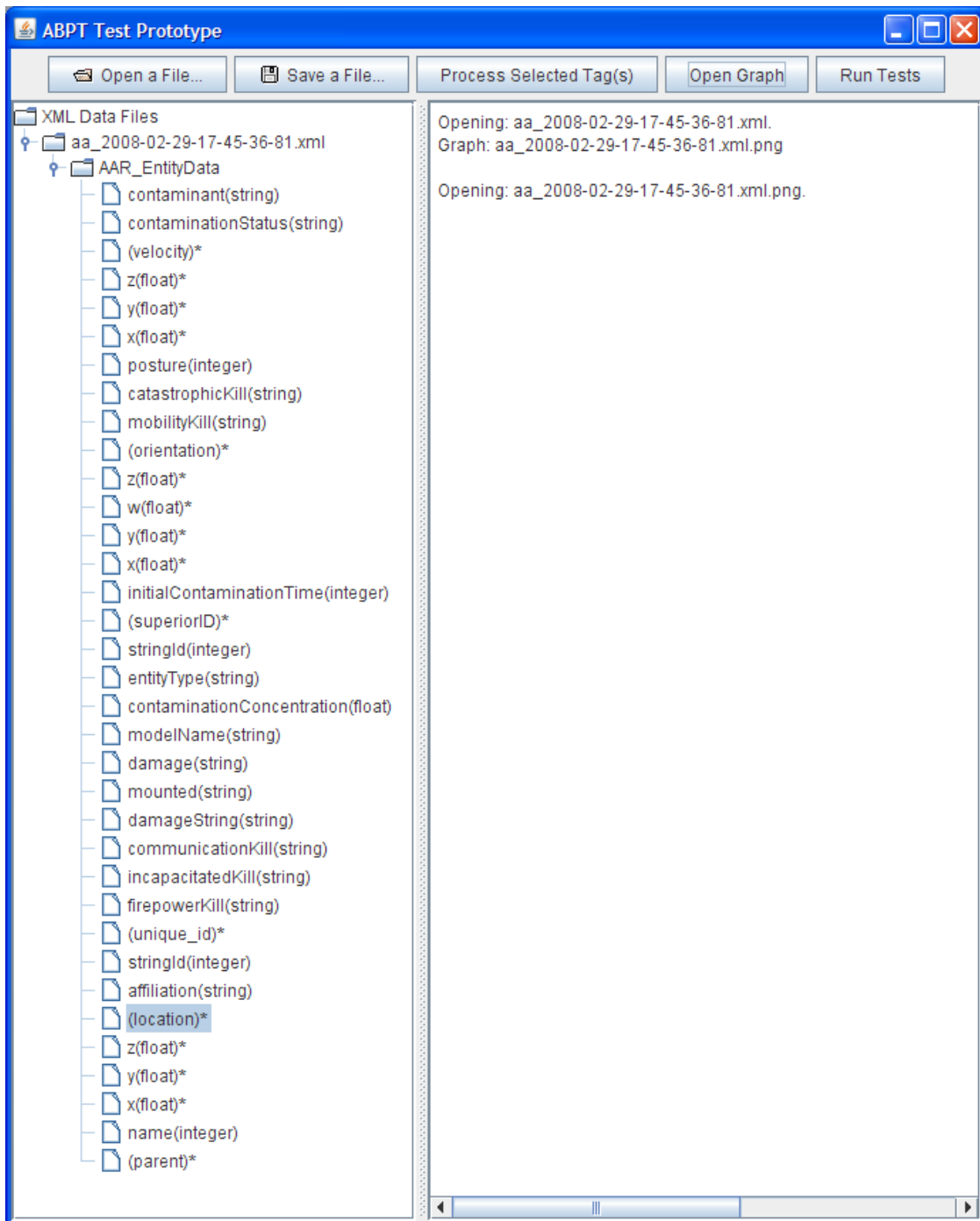


Figure 16 GUI prototype.



Once a GUI prototype was completed, TRAC-Monterey resumed further GUI development. OneSAF Developers requested for a simple one mouse click that would launch a series of automated behavior verification tests. The following eight test parameters were requested:

1. Entities move to the conversion cache.
  - a. If entity is not at conversion cache, check location relative to conversion cache every 1 minute. If entity is not closer to conversion cache for 10 consecutive minutes, then fail.
  - b. Report time when entity is within 50 meters of conversion cache as “Time 1”.
2. At conversion cache, entities assume sitting position.
  - a. After time 1, if entity is within 30 meters of conversion cache, and does not sit within 1 minute, then Fail.
  - b. Report sitting time as “Time 2”.
3. Entity stands at completion of conversion.
  - a. Report standing time as “Time 3”.
4. Entities stays at conversion cache for the time required to convert mines.
  - a. Difference between “Time 2” and “Time 3” is greater than required conversion time.
5. Entities move the mines to the dump cache.
  - a. After “Time 3”, if entity is not at conversion cache, check location relative to dump cache every 1 minute. If entity is not closer to conversion cache for 10 consecutive minutes, then Fail.
  - b. Record arrival at dump cache as “Time 4”.
6. The number of mines and conversion kits in the conversion cache decrease.
  - a. After “Time 3”, the numbers decrease.
  - b. Data not available.
7. The number of mines and conversion kits in the dump cache increase.
  - a. After “Time 4”, the numbers increase.
  - b. Data not available.
8. Entities move to random locations in the minefield.
  - a. After “Time 4”, the entities move to a location within the minefield within 1 minute, otherwise Fail.\

Tests six through eight did not pass with any of the data sets. Either the data collection or the caches failed to update cargo quantities. ECM scenario would always fail once the entities in the HMMWV stopped at the Dump Cache. The OneSAF simulation recognized the scenario failed but would not close the data files appropriately with closing tags. Figure 18 shows one of the more common error messages once the scenario fails.

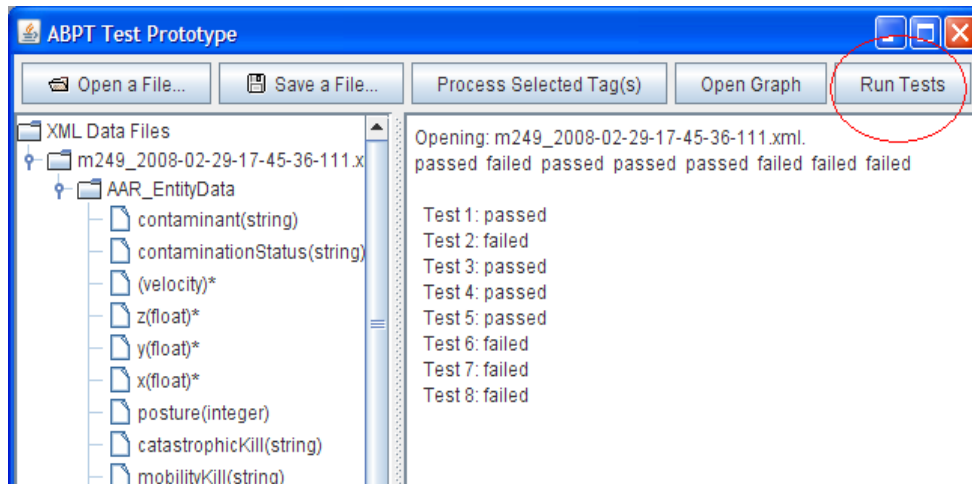


Figure 17 Automated testing suite results.

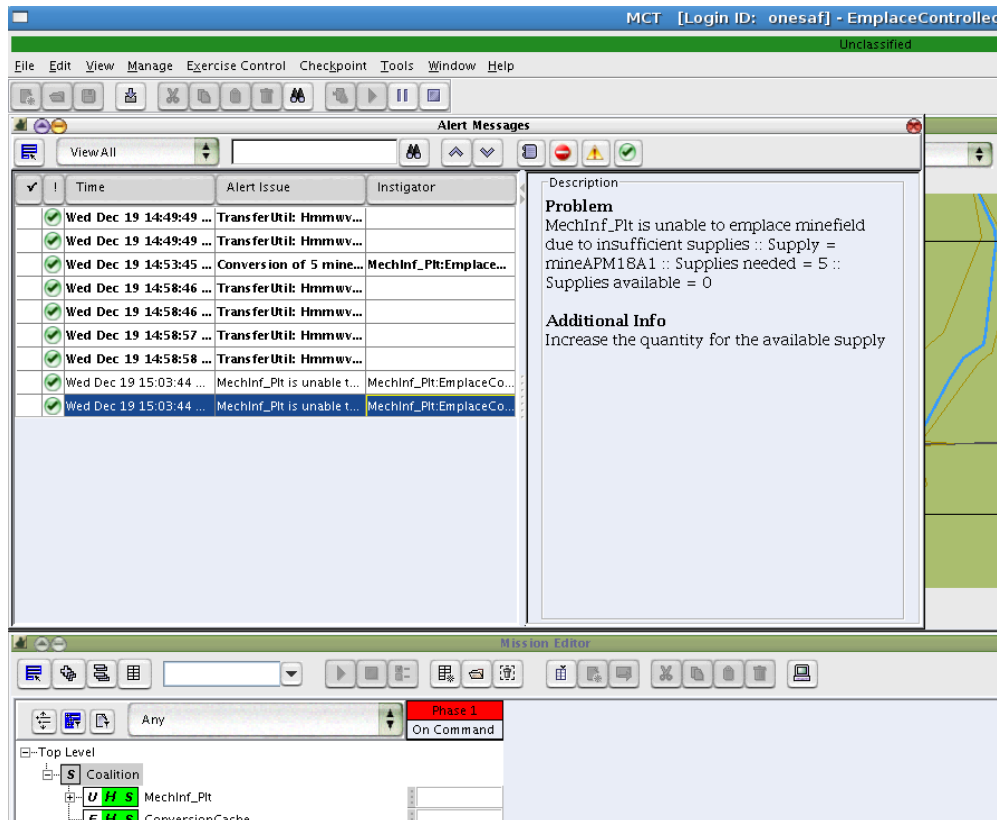


Figure 18 Common error message.<sup>23</sup>

<sup>23</sup> OneSAF screen capture.

## IV. CONCLUSION

### A. SOFTWARE TESTING

The purpose of the work for this thesis was to provide a blueprint on how to approach system behavior verification on a system that may or may not have adequate documentation. While the work concentrated on entity behavior, the approach would be parallel to any multitude of system behaviors. The key is to have an independent data collection tool.

From mid 2005 to mid 2007, TRAC-Monterey was successful in verifying three out of 51 scenarios as per OneSAF Verification & Validation (OV&V). According to TRAC-Monterey, the greatest factor in contributing to lack of success was that the required manpower was not available<sup>24</sup>. Manpower, as this thesis shows, was not the greatest issue that hindered entity behavior verification. The greatest hindrance to entity behavior verification was the lack of documentation, followed closely by lack of developer support (although with sufficient documentation, developer support may not be necessary).

TRAC-Monterey perceived manpower as the main problem because all steps involved with entity verification was done manually. These steps included scenario creation, test the scenario, modify the scenario to create variations that were of interest to the entity being verified, re-test the modified scenario, and record all behavior observed during the tests. For any one scenario, this was an extensive manpower issue. If TRAC-Monterey had one person for every scenario, all scenarios could be tested and entity behaviors could be verified in a span of three weeks time. The only thing missing, however, is the lack of data to enforce the analysis and all analysis would have come from the tester's screen<sup>25</sup>. The question then becomes, "How do you verify the analysis?" Traceability and enforcement is not available "observing screen outputs." Since OneSAF is a semi-autonomous system, entities should not necessarily behave

---

<sup>24</sup> Appendix J TRAC-Monterey Verification Process Methodology , slides # one and five.

<sup>25</sup> Appendix J TRAC-Monterey Verification Process Methodology , slide # six.

exactly the same during each iteration of testing. The conclusion is simple: the analysis by method of observation alone is not enough for entity behavior objective verification or traceability<sup>26</sup>.

## **B. POSITIVES**

The scenarios in OneSAF Objective System (OOS) are all user definable. The only limitations to scenario generation (in theory) are the limitations the real world imposes. For example, a human entity is not allowed to fly unless he is in a vehicle entity capable of flying. OneSAF developers did provide samples of some scenarios but unfortunately, not the ones OV&V listed on their requirements. Therefore, the scenarios had to be created by manually.

During the work of this thesis, data collection was implemented, although with limited capabilities. Enabling data collection freed the tester from having to analyze data and record scenario outputs during the test execution. This allowed multiple simultaneous testing of one scenario, limited only by the number of machines that were available. At the end of the tests, all the data files were collected and analyzed on different machines.

Twice during the work of this thesis, the physical machines had software configuration and/or operating system (OS) errors and required reinstallation of the OS and the OOS. After the two environment failures, virtual machines (VM) were configured and used for the remainder of the work. In the span of time it took to rebuild the second physical machine after failure, two VMs (one with Windows® and one with Linux) were created. Virtualization proved indispensable for the remaining work for this thesis. When the VMs failed, a 15 minute restore brought the entire OOS environment back online. Virtualization of the OOS environment also allowed for smoother transitions to the newer versions of the OOS.

The creation of scenario files were the only step left unchanged during the work of this thesis. Once the scenario was created, and forgone enough testing, the next step

---

<sup>26</sup>Appendix K OneSAF Users Conference Orlando Florida Presentation, slide # four.

was to modify the scenarios to emphasize entity behaviors. This was done by opening the scenario, modify the scenario, and then execute the scenario enough times to have adequate data. This was a long and tedious process and often the modified scenario would not run<sup>27</sup>. A more streamlined and automated way to modify the scenario file was done by Christopher Eatinger in his thesis titled "*TESTING AUTOMATION TOOLS FOR SECURE SOFTWARE DEVELOPMENT*".<sup>28</sup> The modified scenario file was tested in the same manner as the initial tests. This process was repeated until all variations of interest were tested.

As mentioned earlier, discovering how to use the external data collection module was the key to the entity data verification. However, if the data files were not useable, then the efforts would have been in vain. If the first break was the discovery of the collection tool, then the second break was that the data files were XML files. They could have been any flat text files but OneSAF developers chose to use XML throughout the system. The scenario files, data collection files, and entity specification files were all XML files. The work of this thesis would not have reached the level it had if the data collection files were binary or any other formatted files.

Ruby programming language proved indispensable for parsing the data files. Ruby was easy to use and to learn. It proved extremely efficient in that it could parse, calculate, coordinate multiple input and output files, and execute external programs via system calls when needed. Ruby was used to parse the data files, create intermediate files, and then fed those intermediate files to gnuplot along with created gnuplot scripts that generated either Portable Network Graphics (PNG) or Postscript (PS) format visualizations depending on the OS environment. Once the visualizations were created, Ruby was able to make a system call to open the graph, displaying the content to the user, provided immediate feedback on entity behavior verification. Furthermore, Ruby was

---

<sup>27</sup> One of the challenges of OOS was the lack of overall system stability. Often when a modified scenario was executed, the system, or one of the components of the system would throw unrecoverable Java exceptions, which left the tester to re-initialize the OOS, or sometimes reboot the OS.

<sup>28</sup> "TESTING AUTOMATION TOOLS FOR SECURE SOFTWARE DEVELOPMENT," Christopher Eatinger, June 2008, NPS.

platform independent and was incorporated into the GUI prototype's executable jar file with the use of JRuby, Java implementation of Ruby.

This thesis work also brought to the attention of OneSAF developers couple of errors found during scenario executions. The first is the ability of a HMMWV entity being able to cross a river without using a bridge. In the real world, the HMMWV could cross a river if the water level was low and the flow of the water was fairly slow. However, this was not the case in the scenario. Another error was the posture of the human entity in vehicles. In the ECM scenario, the two human entities posture was *standing* while riding (presumably driving the HMMWV) in the HMMWV. While this is not an error to reprogram the entire system over for, it does point to the fact that behavior entity verification on large systems can be useful to detect errors.

Additionally, the work of this thesis was presented during the 2008 OOS User's Conference, where it was received well by the attendees. The OV&V and OOS developers were also impressed enough to incorporate either the ideas or parts of this thesis into future releases of OneSAF Objective System.

### **C. NEGATIVES**

Stated throughout this thesis is the fact that OOS lacked available documentation. TRAC-Monterey was not successful in attaining even a user's guide to the system. To OneSAF's credit, their secure site provided documentations about the architecture, but not of the modules themselves.

How could such a system be created without documentation? It is very possible the developers do have documentation but that OV&V were not in the know to acquire them. Perhaps issue with trade secrets or copyright issues also played a role. Whatever the reason, TRAC-Monterey did not have any documentation.

Besides lack of documentation, there were several other negative issues discovered. For example, OOS lacked an XML viewer. This was rather an odd discovery considering almost everything within OOS communicated via XML files.

OOS stored states, data, configurations, and others in XML files. Having a native XML browser that could open XML files created by the OOS would have had significant impact to this thesis.

Another negative time consuming issue was the fact that TRAC-Monterey was responsible in creating the scenario files. This meant TRAC-Monterey testers really had to learn a system, without any documentation. The learning curve to the OOS interface was incredibly steep. Learning the system may have had the greatest impact to why TRAC-Monterey was only able to verify three out of 51 scenarios in over a years' time.

TRAC-Monterey was representative of an independent verification and testing unit. The perspective of the OOS to TRAC-Monterey and the work of this thesis was that the OOS system was unstable as a whole. For example, when the system initiates, a major unrecoverable Java exception is uncaught when a network interface card (NIC) was not detected or enabled. The system would not start, nor provide any useful exception error message. The NIC issue did cost the progress by a weeks' time. OOS would through Java exceptions throughout the entire verification testing process. An interesting thing to point out is that once an exception is thrown, the rest of the OOS is not notified. Often, an unresponsive system was anticipated upon the third execution of the same scenario. The question that lingered after so many system exceptions was, "Why was Java used for this real-time simulation system?" OV&V did not have a clear answer to this question, but hinted to the fact that OOS was developed ground-up to run in multiple platforms, well two actually. The performance of the OOS (as noted in this thesis) was much better in Linux than in Windows® machines. In Windows®, the average memory used after OOS initialized was around 1.25 GB of memory and utilizing around 25-75% on a dual-core processor. In Debian® Linux, the ram usage was around 720 MB and the processor usage was around 10-50%.<sup>29</sup> OOS emphasizes platform independence and yet during an expo held at NPS in 2007 and at the 2008 User's

---

<sup>29</sup> These observations were consistent on two Dell Laptops: Dell XPS 1210 4GB Ram 2.1GHz dual-core Windows® XP, and Dell XPS 1530, 4GB Ram, 2.4 GHz dual-core running Debian® Linux. Both configured with VMware™ Workstation® and running identical VMs, one Windows® and the other Linux.

Conference, all machines were running Windows® XP. Having a system intended to run on multi-platforms is always welcome; however, the implementation and performance should match industry expectations.

The single most important negative issue with the OOS for entity verification was the data collection module. Even toward the end of the work on this thesis, the data collector was incredibly unstable and the single most cause of progress delays. Yet, without it producing the data provided, the work would have come to a halt.

Lastly, the OOS modular development was not seamless. Data inconsistencies, unit inconsistencies, label inconsistencies, file inconsistencies, and operational inconsistencies all pointed to the fact that many developers from all over contributed to the OOS. Documentation would have certainly helped. Because of the lack of documentation, when inconsistencies were found TRAC-Monterey would contact OV&V and await their response. OV&V would inquire OOS developers pertaining to the inconsistency area. More often than not, TRAC-Monterey would not receive a response. It was clear that a break in communication between the developers and OV&V, and/or with OV&V and TRAC-Monterey existed.

Despite the numerous shortfalls, the work progressed, and tools were developed. More importantly, many lessons of the trade were learned.

#### **D. FUTURE WORK**

Any system worth building should be worth studying. OOS is a great candidate for research because it offered all the elements of any large system, both pros and cons. One thing that is apparent at this stage is that there are numerous things that remain to be done. The hard challenge of learning and familiarizing with the system really took the most resources.

The project had great vision that included producing a compiler that would take requested behavior inputs and would produce a Ruby script that would in turn process a data file. Another area to improve would be to incorporate a library of prebuilt scripts that a user could select, run, modify and save as a user defined test. Thus far, this thesis



has made great progress in creating a blueprint to verify entity behavior of a CGF simulation system (with a major handicap of not being provided any documentation). The work started at an extremely slow rate but progress sped up once data files were collected.

THIS PAGE INTENTIONALLY LEFT BLANK

## LIST OF REFERENCES

- Boehm, Barry. "Verifying and Validating Software Requirements and Design Specifications," In *IEEE Software* 1 January 1984, 75-88.
- "OneSAF Verification & Validation." One Semi-Automated Forces OneSAF Developer Site. Dev.OneSAF.net. 7 Aug 2008  
<[https://dev.onesaf.net/devsite/Development/Verification\\_and\\_Validation/](https://dev.onesaf.net/devsite/Development/Verification_and_Validation/)>.
- Ruby. <<http://www.ruby-lang.org/en/>>. Last accessed July 2008.
- Jruby. <<http://jruby.codehaus.org/>>. Last accessed July 2008.
- Eatinger, Christopher, "TESTING AUTOMATION TOOLS FOR SECURE SOFTWARE DEVELOPMENT", NPS Monterey, CA, June 2008.
- Thomas, Dave, and Chad Fowler. Programming Ruby, The Pragmatic Programmers' Guide. 2nd ed. Dallas, TX: Pragmatic Bookshelf, 2005.

THIS PAGE INTENTIONALLY LEFT BLANK

## APPENDIX A MOVE TACTICALLY (MT) SCENARIO RUBY SCRIPT

```
#!/usr/bin/env ruby
class Parse
# behavior verification prototype in Ruby This is the FINAL Ver for Phase I
# written by John Leo, August 2007
# Monterey, CA
#
# modified by Mikhail Auguston, August 29, 2007
=====

### global vars ALL CONST VARS CAPITALIZED LETTERS
=====

### PULLS SYSTEM DATE/TIME INTO RUBY SCRIPT
require 'time'
@@time=0

### LABELS
@@searchLabel = "null"
@@parseLabel = "null"

### SPEED
@@speedCommanded = 0
@@DEV = 1.15 #modify this to change what slow speed deviation to catch
@@DEVGR = 1.05
@@numOfSlow = 0;
@@numOfFast = 0;
@@numOfSpeedEntry=0
@@entitySpeed=0
@@SPEED_LOW_DEV = 0
@@SPEED_HIGH_DEV = 0
@@SPEED_KM_HR = 0
### Used later, not implemented
@@numOfSpeedZero=0

### DISTANCE
@@distanceNumOf=0
@@distanceNumOfDeviation=0
@@distanceMaxDeviation=0
@@totalNumOfMovement=0;
### Not used this iteration
Float @@distanceSegment=0
=====

Float @@distanceDeviation=0
Float @@distanceCurrent=0
Float @@DISTANCE_MAX=0;
Float @@distancePrevious =0;

### ENTITY
@@numOfEntityLocations=0;
@@entityType="null"
@@taskName="null"
### STRUCT to contain a GCC Coord unit and later more information.
Coordinate = Struct.new( :x, :y, :z, :s)
@@CoordEnti = Coordinate.new( "0", "0", "0", "0")
@@CoordDest = Coordinate.new( "0", "0", "0", "0")
@@CoordInit = Coordinate.new( "0", "0", "0", "0")

### GLOBAL FILE HANDLERS
@@scenarioFile = "null"
@@dataFile = "null"
@@fprintfSpeedReport=0
@@fprintfDeviationFromPathReport=0
```



```

sourceZ = "z"
sourceY = "y"
sourceX = "x"

getCoords = false
### INIT LABELS
if @@searchLabel == "null"
    @@searchLabel = "speed"
end
if @@scenarioFile == "null"
    @@scenarioFile = "Scenario.xml"
end
### LOOK FOR INIT AND TARGET COORDS
File.open(@@scenarioFile,"r") do |file|
    file.each do |line|
        ### Get the initial coordinates
        if (line =~ /(#{sourceLabel})>/ && getCoords == false)
            getCoords = true
        end
        if ( (line =~ /(<z>)(-?\d+\.\d*)/) && getCoords == true )
            @@CoordInit[:z] = $2
        end
        if ( (line =~ /(<y>)(-?\d+\.\d*)/) && getCoords == true )
            @@CoordInit[:y] = $2
        end
        if ( (line =~ /(<x>)(-?\d+\.\d*)/) && getCoords == true )
            @@CoordInit[:x] = $2
            getCoords = "getDest"
        end
        if (line =~ /(#{destinationPoint})>/ && getCoords == "getDest")
            getCoords = "dest"
        end
        if ( (line =~ /(<z>)(-?\d+\.\d*)/) && getCoords == "dest" )
            @@CoordDest[:z] = $2
        end
        if ( (line =~ /(<y>)(-?\d+\.\d*)/) && getCoords == "dest" )
            @@CoordDest[:y] = $2
        end
        if ( (line =~ /(<x>)(-?\d+\.\d*)/) && getCoords == "dest" )
            @@CoordDest[:x] = $2
            getCoords = "nomore" #do NOT reset to 'false'
        end
        ### Get Commanded Speed
        if (line =~ /(#{@@searchLabel})>(\d+\.\d*)/)
            @@speedCommanded = line.slice(/\d+\.\d*/)
        end
        if (line =~ /<entityType>([a-zA-Z0-9]+\.\d+\.\d*)/)
            if ( !$1.eql?("unknown") && (@@entityType.eql?("null")))
                @@entityType = $1
            end
        end
        if (line =~ /<taskName>(\d+\.\d*)</taskName>/)
            if ( !$1.eql?("unknown") && (@@taskName.eql?("null")))
                @@taskName = $1
            end
        end
    end
end
end
### Scenario is read, now init the output files
openOutputFiles()

#calc() called to get line distance. Entity Struct not yet created.
calc()

### initial setting to MAX distance
@@distancePrevious = @@DISTANCE_MAX
@@distanceCurrent = @@DISTANCE_MAX

@@SPEED_LOW_DEV = (@@DEV.to_f - 1) * 100

```





```

        @@distanceNumOfDeviation=@@distanceNumOfDeviation+1
        end
    end
    if((@@distancePrevious.to_f < @@distanceCurrent.to_f ) &&
(@@entitySpeed.to_f != 0.0))
        @devFlag = 4
        @@distanceNumOf=@@distanceNumOf+1
    end
    end
    if (xline =~ /<TIME>(\d+)/ )
        @@time = $1
        if(@devFlag > 0)
            $statement= "TIME:    #{ $1}:      Entity    Loc.\tx:
#{ @@CoordEnti.x}\n\t\t\tty: #{ @@CoordEnti.y}\n\t\t\t\ttz: #{ @@CoordEnti.z}"
            if (@devFlag < 3)
                $statement1
                =
                "#{ $1}\t#{ @@entitySpeed}\t#{ @@CoordEnti.s}"
                if (@@entitySpeed.to_f > 0.0)

                @@fprintfSpeedReport.puts("#{ $statement1}")
                end
                end
                if (@devFlag == 3 )

                @@fprintfDeviationFromPathReport.puts("#{ $1}\t\t#{ @@distanceDeviation}")
                end
                if (@devFlag == 4 )

                @@fprintfDistanceToTargetReport.puts("#{ $1}\t#{ @@distancePrevious}\t#{ @@distanceCu
rrent}")
                end
                @devFlag =0
            end
        end
        if ((@time_temp != @@time )&&( @linecounter > 100)&& (
@@numOfSpeedZero < 25 )) #SKIP THE XML HEADER INFO and limit ZERO speed entries less than
25.

        @@printfSpeedRaw.puts("#{ @@time}\t#{ @@entitySpeed}\t#{ @@CoordEnti[:s]}")

        @@printfDistanceToTargetRaw.puts("#{ @@time}\t#{ @@distancePrevious}\t#{ @@distanceCu
rrent}")

        @@fprintfDeviationFromPathReportRaw.puts("#{ @@time}\t\t#{ @@distanceDeviation}")
        @speed_temp = @@entitySpeed
        @time_temp = @@time
        @@distancePrevious = @@distanceCurrent
        @time_temp = @@time
    end
end
end

```

---



---

```

###      REPORT SUMMARY GENERATED HERE

```

---



---

```

    @@fprintfSummaryReport.puts("\n\n\n\t\t\tTEST SUMMARY\n\n\n")
    @@fprintfSummaryReport.puts("\n\n\n\t\t\tDeviation of Entity's speed")
    @@fprintfSummaryReport.puts("\n\n\n\t\t\tOutput generated when Entity's speed deviates more
than #{ @@SPEED_LOW_DEV}% down\n or more than #{ @@SPEED_HIGH_DEV}% up from commanded speed
#{ @@speedCommanded}m/s.")
    @@fprintfSummaryReport.puts("\n\n\n\t\t\tTotal          speed          measurements:
#{ @@totalNumOfMovement}")
    @@fprintfSummaryReport.puts("\n\n\n\t\t\tNumber of entries where Entity's speed was more
than #{ @@SPEED_LOW_DEV}% down than commanded speed: #{ @@numOfSlow}")
    @@fprintfSummaryReport.puts("\n\n\n\t\t\tNumber of entries where Entity's speed was more
than #{ @@SPEED_HIGH_DEV}% up than commanded speed: #{ @@numOfFast}")
    @@fprintfSummaryReport.puts("\n\n\n\t\t\tThe data set for SPEED measurments that deviate
from the commanded speed is located in file:\n\t\t\t#{ @@dataFile}_Speed_Test_Rpt.txt")

```

```

        @@fprintfSummaryReport.puts("\n\n\tThe column format for data items in that
file:")
        @@fprintfSummaryReport.puts("\tTIME\tCURRENT_SPEED\tSLOPE_OF_TERRAIN")
        @@fprintfSummaryReport.puts("\n\tThe RAW data set of Speed measurments is located
in file:\n\t #{@dataFile}_Speed_Raw_Data.txt")
        @@fprintfSummaryReport.puts("\n\tThis RAW data file is TAB delimited with NO
HEADERS. The file format is similar to the REPORT
file:\n\tTIME\tCURRENT_SPEED\tSLOPE_OF_TERRAIN")
        if(@@totalNumOfMovement.to_f == 0.0)
            @@fprintfSpeedReport.puts("\n\t ZERO DEVIATION ENTRIES FOUND.")
        end

        ###DISTANCE TO TARGET
        @@fprintfSummaryReport.puts("\n\n2. Deviation of Entity's Distance To Target:\n")
        @@fprintfSummaryReport.puts(" *Note: Distance calculations use (x, y) coordinates
ONLY. \n")
        @@fprintfSummaryReport.puts("\n\tTotal Entity Coordinate entries:
#{@numOfEntityLocations}")
        @@fprintfSummaryReport.puts("\tTotal number of Entity entries when distance DOES
NOT decrease: #{@distanceNumOf}")
        @@fprintfSummaryReport.puts("\n\tANY Distance to Target measurements that are not
strongly less than previous distance are located in file:\n \t
#{@dataFile}_Distance_To_Target_Rpt.txt")
        @@fprintfSummaryReport.puts("\n\tThe column format for data items in that file:")
        @@fprintfSummaryReport.puts("\tTIME\tENTITY'S_PREVIOUS_DISTANCE\tENTITY'S_CURRENT_
DISTANCE")
        @@fprintfSummaryReport.puts("\n\tThe RAW data set of Distance To Target
measurments is located in file:\n\t #{@dataFile}_Distance_To_Target_Raw_Data.txt")
        @@fprintfSummaryReport.puts("\n\tThis RAW data file is TAB delimited with NO
HEADERS. The file format is similar to the REPORT
file:\n\tTIME\tENTITY'S_PREVIOUS_DISTANCE\tENTITY'S_CURRENT_DISTANCE")
        if(@@distanceNumOf.to_f == 0.0)
            @@fprintfDistanceToTargetReport.puts("\n\t ZERO DEVIATION ENTRIES FOUND.")
        end

        ###ROUTE DEVIATION FROM THE ROUTE/ROAD
        @@fprintfSummaryReport.puts("\n\n3. Deviation of Entity's Location from the route
to the target location.\n")
        @@fprintfSummaryReport.puts("\nOutput generated when Entity deviates more than 10m
from the commanded route.\n")
        @@fprintfSummaryReport.puts(" *Note: Distance calculations use (x, y) coordinates
ONLY. \n")
        @@fprintfSummaryReport.puts("\n\tTotal Entity Coordinate entries:
#{@numOfEntityLocations}")
        @@fprintfSummaryReport.puts("\tNumber of entries where Entity deviates more than
10m from the route: #{@distanceNumOfDeviation}")
        @@fprintfSummaryReport.puts("\tMAX deviation detected for this scenario:
#{@distanceMaxDeviation}m")
        @@fprintfSummaryReport.puts("\n\tThe data set of deviation measurments is located
in file:\n\t #{@dataFile}_Distance_Deviation_Rpt.txt")
        @@fprintfSummaryReport.puts("\n\tThe column format for data items in that file:")
        @@fprintfSummaryReport.puts("\tTIME\tENTITY'S_DEVIATION_FROM_THE_PATH")
        @@fprintfSummaryReport.puts("\n\tThe RAW data set of Deviation Distance
measurments is located in file:\n\t #{@dataFile}_Distance_Deviation_Raw_Data.txt")
        @@fprintfSummaryReport.puts("\n\tThis RAW data file is TAB delimited with NO
HEADERS. The file format is similar to the REPORT
file:\n\tTIME\tENTITY'S_DEVIATION_FROM_THE_PATH")
        if(@@distanceNumOfDeviation.to_f == 0.0)
            @@fprintfDeviationFromPathReport.puts("\n\t ZERO DEVIATION ENTRIES
FOUND.")
        end
    end
end

```

---



---

```

### INPUT FILE PARSED FOR AUTOMATED TESTING OF SEVERAL SCENARIO AND DATA FILES

```

---



---

```

def Parse.inputfile()
    if File.exists?("inputtest.txt")
        File.open("inputtest.txt","r") do |infile|
            while iLine = infile.gets

```

```

        array = iLine.split
        iLine.split(' ')
        if array.size == 4
            @@searchLabel = array[2]
            @@dataFile = array[1]
            @@scenarioFile = array[0]
            @@parseLabel = array[3]
            searchdata()

        else if array.size == 1
            @@parseLabel = array[0]
            searchdata()

        else
            puts "ERROR INPUT FILE WITH THIS SEARCH! "
        end
    end
end

end

else
    if @@dataFile == "null"      ### HARD CODED FILE FOR ABPVT DEVELOPMENT
ONLY
        @@dataFile = "entity_CollectionTestTank_2007-08-15-09-49-29-
496.xml"
    end
    if @@parseLabel == "null"    ### HARD CODED FILE FOR ABPVT DEVELOPMENT
ONLY
        @@parseLabel = "currentSpeed"
    end

    searchdata()

end

end

#def Parse.calc( Float cx, Float cy, Float ax, Float ay ,
#               Float bx, Float by, #Float &distanceSegment,Float &distanceLine )
#               Float distanceSegment,Float distanceLine )
#   Based on algorithm from:
#       http://www.codeguru.com/forum/printthread.php?t=194400
#       By Philip Nicoletti, posted 06-14-2002 05:18 PM
#   Converted C code algorithm to Ruby
#
def Parse.calc()
    Float ax = @@CoordInit.x.to_f;Float ay = @@CoordInit.y.to_f;Float az =
@@CoordInit.z.to_f;
    Float bx = @@CoordDest.x.to_f;Float by = @@CoordDest.y.to_f;Float bz =
@@CoordDest.z.to_f;
    Float cx = @@CoordEnti.x.to_f;Float cy = @@CoordEnti.y.to_f;Float cz =
@@CoordEnti.z.to_f;
    # use only x and y coordinates for distance calculations
    ### INITIAL CALL JUST NEEDS TO CALC MAX DISTANCE; OTW, DO 3 POINT CALCULATION
    if @@DISTANCE_MAX != 0.0
        Float dista = (bx-cx)*(bx-cx)+(by-cy)*(by-cy)
        @@distanceCurrent = Math.sqrt(dista)
        Float r_numerator = (cx-ax)*(bx-ax) + (cy-ay)*(by-ay)
        Float r_denominator = (bx-ax)*(bx-ax) + (by-ay)*(by-ay)
        Float r = r_numerator / r_denominator
        Float px = ax + r*(bx-ax)
        Float py = ay + r*(by-ay)
        Float s = ((ay-cy)*(bx-ax)-(ax-cx)*(by-ay)) / r_denominator
        @@distanceDeviation = s.abs * Math.sqrt(r_denominator)
        #// (xx,yy) is the point on the lineSegment closest to (cx,cy)
        Float xx = px
        Float yy = py

        if ( (r >= 0) && (r <= 1) )
            @@distanceSegment = @@distanceDeviation;

```

```

else
    Float dist1 = (cx-ax)*(cx-ax) + (cy-ay)*(cy-ay)
    Float dist2 = (cx-bx)*(cx-bx) + (cy-by)*(cy-by)

    if (dist1 < dist2)
        xx = ax
        yy = ay
        #distanceSegment = sqrt(dist1);
        @@distanceSegment = Math.sqrt(dist1)

    else
        xx = bx
        yy = by
        #distanceSegment = sqrt(dist2);
        @@distanceSegment = Math.sqrt(dist2)

    end

    end
    if( cx == 0 ) ### INITIAL VALUE
        @@distanceDeviation =0
        @@distanceSegment=0
    end
else
    Float dista = (ax-bx)*(ax-bx)+(ay-by)*(ay-by)
    @@DISTANCE_MAX = Math.sqrt(dista)

end

end
def Parse.searchdata()
    scenario()
    data()          #run it on the data collection file
    closeOutputFiles()

end
inputfile()
end

```

## APPENDIX B PRESCRIPT

```
#!/usr/bin/env ruby
#require 'rexml/document' #Can't use because more often than not, the xml data files are not closed properly!!!

#
# The purpose of this script is to parse data collection files and return the (if one exist) entity in that file. Input to
# this script are any file names... TODO exact path of file names...
# The output is " " and ":" delimited string as follows:
# <XML collection data file1>.xml <entity name> <property1> <p2> <p3> ... : <XML collection data file2>.xml
# <entity name2> <property1> <p2> <p3> ... : ...
#

class ParseControl
  ### STRUCT to contain a GCC Coord unit and later more information.
  #require 'rexml/document'

  @@xml=0
  @@entityFilename = "entity.tags.txt"
  @@entityFile=""

  @@propertyListStdOut=""
  @@propertyList=""
  @@terminalList=""

  #
  # .scenario() // Processes the Scenario file

  def ParseControl.scenario(input)
    ### LABEL VARS
    @inputFile=input
    @linecount = 0

    @propertyCount=-1
    @propertyList=""
    @terminalList=""
    @entityName=""
    #CAN'T use because xml are mal-formed
    @@xml = REXML::Document.new(File.open("#{input}"))

    #Get any and all co-ords
    File.open(@inputFile,"r") do |file|
      file.each do |line|
        if @linecount < 30
          @linecount += 1
          if ((line =~ /(<TYPE>)(null)/) && (@linecount < 30)) then
            #puts "File: #{@inputFile} contain NULL entries for entity TYPE in the header."
            #puts @linecount
            break
          end
        end
        if (line =~ /(<VALUES>)/) then
          @propertyCount=0
          #@propertyList = "#{ @linecount}"
        end
        if ((line =~ /(<)(.)(\\WrefID=)/) && @propertyCount==0) then
          @entityName = $2
          if (($2 != "uniqueid") && ($2 != "ENTRY") && ($2 != "SuppressionSpeedLimit")
            && \
              ($2 != "WeaponControlModel") && ($2 != "SpeedDataCollection") && \
              ($2 != "DriverFSM") && ($2 != "DirectiveDataCollection") ) then
```

```

        if ( $2 =~ /Sensor/)
            #throw away
        else
            @terminalList
            @propertyList = "#{ @inputFile} #{ @entityName} "
            @propertyCount = 1
        end
    end
end
# if ((line =~ /( < )([A-Za-z]+)( > )(-? \w+ \. * \w * - ? \d ?)( < ) / ) && ( @propertyCount > 0 ) )
# if ((line =~ /( < )(.+)( > )(. * )( < ) / ) && ( @propertyCount > 0 ) ) then
    local_var = $2
    local_var_field = $4
    local_var_type = ""
    @propertyCount += 1

    if ( local_var_field =~ /- ? \d + \. \d + E ? - ? \d ? / ) then

        local_var_type = "(float)"

    else

        if ( local_var_field =~ ^ \d + / ) then local_var_type = "(integer)"
        else
            if ( local_var_field =~ ^ \w + / ) then
                local_var_type = "(string)"
            end
        end
    end

    # puts "#{local_var_field} #{local_var_type}"
    ##### formatting or not?

    #####
    #
    # JAVA needs to return this list w/o the data types!
    @terminalList = "#{ @terminalList} #{local_var} #{local_var_type} \n \t"

    @propertyList = "#{ @propertyList} #{local_var} #{local_var_type} "
    #
    #####

end

if ((line =~ /( < )(\w+)( > \n ) / ) && ( @propertyCount > 0 ) ) then

    @propertyList = "#{ @propertyList} ({ $2 })* "
    @terminalList = "#{ @terminalList} ({ $2 })* \n \t"

end

if ((line =~ /( < \VALUES > ) / ) && ( @propertyCount < 6 ) ) then
    @propertyCount -= 1
    @propertyList = ""
end

if ((line =~ /( < \VALUES > ) / ) && ( @propertyCount >= 6 ) ) then

    if ( ( @propertyList =~ /slopei / ) || ( @propertyList =~ /stgCodei / ) || ( @propertyList =~
/ entityLocation / i ) \
        || ( @propertyList =~ ^ \W orientation \W / i ) )
        # puts "Entity: #{ @entityName } contains the following proprty tags:"

        ##### todo: return to calling Java GUI
        #####

        # puts "#{ @propertyList}"
        # @ @entityFile.puts "#{ @propertyList} : "

        @ @propertyList = "#{ @ @propertyList} #{ @propertyList} : "
        @ @terminalList = "#{ @ @terminalList} #{ @terminalList} : "
        @terminalList = ""
    end
end

```

```

@propertyCount=-1
end
else
break
end
#puts "#{ @propertyList} but continuing..."
@propertyList=""
@terminalList=""
@propertyCount=-1
end
end
end
if @entityName == "" then puts "FILE #@inputFile contains no ENTITY information." end
end

#
#

def ParseControl.process()
ARGV.each { | x |
if File.exists?("#{x}")
#puts x
@@newFile =true
scenario(x)
#puts
else
puts
puts "File: #{x} not found!"
puts
end
}
end

#%x{clear}
#puts
#puts
#puts Time.now
#puts "#{ARGV.size} FILENAMES entered..."

# UNCOMMENT TO OPERATED
if ARGV.size > 0 then
@@@entityFile= File.new("#{ @ @entityFilename}", "w")
process()
@@@entityFile.close
end

#puts
#puts "property tags:"
#puts "#{ @ @propertyList}"
puts "#{ @ @terminalList}"
#puts Time.now

end

```

THIS PAGE INTENTIONALLY LEFT BLANK



## APPENDIX C POSTSCRIPT

```
#!/usr/bin/env ruby
class ParseControl
  ### STRUCT to contain a GCC Coord unit and later more information.
  Coordinate = Struct.new( :x, :y, :z, :s)
  @@CoordInit = Coordinate.new( "0", "0", "0", "0")
  @@CoordInit2 = Coordinate.new( "0", "0", "0", "0")

  @@savePltFilename="save.plt"
  @@savePltFile=0

  @@linux=false
  @@inputFile=""
  @@xyOutputFilename=0
  @@xyOutputFile=0
  @@plotScriptName=0
  @@plotScriptFile=0
  @@speedLabel=0
  #@ @newFile=true

  @@entityName=""
  @@ ARGV=""
  @@argv_num=0
  @@lineOutputToFile=""

  @@CoordFile=0
  @@CoordFilename=""

  #
  #

  def ParseControl.getOS()
    #puts ARGV
    #os = %x{uname}
    #if /^cyg/i =~ os
    os = "cygwin...."
    if os =~ /^cyg/i
      #puts "CYGWIN Environment Detected..."
    else
      #puts "Linux Environment Detected..."
      @ @linux=true
    end
  end

end

#
#

  def ParseControl.plotScript()
    @ @plotScriptName="#{ @ @xyOutputFilename}.plt"
    @ @plotScriptFile = File.new("#{ @ @plotScriptName}", "w")

    # @ @savePltFilename is "save.plt" which is a generic plot formatter for plot-to-file
    @ @savePltFile = File.new("#{ @ @savePltFilename}", "w")

    if @ @linux
      @ @savePltFile.puts "set terminal postscript landscape enhanced color dashed " + \
        "lw 1 \"Helvetica\" 14"
      @ @savePltFile.puts "set size 1.0, 0.8"
      @ @savePltFile.puts "set output \"#{ @ @xyOutputFilename}.ps\""
      @ @savePltFile.puts "replot"
      # @ @savePltFile.puts "set terminal postscript"
    else

```

```

    @@savePltFile.puts "set terminal png"
    @@savePltFile.puts "set size 1.0, 0.8"
    @@savePltFile.puts "set output \"#{ @@xyOutputFilename }.png\""
    @@savePltFile.puts "replot"
    #@ @savePltFile.puts "set terminal windows"
end

@@savePltFile.close
##### Script file controls WHAT is plotted by simple one liner "plot xFile" #####
usingWhat="1:2"
#Time VS One
if ( @@argv_num < 3 ) then
    usingWhat = "1:2"
end
#X VS Y default
if ( @@ARGV =~ /entityLocation/i && @@argv_num < 3 ) then
    usingWhat = "2:3"
end

if ( @@ARGV =~ /time/i && @@ARGV =~ /xcord/i ) then
    usingWhat = "1:2"
end

if ( @@ARGV =~ /time/i && @@ARGV =~ /ycord/i ) then
    usingWhat = "1:3"
end

if ( @@ARGV =~ /time/i && @@ARGV =~ /slope/i ) then
    usingWhat = "1:3"
end

if ( @@ARGV =~ /currentspeed/i && @@ARGV =~ /slope/i ) then
    usingWhat = "2:3"
end

##### SOMETHING TO CUSTOMIZE PLOTS... #####
#@ @plotScriptFile.puts("plot \"#{ @@xyOutputFilename }\" using #{usingWhat}")
#@ @plotScriptFile.puts("plot \"#{ @@xyOutputFilename }\" using 2:3")
#@ @plotScriptFile.puts("load \"#{ @@savePltFilename }")

#if @linux then
#    #@ @plotScriptFile.puts("!mv my-plot.ps #{ @@xyOutputFilename }.ps")
#else
#    #@ @plotScriptFile.puts("!mv my-plot.png #{ @@xyOutputFilename }.png")
#end

#@ @plotScriptFile.close
end

```

---



---

```

#
# experimental multiplots

```

---



---

```

def ParseControl.plotScripts(x,y,z)
    @@plotScriptName="multiPlot.plt"
    @@plotScriptFile = File.new("#{ @@plotScriptName}", "w")
    @@plotScriptFile.puts("plot \"#{x}\" using 2:3,\"#{y}\" using 2:3,\"#{z}\" using 2:3")
    @@plotScriptFile.puts("load \"#{ @@savePltFilename }")
    @@plotScriptFile.puts("!mv my-plot.ps #{ @@plotScriptName }.ps")
    @@plotScriptFile.close
    plot()
end

```

---



---

```

#
#

```

---



---

```

def ParseControl.plot()
    #exec "gnuplot #{ @@filenameRawDistanceDataScript} #{ @@filenameRawSpeedDataScript}"

```

```

        #exec "gnuplot #{ @ @plotScriptName}"
if @ @linux
    %x{gnuplot #{ @ @plotScriptName}}
        %%x{rm -f my-plot.ps}
else
    %x{plot #{ @ @plotScriptName}}
        %%x{del my-plot.png}
end
end
def ParseControl.openOutputScripts()
end

```

---

```

#
# .scenario() // Processes the data XML file

```

---

```

def ParseControl.dataFile()
    ### LABEL VARS
    @ @entityName = ARGV[1]
    @ @speed = 0
    @ @time = 0
    @entityLocked = -1
    @var = ""

    @subLockVar = ""
    @subLock = false

    #Get any and all co-ords
    File.open(@ @inputFile, "r") do |file|
        file.each do |line|

            # New set of entity tags found, set flag
            if (line =~ /( <#{ @ @entityName} )/) then
                @entityLocked = 0
            end
            # once flagged, then process each line until flag is off
            if ( ( line =~ /( \w+ )/ ) && ( @entityLocked == 0 ) ) then
                @var = $1
                # If the line matches one of the search tags entered
                if ( @ @ARGV =~ /#{ @var} / ) then

                    # IF the tag is alone <xxx>, then a set of other tags are next
                    if ( line =~ /( <#{ @var} > \n ) ) # Only <xxxx> found within an entity tag set
                        @subLockVar = @var
                        @subLock = true
                        #puts @subLockVar #this is correct
                    else
                        # Count tag as an entity property tag
                        @ @argv_num += 1
                    end

                    # If line contains a form of grid coordinate, then grab it
                    if ( ( line =~ /( <#{ @var} > GCC: \W \W ( - ? \d + . ? \d * ) ( \W \W ) ( - ? \d + . \d * ) ( \W \W ) ( - ? \d + . \d * ) ) \
                        && ( @entityLocked == 0 ) ) ) then

                        @ @CoordInit[:x] = $2
                        @ @CoordInit[:y] = $4
                        @ @CoordInit[:z] = $6 # :z not used...
                        @ @lineOutputToFile = "#{ @ @lineOutputToFile }#{ @ @CoordInit[:x] } \t" +

                    "#{ @ @CoordInit[:y] } \t"
                        # @ @argv_num = @ @argv_num + 2
                    end

                    # Some string to int conversion, like posture
                    if ( ( line =~ /( <#{ @var} > ) ( - ? \d * . ? \d * E ? - ? \d ? \w * ) ( < / # { @var} > ) ) && \
                        ( @entityLocked == 0 ) ) then
                        var_num = $2
                        if ( var_num =~ /false/i || var_num =~ /sitting/i ) then

```

```

        var_num = 0
    end
    if ( var_num =~ /true/i || var_num =~ /standing/i ) then
        var_num = 1
    end
    @@lineOutputToFile = "#{ @@lineOutputToFile}#{ var_num}\t"
end
end
end
if ( @subLock && line =~ /<([xyz])>(-?\d+\.\d+)<\/[xyz]>/ ) then
    #if ( line =~ /<([x-z])>(>)(-?\d+\.\d+)<\/[x-z]>\n/ ) then
        @CoordInit2["#{ $1 }"] = $2
        var = $1
        #puts line
    if var == "x" then
        @subLock=false
        @@lineOutputToFile = "#{ @@lineOutputToFile}#{ @@CoordInit2[:x]}\t" +
            "#{ @@CoordInit2[:y]}\t"
        #puts "#{ @@CoordInit2[:x]}\t#{ @@CoordInit2[:y]}"
    end

    end
    if ( line =~ /<\/#{ @@entityName}>\/ ) && ( @entityLocked == 0 ) then
        @entityLocked = 1
    end
    if ( line =~ /<TIME>(\d+)<\/TIME>\/ ) && ( @entityLocked == 1 ) && \
        ( @entityLocked == 1 ) then
        @@lineOutputToFile = "#{ $1 }\t#{ @@lineOutputToFile}"
        @@xyOutputFile.puts "#{ @@lineOutputToFile}"
        #puts @@lineOutputToFile

        @@lineOutputToFile=""
        @entityLocked = -1
        @@argv_num=0

    end
end
end
end

@@xyOutputFile.close
end

```

---

```

#
#

```

---

```

def ParseControl.process()
    #ARGV.each { | x |
        if File.exists?("#{ ARGV[0] }")
            #puts ARGV[0]
            #@newFile =true
        @@inputFile = ARGV[0]
        @@xyOutputFilename = "#{ @@inputFile }.#{ ARGV[2] }.txt"
        @@xyOutputFile = File.new("#{ @@xyOutputFilename }", "w")
        #scenario(@@inputFile)
        dataFile()
        plotScript()
        #puts "plotting... #{ ARGV[0] }"
        plot()
        #puts

    else

        #puts
        puts "File: #{ ARGV[0] } not found!"
        #puts

    end

end
end

```

---

```

#
#   START HERE
#
#####

getOS()
#%x{clear}
#puts
#puts
#puts Time.now
@@argv_num = ARGV.size - 1
#puts "#{ARGV.size} {PARAMETERS} entered..."

# build the argument string to use as a RE parser
for i in 2..@@argv_num
  #puts ARGV[i]
  @@ARGV = "#{@@ARGV}#{ARGV[i]} "
end
#puts @@ARGV
# UNCOMMENT TO OPERATED
if ARGV.size > 0 then process() end

#clean up temp files
if @linux
  %x{rm -f #{@@savePltFilename}}
  %x{rm -f #{@@plotScriptName}}
else
  %x{del #{@@savePltFilename}}
  %x{del #{@@plotScriptName}}
end
end

```

THIS PAGE INTENTIONALLY LEFT BLANK

## APPENDIX D MOVE TACTICALLY (MT) SAMPLE RAW DATA FILES

entity\_CollectionTestTank\_2007-08-15-09-49-29-496.xml\_Speed\_Raw\_Data.txt

21	0	0	
309	0	0	
408	0	0	
608	0.42843525442672326	-0.007060990631857278	
815	0.42843525442672326	-0.007060990631857278	
916	0.42843525442672326	-0.007060990631857278	
1008	0.8038322418472359	-0.007060990605229023	
1208	1.2672813683295319	-0.007060990661850841	
1408	1.730085814759963	-0.007060990617765217	
1608	2.1954382146037354	-0.007060990938389411	
1808	2.660595066134107	-0.007060990934562694	
2012	3.121721578033425	-0.0070609913201220564	
2208	3.5500522114181114	-0.0070609913505428334	
2408	3.9728599452675497	-0.007060991574751263	
2608	4.381451261475838	-0.007060991675884587	
2808	4.775808190967702	-0.007060991955380569	
3008	5.156388180118532	-0.0070609922137285785	
3124	5.156388180118532	-0.0070609922137285785	
3208	5.523071010305859	-0.007060992614881911	
3408	5.875534826692347	-0.00706099275698957	
3608	6.214330955285862	-0.007060993177459451	
3808	6.539714983720519	-0.007060993532419957	
3964	6.539714983720519	-0.007060993532419957	
4008	6.852064820787036	-0.0070609938504293535	
4208	7.151902588058618	-0.0070609944193040786	
4408	7.442100818498093	-0.007060994819886979	
4608	7.725204561797142	-0.007060995343038279	
4808	8.001387257148904	-0.007060995605431053	
5008	8.270825062925232	-0.007060996107677742	
5208	8.535027561832864	-0.0070609966120411816	
5292	8.535027561832864	-0.0070609966120411816	
5408	8.793408840618643	-0.007060997177282813	
5608	9.045145338523852	-0.0070609976404916175	
5808	9.239684882820328	-0.00706099821151307	
6008	9.392993631345002	-0.007060998797643769	
6208	9.52609585465716	-0.007060999202391782	
6408	9.50098594017439	-0.007060999867755768	
6452	9.50098594017439	-0.007060999867755768	
6607	9.444262747065336	-0.0070610004059010745	
6807	9.387512289431182	-0.007061000822560226	
7007	9.336798002266386	-0.007061001469055528	
7207	9.292479261160183	-0.007061001944445033	
7407	9.253944107074545	-0.007061002472492417	
7607	9.22047644532638	-0.007061003106418662	
7808	9.19127229972116	-0.007061003645920216	
8007	9.166188237714188	-0.007061004128632975	
8133	9.166188237714188	-0.007061004128632975	
8187	9.166188237714188	-0.007061004128632975	
8207	9.144284182610876	-0.007061004508102542	
8407	9.125267125242841	-0.007061005081376637	
8607	9.108756565948076	-0.007061005678802745	
8807	9.09442214417055	-0.007061006292748306	
9007	9.081977039442355	-0.007061006783585233	
9207	9.071172234568875	-0.00706100728749659	

entity\_CollectionTestTank\_2007-08-15-09-49-29-496.xml\_Distance\_To\_Target\_Raw\_Data.txt

21	2871.41395446089	2871.41395446089
309	2871.41395446089	2871.41395446089
408	2871.41395446089	2871.41395446089

608	2871.41395446089	2871.29866380205
815	2871.29866380205	2871.29866380205
916	2871.29866380205	2871.29866380205
1008	2871.29866380205	2871.14355246423
1208	2871.14355246423	2870.8901424028
1408	2870.8901424028	2870.54418833091
1608	2870.54418833091	2870.1051807533
1808	2870.1051807533	2869.57315877546
2012	2869.57315877546	2868.93644373432
2208	2868.93644373432	2868.24076043617
2408	2868.24076043617	2867.44633338114
2608	2867.44633338114	2866.57020298652
2808	2866.57020298652	2865.61521561551
3008	2865.61521561551	2864.58412615892
3124	2864.58412615892	2864.58412615892
3208	2864.58412615892	2863.47971354705
3408	2863.47971354705	2862.30482106947
3608	2862.30482106947	2861.06218177103
3808	2861.06218177103	2859.75447758796
3964	2859.75447758796	2859.75447758796
4008	2859.75447758796	2858.38431488897
4208	2858.38431488897	2856.95419563712
4408	2856.95419563712	2855.46604739414
4608	2855.46604739414	2853.92128880452
4808	2853.92128880452	2852.32130383159
5008	2852.32130383159	2850.66744121307
5208	2850.66744121307	2848.96074782584
5292	2848.96074782584	2848.96074782584
5408	2848.96074782584	2847.20238770779
5608	2847.20238770779	2845.39368957886
5808	2845.39368957886	2843.54609074328
6008	2843.54609074328	2841.66783585627
6208	2841.66783585627	2839.76296548742
6408	2839.76296548742	2837.86311627299
6452	2837.86311627299	2837.86311627299
6607	2837.86311627299	2835.9840522615
6807	2835.9840522615	2834.10689379968
7007	2834.10689379968	2832.23987642541
7207	2832.23987642541	2830.3817212627
7407	2830.3817212627	2828.53127180511
7607	2828.53127180511	2826.6875147393
7808	2826.6875147393	2824.84040791319
8007	2824.84040791319	2823.01667119431
8133	2823.01667119431	2823.01667119431
8187	2823.01667119431	2823.01667119431
8207	2823.01667119431	2821.18815004337
8407	2821.18815004337	2819.36343169125
8607	2819.36343169125	2817.54201493001
8807	2817.54201493001	2815.72346461179
9007	2815.72346461179	2813.90740294245
9207	2813.90740294245	2812.09350192202
9407	2812.09350192202	2810.28147678207
9607	2810.28147678207	2808.47108028727
9655	2808.47108028727	2808.47108028727
9807	2808.47108028727	2806.66209778891
10007	2806.66209778891	2804.85434292932
10207	2804.85434292932	2803.04765391359
10407	2803.04765391359	2801.24189027165
10603	2801.24189027165	2799.4730135787
10807	2799.4730135787	2797.63266574195
11007	2797.63266574195	2795.82900870231
11207	2795.82900870231	2794.02587745867
11403	2794.02587745867	2792.25924734164
11607	2792.25924734164	2790.42092347022
11807	2790.42092347022	2788.61898922216
12007	2788.61898922216	2786.81735374696
12203	2786.81735374696	2785.05200018731

entity\_CollectionTestTank\_2007-08-15-09-49-29-496.xml\_Distance\_Deviation\_Raw\_Data.txt



21	0
309	0
408	0
608	0.00095756711790557
815	0.00095756711790557
916	0.00095756711790557
1008	0.0022458702062105
1208	0.00435060676915896
1408	0.00722397453180222
1608	0.0108701985460707
1808	0.0152889439957207
2012	0.020577195382559
2208	0.0263551689474201
2408	0.0329532038086292
2608	0.0402297510373278
2808	0.0481611468992202
3008	0.0567244878401434
3124	0.0567244878401434
3208	0.0658966659602398
3408	0.0756540392819832
3608	0.085973879607918
3808	0.0968338819072812
3964	0.0968338819072812
4008	0.10821236788214
4208	0.120088525447509
4408	0.132446313721869
4608	0.145273917632107
4808	0.158559808038961
5008	0.172292749159254
5208	0.186464011505775
5292	0.186464011505775
5408	0.201063892574449
5608	0.216081322433598
5808	0.231421303983066
6008	0.24701535655974
6208	0.262829910379761
6408	0.278602300078891
6452	0.278602300078891
6607	0.294201661996188
6807	0.309784739560092
7007	0.325283169202989
7207	0.340707574283677
7407	0.356067561006759
7607	0.371371546992257
7808	0.386702887811379
8007	0.401839807838191
8133	0.401839807838191
8187	0.401839807838191
8207	0.41701599823074
8407	0.432160187132825
8607	0.447276535880607
8807	0.462368658184164
9007	0.47743969064092
9207	0.492492358542043
9407	0.507529025371762
9607	0.522551744093102
9655	0.522551744093102
9807	0.537562296635114
10007	0.552562230881782
10207	0.567552890366388
10407	0.582535440565685
10603	0.597211516245326
10807	0.612480146557917
11007	0.627443929107376
11207	0.642402919994531
11403	0.657058672352201

THIS PAGE INTENTIONALLY LEFT BLANK

## APPENDIX E MOVE TACTICALLY (MT) SAMPLE REPORTS

Thu, 03 Jul 2008 11:48:05 -0700

### BEHAVIOR VERIFICATION SUMMARY FOR: tankAbramsM1A1 for simple Move Tactically behavior

Behavior inputs from scenario file:

/PAIR/compositions/behavior/composite/mr/moveTactically\_CB.xml

Properties checked in data collection file:

entity\_CollectionTestTank\_2007-08-15-09-49-29-496.xml

The COMMANDED SPEED from scenario, 9.0m/s (32.4 Km/hr).

Initial Coordinates,

x: -287361.97703322954  
y: -5464905.16566183  
z: 3265213.5159455426

Target Coordinates,

x: -284493.6358636792  
y: -5465037.970153228  
z: 3265149.889087235

Total distance 2871.41395446089m from Initial to Target.

File: entity\_CollectionTestTank\_2007-08-15-09-49-29-496.xml\_Distance\_To\_Target\_Rpt.txt

TIME	ENTITY'S_PREVIOUS_DISTANCE	ENTITY'S_CURRENT_DISTANCE
ZERO DEVIATION ENTRIES FOUND.		

Thu, 03 Jul 2008 11:48:05 -0700

### BEHAVIOR VERIFICATION SUMMARY FOR: tankAbramsM1A1 for simple Move Tactically behavior

Behavior inputs from scenario file:

/PAIR/compositions/behavior/composite/mr/moveTactically\_CB.xml

Properties checked in data collection file:

entity\_CollectionTestTank\_2007-08-15-09-49-29-496.xml

The COMMANDED SPEED from scenario, 9.0m/s (32.4 Km/hr).

Initial Coordinates,

x: -287361.97703322954  
y: -5464905.16566183  
z: 3265213.5159455426

Target Coordinates,

x: -284493.6358636792  
y: -5465037.970153228  
z: 3265149.889087235

Total distance 2871.41395446089m from Initial to Target.

File: entity\_CollectionTestTank\_2007-08-15-09-49-29-496.xml\_Speed\_Test\_Rpt.txt

TIME	CURRENT_SPEED	SLOPE_OF_TERRAIN
608	0.42843525442672326	-0.007060990631857278
815	0.42843525442672326	-0.007060990631857278
1008	0.8038322418472359	-0.007060990605229023
1208	1.2672813683295319	-0.007060990661850841
1408	1.730085814759963	-0.007060990617765217
1608	2.1954382146037354	-0.007060990938389411
1808	2.660595066134107	-0.007060990934562694
2012	3.121721578033425	-0.0070609913201220564
2208	3.5500522114181114	-0.0070609913505428334
2408	3.9728599452675497	-0.007060991574751263
2608	4.381451261475838	-0.007060991675884587
2808	4.775808190967702	-0.007060991955380569
3008	5.156388180118532	-0.0070609922137285785
3208	5.523071010305859	-0.007060992614881911
3408	5.875534826692347	-0.00706099275698957
3608	6.214330955285862	-0.007060993177459451
3808	6.539714983720519	-0.007060993532419957
4008	6.852064820787036	-0.0070609938504293535

4208	7.151902588058618	-0.0070609944193040786
4408	7.442100818498093	-0.007060994819886979
4608	7.725204561797142	-0.007060995343038279
6208	9.52609585465716	-0.007060999202391782
6408	9.50098594017439	-0.007060999867755768
110605	7.819840467467071	0.008851702047446564
110805	7.78076647184685	0.008851699755941578
111005	7.766065484096864	0.008851697590139596
111205	7.751364496688694	0.008851695052097819
111405	7.736663509558387	0.008851692988930182
111605	7.7219625227189335	0.008851690829287717
111805	7.707261536220452	0.008851688297520477
112005	7.692560550052885	0.008851685840425949
112205	7.677859564137374	0.008851683968874147
112405	7.663158578553312	0.008851681507809683
112604	7.6487458111942965	0.008851679194719964
112805	7.634675337598406	0.00885167698302447
113004	7.6212578569386835	0.008851674513260743
113204	7.608334798400123	0.008851672313270331
113404	7.596010530756319	0.008851670305100257
113604	7.58432017057321	0.008851667801070606
113804	7.573296313338535	0.008851665544127973
114004	7.5646705610717735	0.008851663493500306
114208	7.560936291301335	0.008851661314250414
114408	7.5619502223197905	0.008851658666687579
114608	7.567589924344026	0.008851656656259976
114808	7.5777293057908715	0.008851654327844516
115008	7.592199499799755	0.008851651825561024
115208	7.610798679996644	0.008851649808622852
115408	7.634048504657821	0.00885164744528355
115608	7.6628170008158545	0.008851645222611948
115808	7.696765933693851	0.008851642800750792
116008	7.735535480162836	0.008851640645963554
116208	7.778749885077808	0.008851638194158573
116408	7.826021604583755	0.008851635835011473
123807	7.793624346883217	0.008851542846615157
124007	7.777483198162732	0.00885154047569281
124207	7.7627822318869235	0.008851538141037674
124407	7.748081265906763	0.008851535945806344
124607	7.733380300261873	0.008851533456381011
124807	7.718679334936104	0.00885153108685266
125007	7.703978369920438	0.008851528784290519
125207	7.689277405175763	0.008851526772135854
125407	7.67457644078027	0.008851524179406933
125607	7.660003610888861	0.008851522053288763
125807	7.645887574826878	0.00885151964370201
126007	7.632270140136243	0.008851517376009532
126207	7.619191387268283	0.008851515142016764
126407	7.606689399123925	0.008851512916503435
126607	7.594799980790489	0.008851510543620877
126806	7.583607592026687	0.008851508277733622
127006	7.573775363759782	0.008851506320139757
127206	7.568710314886449	0.008851503852386644
127406	7.568367104902753	0.008851501526922823
127606	7.57265882250181	0.008851499375647887
127807	7.581546041496571	0.008851496919237922
128006	7.594616334142856	0.008851494622325706
128206	7.611923431446251	0.008851492505968839
128406	7.633375740967703	0.008851490293273256
128606	7.66041118931869	0.008851488032785015
128806	7.692700398602957	0.008851485780637436
129006	7.72989110171979	0.008851483465677434
129206	7.771614032584505	0.008851481138347106
129406	7.817487108276351	0.008851478717015304
136809	7.790513202503865	0.00897085570131928
137009	7.775573250388492	0.008976772090645069
137209	7.760855483481168	0.008976297888467322
137409	7.746138510789903	0.0089703995603434

137609	7.731422330858784	0.008964512100344546
137810	7.716633365108576	0.008958635745947685
138009	7.701992343232535	0.00895274171542515
138209	7.687278532588095	0.00894691684416804
138409	7.672565508680277	0.00894107433790814
138609	7.658004736633625	0.008935242963571843
138809	7.643904224699935	0.008929422303936008
139009	7.630305876955325	0.008923612400783432
139209	7.617249892206271	0.008917812977624173
139409	7.60477448578443	0.00891202322586393
139609	7.592915365733645	0.008907130253845263
139809	7.5817044184853035	0.008907128027286326
140009	7.572174777470604	0.008907125984650932
140209	7.567403652788363	0.008907123978016074
140409	7.567345521646484	0.0089071218454011
140609	7.571911757748897	0.00890711958410706
140809	7.580975595927996	0.008907117605767345
141009	7.594375306805454	0.008907115044572755
141204	7.611288612442124	0.008907112945595319
141408	7.633592461185847	0.008907111153316771
141604	7.660197856674736	0.008907109067214147

Thu, 03 Jul 2008 11:48:05 -0700

#### BEHAVIOR VERIFICATION SUMMARY FOR: tankAbramsM1A1 for simple Move Tactically behavior

Behavior inputs from scenario file:  
 /PAIR/compositions/behavior/composite/mr/moveTactically\_CB.xml  
 Properties checked in data collection file:  
 entity\_CollectionTestTank\_2007-08-15-09-49-29-496.xml  
 The COMMANDED SPEED from scenario, 9.0m/s (32.4 Km/hr).  
 Initial Coordinates,

x: -287361.97703322954  
 y: -5464905.16566183  
 z: 3265213.5159455426

Target Coordinates,

x: -284493.6358636792  
 y: -5465037.970153228  
 z: 3265149.889087235

Total distance 2871.41395446089m from Initial to Target.

#### TEST SUMMARY

##### 1. Deviation of Entity's speed

Output generated when Entity's speed deviates more than 15.0% down  
 or more than 5.0% up from commanded speed 9.0m/s.

Total speed measurements: 1666  
 Number of entries where Entity's speed was more than 15.0% down than commanded speed: 279  
 Number of entries where Entity's speed was more than 5.0% up than commanded speed: 5

The data set for SPEED measurements that deviate from the commanded speed is located in file:  
 entity\_CollectionTestTank\_2007-08-15-09-49-29-496.xml\_Speed\_Test\_Rpt.txt

The column format for data items in that file:

TIME CURRENT\_SPEED SLOPE\_OF\_TERRAIN

The RAW data set of Speed measurements is located in file:

entity\_CollectionTestTank\_2007-08-15-09-49-29-496.xml\_Speed\_Raw\_Data.txt

This RAW data file is TAB delimited with NO HEADERS. The file format is similar to the REPORT file:

TIME CURRENT\_SPEED SLOPE\_OF\_TERRAIN

2. Deviation of Entity's Distance To Target:

\*Note: Distance calculations use (x, y) coordinates ONLY.

Total Entity Coordinate entries: 1666

Total number of Entity entries when distance DOES NOT decrease: 0

ANY Distance to Target measurements that are not strongly less than previous distance are located in file:

entity\_CollectionTestTank\_2007-08-15-09-49-29-496.xml\_Distance\_To\_Target\_Rpt.txt

The column format for data items in that file:

TIME      ENTITY'S\_PREVIOUS\_DISTANCE      ENTITY'S\_CURRENT\_DISTANCE

The RAW data set of Distance To Target measurements is located in file:

entity\_CollectionTestTank\_2007-08-15-09-49-29-496.xml\_Distance\_To\_Target\_Raw\_Data.txt

This RAW data file is TAB delimited with NO HEADERS. The file format is similar to the REPORT file:

TIME      ENTITY'S\_PREVIOUS\_DISTANCE      ENTITY'S\_CURRENT\_DISTANCE

3. Deviation of Entity's Location from the route to the target location.

Output generated when Entity deviates more than 10m from the commanded route.

\*Note: Distance calculations use (x, y) coordinates ONLY.

Total Entity Coordinate entries: 1666

Number of entries where Entity deviates more than 10m from the route: 931

MAX deviation detected for this scenario: 23.1664158903602m

The data set of deviation measurements is located in file:

entity\_CollectionTestTank\_2007-08-15-09-49-29-496.xml\_Distance\_Deviation\_Rpt.txt

The column format for data items in that file:

TIME      ENTITY'S\_DEVIATION\_FROM\_THE\_PATH

The RAW data set of Deviation Distance measurements is located in file:

entity\_CollectionTestTank\_2007-08-15-09-49-29-496.xml\_Distance\_Deviation\_Raw\_Data.txt

This RAW data file is TAB delimited with NO HEADERS. The file format is similar to the REPORT file:

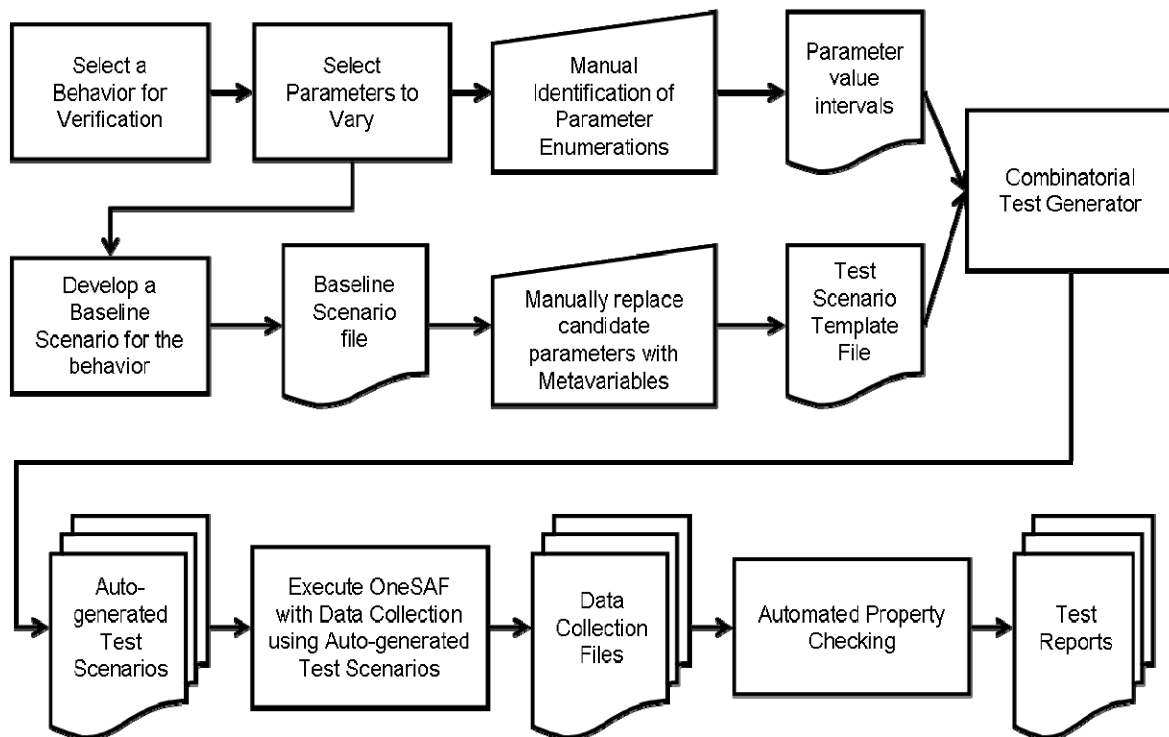
TIME      ENTITY'S\_DEVIATION\_FROM\_THE\_PATH

## APPENDIX F MOVE TACTICALLY (MT) PRESENTATION REPORT

Phase 1 Prototype Report  
28 August 2007

Executive Summary: TRAC-Monterey has successfully created a working prototype of the OneSAF Behavior Verification Automation tool. In its prototype form, the software developed for this project auto-generates executable OneSAF scenarios and checks the output of data files collected from OneSAF during the execution of these auto-generated scenarios for specified parameter characteristics.

The following is a flow chart representation of the prototype operation.



Prototype Execution: The following portion of the report will describe each portion of the Prototype software flowchart in detail.



The first step in the execution of the prototype software is to select a behavior for verification. In the case of this prototype demonstration, we have selected the MoveTactically behavior for simplicity. It should be noted again, that the tests executed in this prototype demonstration are intentionally simple, and are not intended to constitute a complete verification. We chose move tactically over more robustly

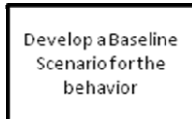
documented behaviors, such as Emplace controlled Minefield in order to facilitate our early work on OneSAF version 1.1. It is also important to stress that this is a prototype, intended to demonstrate the feasibility of the Behavior Verification Automation Concepts. Therefore, it was more important for us to use a behavior that functioned on our existing OneSAF setups, rather than wait to get the most current version of OneSAF functioning with new, robust behaviors. Future work will focus on traceable verification of those robust behaviors using the techniques demonstrated in this prototype.



The next step was to select the parameters of the behavior which we will vary based on the selected scenario. For our purposes in this prototype, we selected 6 parameters to vary in the MoveTactically behavior:

- Movement Technique
- Speed
- Formation Spacing
- Hitch / Unhitch
- Halt Duration
- Weapons Control Status

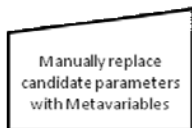
It is important to note that although we input a final destination, we did not vary that parameter. For the prototype, we consciously limited the number of parameters we varied in order to facilitate the demonstrative nature of this report.



We then developed a simple baseline scenario for the behavior. In the case of this execution, the baseline scenario consisted of an Abrams Tank, located in open terrain. The tank is given the Composite Behavior MoveTactically, to a destination point nearby. The destination location is not uniquely significant, though for the purposes of this prototype demonstration, a location was chosen without any apparent intervening terrain; approximately 2km away from the tank's starting location.

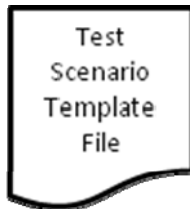


Once manually created, the scenario is saved using OneSAF GUI. This saved scenario file is used as the baseline for our scenario generation. All parameters that will be varied must have an input entered into them, so that they will be changed from their default values. During our initial development, we found that the structure of the scenario file was different depending on whether or not default values were used.



In this manual process, the specific parameter values in the Baseline Scenario are replaced with the metavariables used by the combinatorial testing tool. This is done by manually editing the scenario XML file.

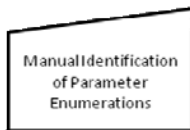




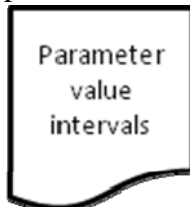
Once, the Baseline Scenario has had its parameter values replaced by metavariables, it is saved as the Test Scenario Template file. This is simply the Scenario XML file with the parameters of interest replaced by metavariables.

Attached example of this file in the “Combinatorial Scenario Generator\Scenario Template” folder:

Scenario.xml



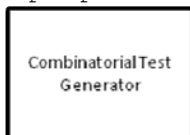
Depending on the given parameter, the value intervals are identified using a combination of the available options in the OneSAF GUI and the enumerations specified in the OneSAF source code. This process is a research / information gathering process.



Once the parameter enumerations are determined, the desired range of values is saved in an input specification file to be used by the Combinatorial Generator. This file will specify either the discrete values to be tested, or a range of possible values, such as a speed parameter between a maximum and minimum value.

Attached example of this file in the “Combinatorial Scenario Generator\Input Description File” folder:

inputSpecification.txt



This is a software process that uses the Input Specification file and the Test Scenario Template File as inputs. The software generates a set of pair-wise combinatorial parameter value tuples from the given input values, and outputs as many test scenarios, based on the Test Scenario Template file, as are required for complete pair-wise combinatorial testing of the parameters.

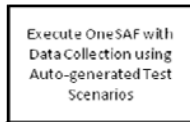


These are the files generated by the Combinatorial Test Generator Process. The number of auto-generated test scenarios corresponds to the number of pair-

wise combinatorial parameter value tuples. With six varied parameters, the Test Generator produced 33 distinct tests scenarios. We have included 5 as examples.

Attached examples of these files in the “Combinatorial Scenario Generator\Output Scenarios” folder. Note that the files are organized as scenario xml files, with corresponding folders:

```
gen_driver0.0
gen_driver1.0
gen_driver2.0
gen_driver3.0
gen_driver4.0
```



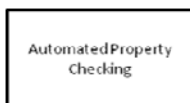
Currently, the resulting auto-generated test scenario files are executed in OneSAF using the Collect Analysis Data option under the Tools menu. In future, we intend this process to be executed in a more automated fashion using scripts or the OneSAF Autopilot mode.



From the OneSAF executions of our test scenarios, we generated these data collection files.

Attached examples of these files are in the “Behavior Verification Prototype\OneSAF Data Collection Files” folder:

```
entity_CollectionTestTank_2007-08-15-09-49-29-496.xml
entity_CollectionTestTank_2007-08-15-16-52-04-366.xml
entity_CollectionTestTank_2007-08-17-15-47-38-512.xml
entity_CollectionTestTank_2007-08-20-16-20-41-807.xml
entity_CollectionTestTank_2007-08-21-10-33-26-867.xml
```



This tool takes the data collection XML files as input, selects necessary data from Data Collection File, and verifies properties of the behavior. These parameter value tests are currently manually written in Ruby. For approximately a 20MB XML data collection file, verification with the scripts took approximately 2-4 seconds per file. In future iterations, we will develop a user interface to automate the generation of those Ruby scripts. Additionally, it is important to note again that these tests were developed for demonstration purposes only. They are not traced back to any documentation. Developing robust tests and documentation traceability will be addressed in future phases of this project, not in this prototype. This prototype is intended to demonstrate feasibility of the methodology.



The Ruby scripts generate text files that contain Test Summary and several data files for each verified property. These files can be used for visualization (graphs) of corresponding aspects of behavior.

Attached examples of these files are in the “Behavior Verification Prototype\Verification Reports and Raw Data output” folder:

```
entity_CollectionTestTank_2007-08-15-09-49-29-496.xml_Distance_Deviation_Raw_Data.txt
entity_CollectionTestTank_2007-08-15-09-49-29-496.xml_Distance_Deviation_Rpt.txt
entity_CollectionTestTank_2007-08-15-09-49-29-496.xml_Distance_To_Target_Raw_Data.txt
entity_CollectionTestTank_2007-08-15-09-49-29-496.xml_Distance_To_Target_Rpt.txt
entity_CollectionTestTank_2007-08-15-09-49-29-496.xml_Speed_Raw_Data
entity_CollectionTestTank_2007-08-15-09-49-29-496.xml_Speed_Raw_Data.txt
entity_CollectionTestTank_2007-08-15-09-49-29-496.xml_Speed_Test_Rpt.txt
entity_CollectionTestTank_2007-08-15-09-49-29-496.xml_Test_Summary_Rpt.txt
entity_CollectionTestTank_2007-08-15-16-52-04-366_2.xml_Distance_Deviation_Raw_Data.txt
entity_CollectionTestTank_2007-08-15-16-52-04-366_2.xml_Distance_Deviation_Rpt.txt
entity_CollectionTestTank_2007-08-15-16-52-04-366_2.xml_Distance_To_Target_Raw_Data.txt
entity_CollectionTestTank_2007-08-15-16-52-04-366_2.xml_Distance_To_Target_Rpt.txt
entity_CollectionTestTank_2007-08-15-16-52-04-366_2.xml_Speed_Raw_Data.txt
entity_CollectionTestTank_2007-08-15-16-52-04-366_2.xml_Speed_Test_Rpt.txt
entity_CollectionTestTank_2007-08-15-16-52-04-366_2.xml_Test_Summary_Rpt.txt
entity_CollectionTestTank_2007-08-17-15-47-38-512_3.xml_Distance_Deviation_Raw_Data.txt
entity_CollectionTestTank_2007-08-17-15-47-38-512_3.xml_Distance_Deviation_Rpt.txt
entity_CollectionTestTank_2007-08-17-15-47-38-512_3.xml_Distance_To_Target_Raw_Data.txt
entity_CollectionTestTank_2007-08-17-15-47-38-512_3.xml_Distance_To_Target_Rpt.txt
entity_CollectionTestTank_2007-08-17-15-47-38-512_3.xml_Speed_Raw_Data.txt
entity_CollectionTestTank_2007-08-17-15-47-38-512_3.xml_Speed_Test_Rpt.txt
entity_CollectionTestTank_2007-08-17-15-47-38-512_3.xml_Test_Summary_Rpt.txt
entity_CollectionTestTank_2007-08-20-16-20-41-807_5.xml_Distance_Deviation_Raw_Data.txt
entity_CollectionTestTank_2007-08-20-16-20-41-807_5.xml_Distance_Deviation_Rpt.txt
entity_CollectionTestTank_2007-08-20-16-20-41-807_5.xml_Distance_To_Target_Raw_Data.txt
entity_CollectionTestTank_2007-08-20-16-20-41-807_5.xml_Distance_To_Target_Rpt.txt
entity_CollectionTestTank_2007-08-20-16-20-41-807_5.xml_Speed_Raw_Data.txt
entity_CollectionTestTank_2007-08-20-16-20-41-807_5.xml_Speed_Test_Rpt.txt
entity_CollectionTestTank_2007-08-20-16-20-41-807_5.xml_Test_Summary_Rpt.txt
entity_CollectionTestTank_2007-08-21-10-33-26-867_4.xml_Distance_Deviation_Raw_Data.txt
entity_CollectionTestTank_2007-08-21-10-33-26-867_4.xml_Distance_Deviation_Rpt.txt
entity_CollectionTestTank_2007-08-21-10-33-26-867_4.xml_Distance_To_Target_Raw_Data.txt
entity_CollectionTestTank_2007-08-21-10-33-26-867_4.xml_Distance_To_Target_Rpt.txt
entity_CollectionTestTank_2007-08-21-10-33-26-867_4.xml_Speed_Raw_Data.txt
entity_CollectionTestTank_2007-08-21-10-33-26-867_4.xml_Speed_Test_Rpt.txt
entity_CollectionTestTank_2007-08-21-10-33-26-867_4.xml_Test_Summary_Rpt.txt
```

## Output Samples:

The following is an example of the test summary output. This test summary, specifically, is found in the file:

```
entity_CollectionTestTank_2007-08-15-09-49-29-496.xml_Speed_Test_Rpt.txt.
Thu, 30 Aug 2007 23:58:00 -0700
        BEHAVIOR VERIFICATION SUMMARY FOR: tankAbramsM1A1
        for simple Move Tactically behavior
Behavior inputs from scenario file:
    /PAIR/compositions/behavior/composite/mr/moveTactically_CB.xml
Properties checked in data collection file:
    entity_CollectionTestTank_2007-08-15-09-49-29-496.xml
The COMMANDED SPEED from scenario, 9.0m/s (32.4 Km/hr).
Initial Coordinates,
    x: -287361.97703322954
    y: -5464905.16566183
    z: 3265213.5159455426

Target Coordinates,
    x: -284493.6358636792
    y: -5465037.970153228
    z: 3265149.889087235

Total distance 2871.41395446089m from Initial to Target.
        TEST SUMMARY
1. Deviation of Entity's speed

Output generated when Entity's speed deviates more than 15.0% down
or more than 5.0% up from commanded speed 9.0m/s.

    Total speed measurements: 1666
    Number of entries where Entity's speed was more than 15.0% down than commanded
speed: 279
    Number of entries where Entity's speed was more than 5.0% up than commanded speed:
5

    The data set for SPEED measurments that deviate from the commanded speed is
located in file:
    entity_CollectionTestTank_2007-08-15-09-49-29-496.xml_Speed_Test_Rpt.txt

    The column format for data items in that file:
    TIME    CURRENT_SPEED    SLOPE_OF_TERRAIN
    The RAW data set of Speed measurments is located in file:
    entity_CollectionTestTank_2007-08-15-09-49-29-496.xml_Speed_Raw_Data.txt
    This RAW data file is TAB delimited with NO HEADERS. The file format is similar
to the REPORT file:
    TIME    CURRENT_SPEED    SLOPE_OF_TERRAIN
2. Deviation of Entity's Distance To Target:
    *Note: Distance calculations use (x, y) coordinates ONLY.
    Total Entity Coordinate entries: 1666
    Total number of Entity entries when distance DOES NOT decrease: 0

    ANY Distance to Target measurements that are not strongly less than previous
distance are located in file:
    entity_CollectionTestTank_2007-08-15-09-49-29-496.xml_Distance_To_Target_Rpt.txt
    The column format for data items in that file:
    TIME    ENTITY'S_PREVIOUS_DISTANCE    ENTITY'S_CURRENT_DISTANCE
    The RAW data set of Distance To Target measurments is located in file:
    entity_CollectionTestTank_2007-08-15-09-49-29-
496.xml_Distance_To_Target_Raw_Data.txt
    This RAW data file is TAB delimited with NO HEADERS. The file format is similar
to the REPORT file:
    TIME    ENTITY'S_PREVIOUS_DISTANCE    ENTITY'S_CURRENT_DISTANCE
3. Deviation of Entity's Location from the route to the target location.

Output generated when Entity deviates more than 10m from the commanded route.
```

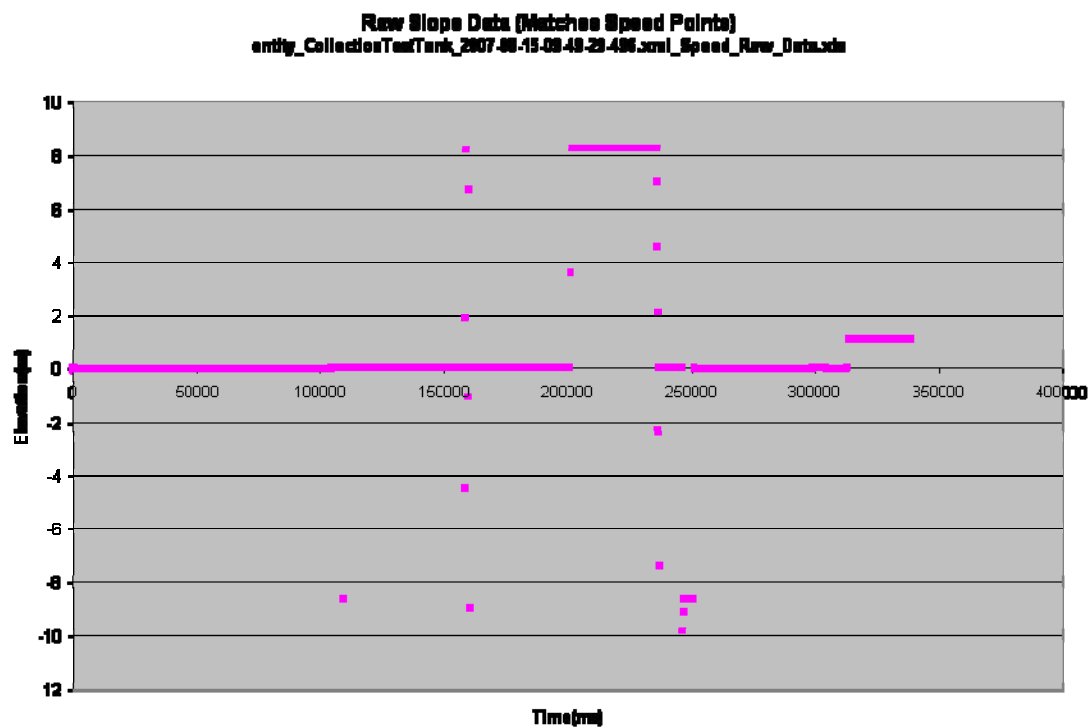
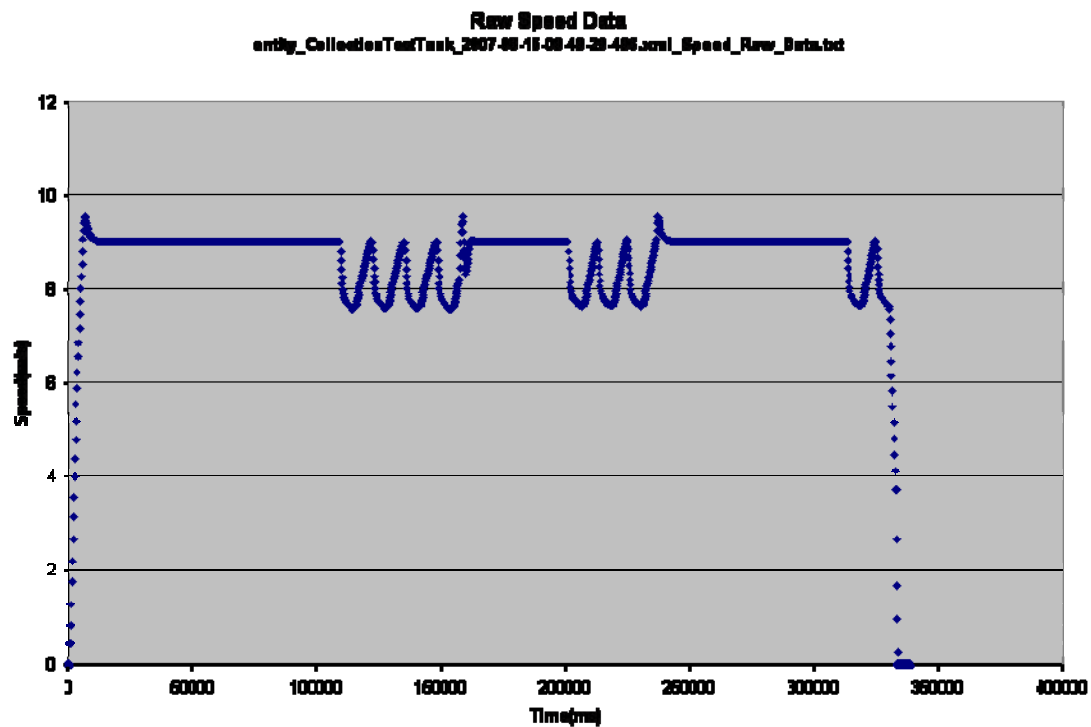
\*Note: Distance calculations use (x, y) coordinates ONLY.  
 Total Entity Coordinate entries: 1666  
 Number of entries where Entity deviates more than 10m from the route: 931  
 MAX deviation detected for this scenario: 23.1664158903602m

The data set of deviation measurements is located in file:  
 entity\_CollectionTestTank\_2007-08-15-09-49-29-496.xml\_Distance\_Deviation\_Rpt.txt  
 The column format for data items in that file:  
 TIME      ENTITY'S\_DEVIATION\_FROM\_THE\_PATH

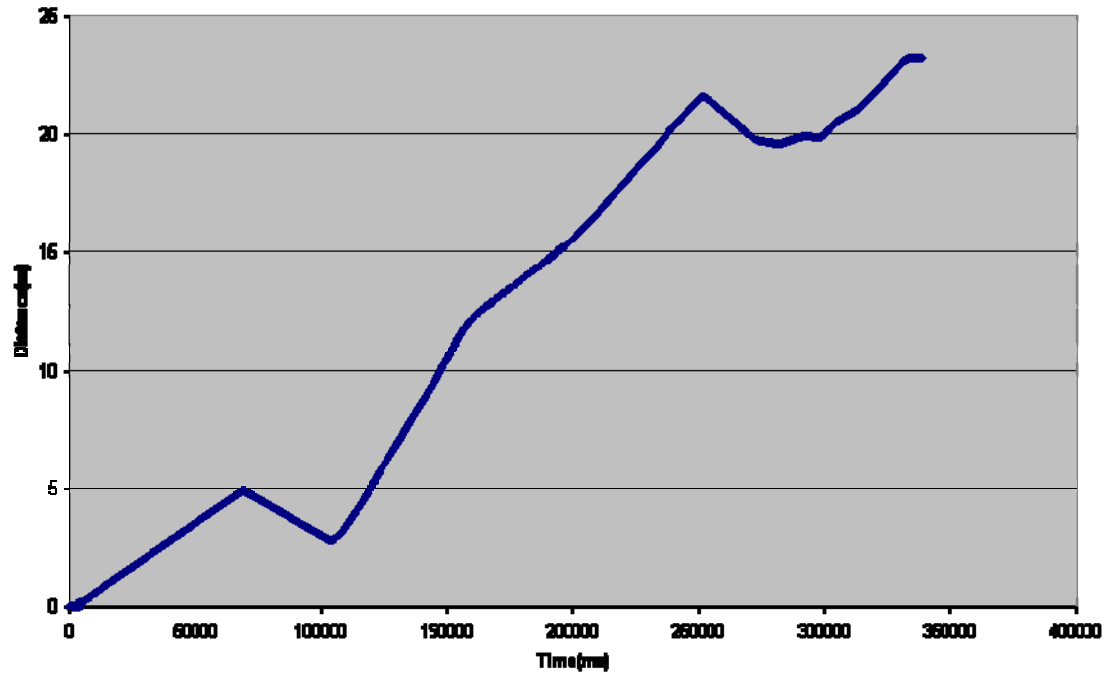
The RAW data set of Deviation Distance measurements is located in file:  
 entity\_CollectionTestTank\_2007-08-15-09-49-29-496.xml\_Distance\_Deviation\_Raw\_Data.txt  
 This RAW data file is TAB delimited with NO HEADERS. The file format is similar to the REPORT file:  
 TIME      ENTITY'S\_DEVIATION\_FROM\_THE\_PATH

### Examples of raw data visualization:

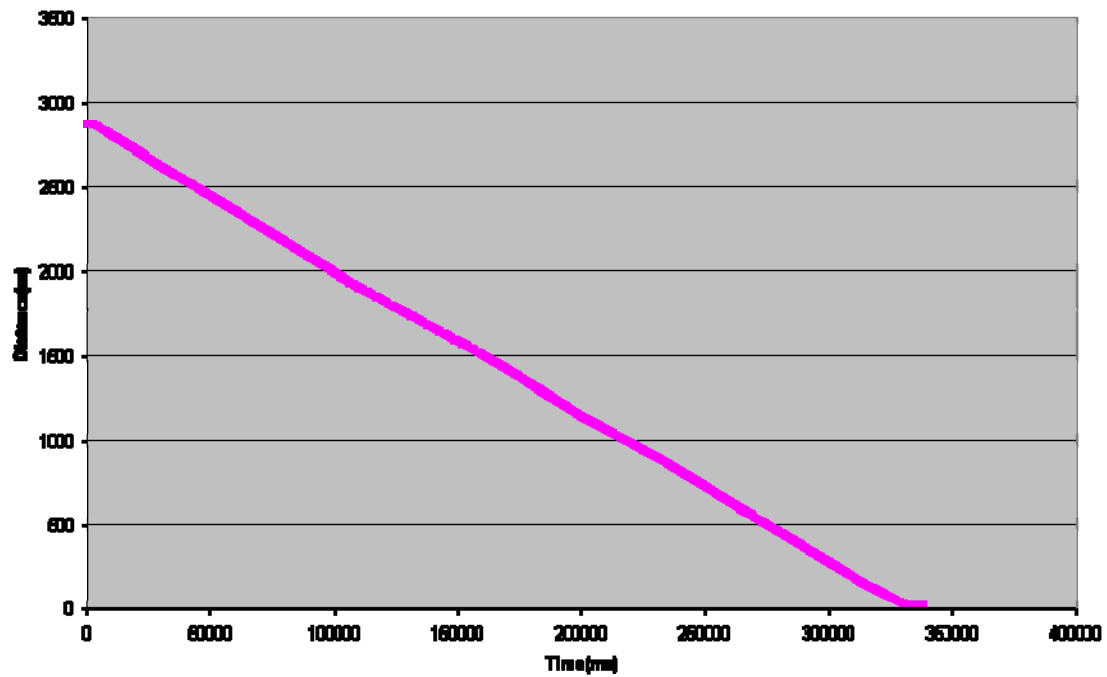
The output text files were imported to Excel to generate the following graphs. For future iterations, we will examine auto generation of data visualization to assist in the verification process.



**Distance From Path Deviation**  
 entity\_CollectionTestTank\_2007-08-15-08:48:28-486.xml\_Distance\_Deviation\_Raw\_Data.txt



**Distance To Target**  
 entity\_CollectionTestTank\_2007-08-15-08:48:28-486.xml\_Distance\_To\_Target\_Raw\_Data.txt



#### Included files:

The following is the file structure and location of the attached files for this prototype. Instructions for implementing the software is located in the Source Code and Documentation folders of each major component of this prototype.

##### Combinatorial Scenario Generator

- Source Code and Documentation
- Scenario Template
- Input Description File
- Output Scenarios

##### Behavior Verification Prototype

- OneSAF Data Collection Files
- Verification Reports and Raw Data Output

#### Limitations of Prototype:

As a prototype, this milestone demonstrates the fundamental concepts which make the Automation of Behavior Verification possible. However, it is a prototype, and at this stage, only a demonstration of concepts. It is not a completed product, and does not execute from start to finish without manual intervention. Similarly, the parameter characteristic tests are not fully mature. Future work will focus on creating “linking software” which binds the significant pieces of software in this prototype together in a user friendly manner. Future work will also focus on developing a means of producing more significant parameter characteristic tests based on expectations extracted from the development documentation.

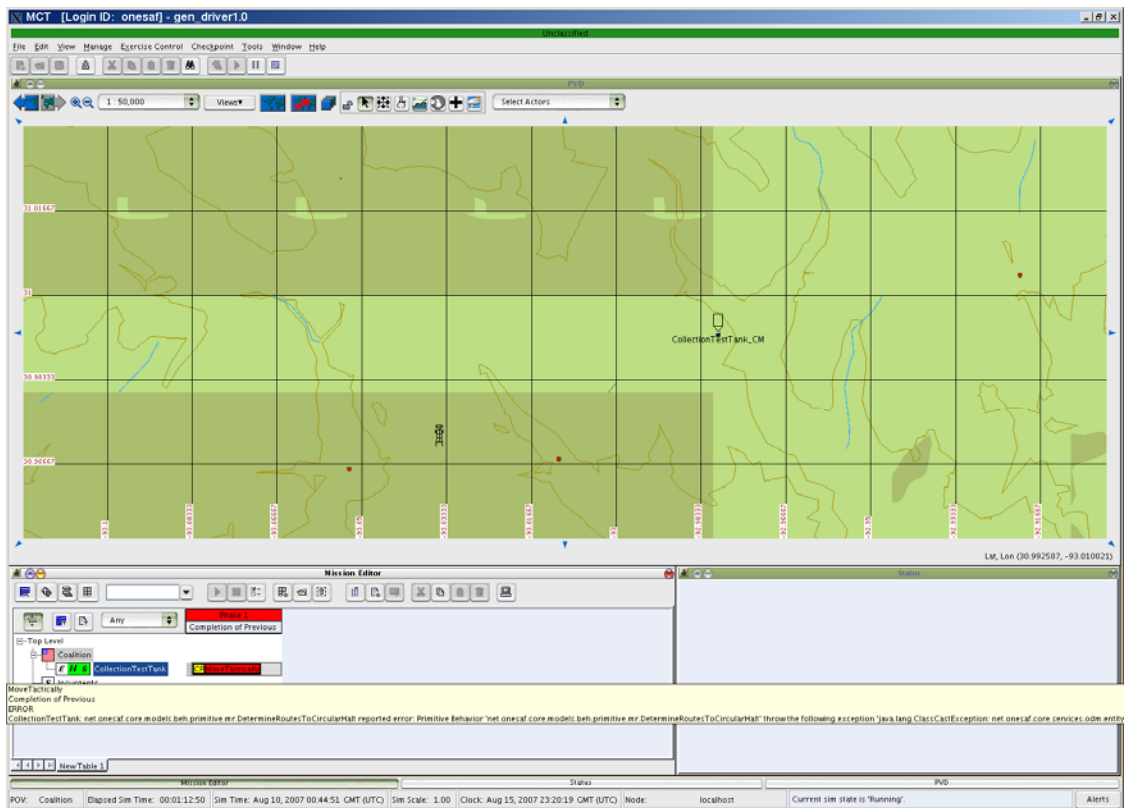
#### Issues:

In the Scenario.xml there are four sets of GCC coordinates. We *assume* the first set contains the Initial (mission start point) followed by the Target (destination, mission end point) coordinates. We would like to know precisely which of the four sets are the actual Initial and Target coordinates.

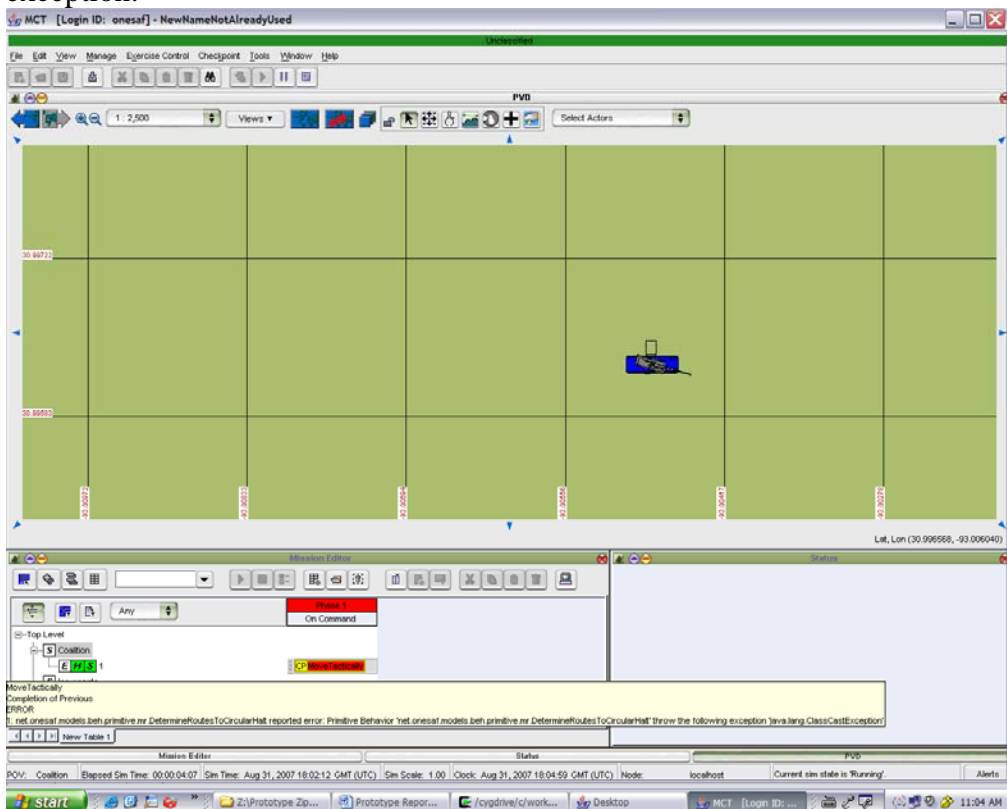
Additionally, when we create our own data collection specification, we could never collect more than the first data point. Consequently, our work-around was to only use the “Collect Analysis Data” function in the Tools menu in the MCT. This provided us with a very basic, but useable, set of information.

Lastly, we encountered a consistent behavior failure when executing our scenarios, both in OneSAF version 1.5 (Engineering Drop) and version 1.1. We verified the fault by reproducing it with manually created scenarios in the MCT. Below are the screen shots from the execution of both the manual and automatically generated scenarios.





The scenario was recreated in OneSAF Version 1.1, and we still received the same exception:



Due to the exception being thrown by the Primitive Behavior `net.onesaf.models.beh.primitive.mr.DetermineRoutesToCircularHalt`, we believe that the source of this fault is setting a `haltDuration` value. 'haltDuration' was randomly selected for the purposes of this Prototype demonstration as a parameter to be varied. The scenarios that successfully completed were those with `haltDuration` set to 0.

#### Conclusion:

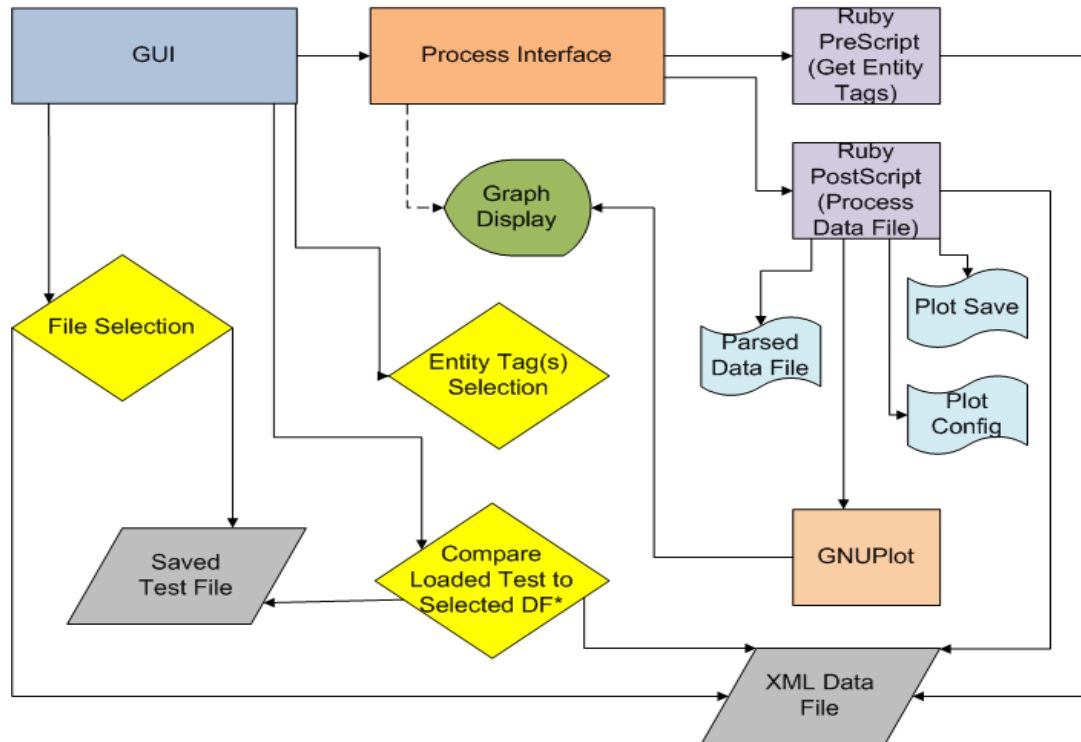
In this prototype, we have successfully demonstrated the feasibility of auto-generating the OneSAF scenario files outside of OneSAF, using our combinatorial test methodology. We also have successfully demonstrated the feasibility of using automated scripts to mine data from the OneSAF generated Data Collection output files to evaluate behavior characteristics of entities.

Future work will focus on three specific areas:

- increasing the level of automation of these processes
- improving traceability of the behavior tests
- creating tools to automate the creating of Ruby test scripts

## APPENDIX G AUTOMATED BEHAVIOR PROPERTY TEST (ABPT) DESIGN DIAGRAM

### ABPT Design Diagram



\* A previously saved test may be applied to a different data file only if the Entity Tags matches.

#### CHANGE #1

1. -----> is new, GUI can call "Open Graphic"
2. PostScript can now call GNUPlot to create plot, then display it!

# Automated Behavior Property Testing (ABPT) Tool

## Design Specification

Automated Behavior Property Testing (ABPT) and Verification Tool is a small set of programs written in Java and Ruby Scripting Language that can parse, analyze, and process specialized XML data collection files generated from OneSAF's Data Collection subcomponent. ABPT is designed with efficiency (minimizing resource usages while maximizing processing speed), portability, and adaptability from concept to deliverable product. The following document provides technical design specifications.

The heart of ABPT is three Ruby scripts, Prescript, Postscript, and Comparator script. These scripts components are designed to communicate with each other and with other applications via standard input/output commonly known as pipe and filter implementation. A graphical user interface (GUI) written in Java with standard Java Swing Class components provides usability to a broader audience for these tools. Between the GUI and the scripts is a Java interface that provides command interpreter services for the GUI, allowing the GUI to run command line applications and passing messages from those applications back to the GUI.

The inputs for Prescript are XML data collection files generated from OneSAF. Prescript parses the data files and returns a set of XML tags that describe characteristics of a particular entity per file parsed. Currently, the GUI allows single data file to be selected, however, the Prescript does allow for multiple files when used directly. Future implementations may remove this limitation if required. The format for Prescript input is the name of a data file for single file usage, or space separated list of files for multiple file. The output of Prescript is sent to standard out (default to the output screen) if used directly, or displayed in the GUI in multi-selectable listing. The format of the output is as follows: `FILENAME ENTITYNAME TAG-1(data type) TAG-2(data type) ... TAG-n(data type) :`. A colon (":") separates each set of tags. There may be some tags that have "child" tags associated with them. The "parent" of those tags will be noted as `"(TAG-x)*"`, notice the parenthesis, an asterisk and no data type. The "child" tag(s) are

noted as “TAG-x(data type)\*”. Lastly, there may be some tags that look like a “parent” tag but do not have any “child” tags.

The format for the input for Postscript is as follows: FILENAME ENTITYNAME TAG-1 TAG-2 ...TAG-n. Each field is space delimited as well. While it may be highly unlikely two have more than two tags as input, the Postscript does not limit the number of tags. However, graphing the parsed data may be limited to two tags as x and y coordinates. Postscript parses the XML data file using the tags specified. The parsed data are written to a data file named as follows: FILENAME.TAG-1.txt. Along with the parsed data file, there are two intermediate files that are also generated. These two files are scripts that allow plotting of the parsed data file and allow the plot to be saved in a file with similar naming convention but in either a Portable Network Graphics (png) or Postscript (ps) depending on the operating system (Windows for former, Linux for latter). When Postscript completes, it returns the name of the plot or an error as to why a plot was not generated. By default, if a plot successfully generates, it will automatically be displayed.

THIS PAGE INTENTIONALLY LEFT BLANK

## APPENDIX H SAMPLE ONESAF ENTITY IN A SCENARIO FILE

```
<actor>
<net.onesaf.core.services.data.dm.rdm.phys.RDMGroundVehicle refID="31" >
  <relativeRank>1</relativeRank>
  <towedEntity>
    <encodableReference refID="0" />
  </towedEntity>
  <currentBehaviorName>null</currentBehaviorName>
  <brakeLightsOn>false</brakeLightsOn>
  <attachment>
    <encodableReference refID="0" />
  </attachment>
  <sectorOfFire>
    <net.onesaf.core.services.data.dm.rdm.phys.SectorOfFire refID="32" >
      <angleFromHeading>0.0</angleFromHeading>
      <angleOfSector>1.0474</angleOfSector>
    </net.onesaf.core.services.data.dm.rdm.phys.SectorOfFire>
  </sectorOfFire>
  <weaponMaxRange>0.0</weaponMaxRange>
  <wcs>Hold</wcs>
  <towingEntity>
    <encodableReference refID="0" />
  </towingEntity>
  <crewState>CREW_HEALTHY</crewState>
  <lowContrast>false</lowContrast>
  <rank>NONE</rank>
  <radarCrossSectionSignatureIndex>0</radarCrossSectionSignatureIndex>
  <unitRole>null</unitRole>
  <engineOn>false</engineOn>
  <priorRouteID>
    <encodableReference refID="0" />
  </priorRouteID>
  <currentBehaviorState>NOT_READY</currentBehaviorState>
  <mass>61326.0</mass>
  <radarEnabled>false</radarEnabled>
  <formationRank>0</formationRank>
  <activity>Undefined</activity>
  <currentRouteID>
    <encodableReference refID="0" />
  </currentRouteID>
  <entityType>tankAbramsM1A1</entityType>
  <movementMedium>NONE</movementMedium>
  <trailingEffectsCode>NoTrail</trailingEffectsCode>
  <smokePlumePresent>false</smokePlumePresent>
  <routeIndex>0</routeIndex>
  <fightingPositionType>none</fightingPositionType>
  <load>0.0</load>
  <oldSpatial>
    <encodableReference refID="0" />
  </oldSpatial>
  <overlay>
    <encodableReference refID="0" />
  </overlay>
  <distanceThreshold>FINE</distanceThreshold>
  <rigStatus>derig</rigStatus>
  <ID>
    <uniqueid refID="33" >
      <stringId>0c42db11-77b3-475c-a778-91c2765db299</stringId>
    </uniqueid>
  </ID>
  <routeOffset>
    <encodableReference refID="0" />
  </routeOffset>
  <followByOffset>
```

```

    <encodableReference refID="0" />
  </followByOffset>
  <address>
    <EPAddress refID="34" >
      <multicast>false</multicast>
      <ID>
        <uniqueid refID="35" >
          <stringId>d9b5dc38-13be-49fa-b71d-d9e54cf7519e</stringId>
        </uniqueid>
      </ID>
    </EPAddress>
  </address>
  <formationPosition>0</formationPosition>
  <entityRole>UNDEFINED</entityRole>
  <repairLevelEnum>Field</repairLevelEnum>
  <contaminationData>
    <encodableReference refID="0" />
  </contaminationData>
  <stuck>false</stuck>
  <sensorMaxRange>0.0</sensorMaxRange>
  <URN>0</URN>
  <tentDeployed>false</tentDeployed>
  <formationRole>null</formationRole>
  <role>NONE</role>
  <spatial>
    <net.onesaf.core.services.data.dm.rdm.phys.SpatialStruct refID="36" >
      <angularVelocity>
        <net.onesaf.core.services.data.dm.rdm.phys.Vector3dStruct refID="37" >
          <z>0.0</z>
          <y>0.0</y>
          <x>0.0</x>
        </net.onesaf.core.services.data.dm.rdm.phys.Vector3dStruct>
      </angularVelocity>
      <linearAcceleration>
        <net.onesaf.core.services.data.dm.rdm.phys.Vector3dStruct refID="38" >
          <z>0.0</z>
          <y>0.0</y>
          <x>0.0</x>
        </net.onesaf.core.services.data.dm.rdm.phys.Vector3dStruct>
      </linearAcceleration>
      <predictionEnum>STATIC</predictionEnum>
      <velocity>
        <net.onesaf.core.services.data.dm.rdm.phys.Vector3dStruct refID="39" >
          <z>0.0</z>
          <y>0.0</y>
          <x>0.0</x>
        </net.onesaf.core.services.data.dm.rdm.phys.Vector3dStruct>
      </velocity>
      <maxExtentVector>
        <net.onesaf.core.services.data.dm.rdm.phys.Vector3dStruct refID="40" >
          <z>0.0</z>
          <y>1.78</y>
          <x>3.67</x>
        </net.onesaf.core.services.data.dm.rdm.phys.Vector3dStruct>
      </maxExtentVector>
      <position>
        <GCC refID="41" >
          <z>3265213.5159455426</z>
          <ellipsoid>WGS_84</ellipsoid>
          <y>-5464905.16566183</y>
          <x>-287361.97703322954</x>
        </GCC>
      </position>
      <orientation>
        <net.onesaf.core.services.geometry.DISEulerAngles refID="42" >
          <psi>-0.04652540040089834</psi>
          <phi>-2.105512361461992</phi>
          <theta>0.0036097009098530826</theta>
        </net.onesaf.core.services.geometry.DISEulerAngles>
      </orientation>
    </net.onesaf.core.services.data.dm.rdm.phys.SpatialStruct>
  </spatial>

```



```

    </net.onesaf.core.services.geometry.DISEulerAngles>
  </orientation>
  <minExtentVector>
    <net.onesaf.core.services.data.dm.rdm.phys.Vector3dStruct refID="43" >
      <z>-1.54</z>
      <y>-1.78</y>
      <x>-3.67</x>
    </net.onesaf.core.services.data.dm.rdm.phys.Vector3dStruct>
  </minExtentVector>
  </net.onesaf.core.services.data.dm.rdm.phys.SpatialStruct>
</spatial>
<mountedOn>
  <encodableReference refID="0" />
</mountedOn>
<currentFuelLevel>0.0</currentFuelLevel>
<name>CollectionTestTank</name>
<moppLevel>Mopp0</moppLevel>
<cfs>Non_CFS</cfs>
<orientationThreshold>FINE</orientationThreshold>
<driverMoving>false</driverMoving>
<configurationName>null</configurationName>
<powerPlantOn>false</powerPlantOn>
<specificRoute>
  <encodableReference refID="0" />
</specificRoute>
<towStatus>false</towStatus>
<damage>NO_KILL</damage>
<compositionName>entity/mr/COMBAT/ARMOR/Tank_M1A1_Abrams_Armor</compositionName>
<affiliation>
  <encodableReference refID="10" />
</affiliation>
<parent>
  <encodableReference refID="0" />
</parent>
<bumperNum>null</bumperNum>
</net.onesaf.core.services.data.dm.rdm.phys.RDMGroundVehicle>
</actor>
</net.onesaf.core.services.data.dm.rdm.org.RDMActorCapabilities>

```

THIS PAGE INTENTIONALLY LEFT BLANK

## APPENDIX I SAMPLE ONESAF DATA COLLECTION FILE

```
<?xml version="1.0" encoding="UTF-8" ?>
<?version version="1.0.0" date="8-15-2007" ?>
<?copyright statement="This work was generated under U.S. Government contract and the government has unlimited data rights
therein." classification="Unclassified" projectName="OneSAF Objective System Architecture & Integration"
contractNumber="#N61339-00-D-0710" taskOrder="0008" copyrights="Copyrights 2001-2003. Science Applications International
Corporation, Lockheed Martin Information Systems, Dynamics Research Corporation. All rights reserved."
distributionStatementD="DISTRIBUTION AUTHORIZED TO THE DEPARTMENT OF DEFENSE AND U.S. DOD
CONTRACTORS ONLY DUE TO CRITICAL TECHNOLOGY, EFFECTIVE 20 JUNE 1994.OTHER REQUESTS SHALL BE
REFERRED TO THE PCO." ?>
<SOD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="entity_CollectionTestTank_2007-08-15-09-49-29-496.xsd" >
<HEADER>
<HEADER_DATA refID="1" >
<DCS>/PAIR/dataCollection/DCS_Entity_CollectionTestTank.xml</DCS>
<RUNID>1</RUNID>
<SCENARIOName>/PAIR/scenario/toolTest/gen_driver0.0/Scenario.xml</SCENARIOName>
<START>1186702320703</START>
</HEADER_DATA>
</HEADER>
<ENTRIES>
<ENTRY refID="2" >
<VALUES>
<net.onesaf.core.models.beh.primitive.mr.PlanMount refID="3" >
<mountMappings>
</mountMappings>
<distances></distances>
<overloadFlag>false</overloadFlag>
<taskAction>hitchToEntity</taskAction>
<towMappings>
</towMappings>
</net.onesaf.core.models.beh.primitive.mr.PlanMount>
</VALUES>
<TYPE>PlanMount</TYPE>
<TIME>21</TIME>
<COMPONENT>BEHAVIOR</COMPONENT>
<ID>
<uniqueid refID="4" >
<stringId>0c42db11-77b3-475c-a778-91c2765db299</stringId>
</uniqueid>
</ID>
</ENTRY>
<ENTRY refID="5" >
<VALUES>
<net.onesaf.core.models.beh.primitive.com.AssertFactOrState refID="6" >
<collActivatedEntityName></collActivatedEntityName>
<inPrimaryPosition>false</inPrimaryPosition>
<overwatchLocation>
<double>0.0</double>
<double>0.0</double>
<double>0.0</double>
</overwatchLocation>
<ENTRY refID="7" >
<VALUES>
<net.onesaf.core.models.beh.primitive.com.AssertFactOrState refID="8" >
<collActivatedEntityName></collActivatedEntityName>
<inPrimaryPosition>false</inPrimaryPosition>
<overwatchLocation>
<double>0.0</double>
<double>0.0</double>
<double>0.0</double>
</overwatchLocation>
<ENTRY refID="9" >
<VALUES>
```

```

<net.onesaf.core.models.beh.primitive.lr.SetSectorOfFire refID="10" >
  <sensorOrientation>0.0</sensorOrientation>
  <sensingArc>60.01159946200243</sensingArc>
</net.onesaf.core.models.beh.primitive.lr.SetSectorOfFire>
</VALUES>
<TYPE>SetSectorOfFire</TYPE>
<TIME>21</TIME>
<COMPONENT>BEHAVIOR</COMPONENT>
<ID>
  <uniqueid refID="11" >
    <stringId>0c42db11-77b3-475c-a778-91c2765db299</stringId>
  </uniqueid>
</ID>
</ENTRY>
<ENTRY refID="12" >
  <VALUES>
    <net.onesaf.core.models.beh.primitive.mr.MoveAlongRoute refID="13" >
      <nextActionLocation>0</nextActionLocation>
      <startAtFirstPoint>>false</startAtFirstPoint>
      <anchorState>>false</anchorState>
      <routeType>CROSS_COUNTRY</routeType>
      <routeCompleted>>false</routeCompleted>
      <linearVelocity>9.0</linearVelocity>
      <gear>Forward</gear>
    </net.onesaf.core.models.beh.primitive.mr.MoveAlongRoute>
  </VALUES>
  <TYPE>MoveAlongRoute</TYPE>
  <TIME>21</TIME>
  <COMPONENT>MoveAlongRoute</COMPONENT>
  <ID>
    <uniqueid refID="14" >
      <stringId>0c42db11-77b3-475c-a778-91c2765db299</stringId>
    </uniqueid>
  </ID>
</ENTRY>
<ENTRY refID="15" >
  <VALUES>
    <DriverFSM refID="16" >
      <currentState>MOVING_ON_ROUTE</currentState>
    </DriverFSM>
  </VALUES>
  <TYPE>BasicData</TYPE>
  <TIME>21</TIME>
  <COMPONENT>BasicData</COMPONENT>
  <ID>
    <uniqueid refID="17" >
      <stringId>0c42db11-77b3-475c-a778-91c2765db299</stringId>
    </uniqueid>
  </ID>
</ENTRY>
<ENTRY refID="18" >
  <VALUES>
    <DirectiveDataCollection refID="19" >
      <RouteName>-</RouteName>
      <NumberOfEntityObstacles>0</NumberOfEntityObstacles>
      <CurrentPath>
        <GCC refID="20" >
          <z>3265213.5159455426</z>
          <ellipsoid>WGS_84</ellipsoid>
          <y>-5464905.16566183</y>
          <x>-287361.97703322954</x>
        </GCC>
        <GCC refID="21" >
          <z>3265199.934094777</z>
          <ellipsoid>WGS_84</ellipsoid>
          <y>-5464965.762319956</y>
          <x>-286363.90710664436</x>
        </GCC>
      </CurrentPath>
    </DirectiveDataCollection>
  </VALUES>

```

```

    </CurrentPath>
    <NumberOfTerrainObstacles>2</NumberOfTerrainObstacles>
  </DirectiveDataCollection>
</VALUES>
<TYPE>BasicData</TYPE>
<TIME>21</TIME>
<COMPONENT>BasicData</COMPONENT>
<ID>
  <uniqueid refID="22" >
    <stringId>0c42db11-77b3-475c-a778-91c2765db299</stringId>
  </uniqueid>
</ID>
</ENTRY>
<ENTRY refID="23" >
  <VALUES>
    <WeaponControlModel refID="24" >
      <targetSpeed>0.0</targetSpeed>
      <wcs>Free</wcs>
      <targetLocation>null</targetLocation>
      <weaponType>null</weaponType>
      <targetRange>0.0</targetRange>
      <munitionType>null</munitionType>
      <currentTargetType>null</currentTargetType>
      <targetActivity>false</targetActivity>
      <acquisitionLevelAchieved>null</acquisitionLevelAchieved>
      <perceptionTime>0</perceptionTime>
      <suppression>false</suppression>
      <targetDirection>null</targetDirection>
    </WeaponControlModel>
  </VALUES>
  <TYPE>tankAbramsM1A1</TYPE>
  <TIME>309</TIME>
  <COMPONENT>WEAPON_CONTROLLER</COMPONENT>
  <ID>
    <uniqueid refID="25" >
      <stringId>0c42db11-77b3-475c-a778-91c2765db299</stringId>
    </uniqueid>
  </ID>
</ENTRY>
<ENTRY refID="26" >
  <VALUES>
    <SuppressionSpeedLimit refID="27" >
      <beingSuppressed>false</beingSuppressed>
      <dayNight>Day</dayNight>
      <entityType>tankAbramsM1A1</entityType>
      <maxSpeed>18.61</maxSpeed>
    </SuppressionSpeedLimit>
  </VALUES>
  <TYPE>tankAbramsM1A1</TYPE>
  <TIME>408</TIME>
  <COMPONENT>MOBILITY_CONTROLLER</COMPONENT>
  <ID>
    <uniqueid refID="28" >
      <stringId>0c42db11-77b3-475c-a778-91c2765db299</stringId>
    </uniqueid>
  </ID>
</ENTRY>
<ENTRY refID="29" >
  <VALUES>
    <SpeedDataCollection refID="30" >
      <modelWithTheLowestSpeed>FORMATION_SPEED_LIMITER</modelWithTheLowestSpeed>
      <CommandedSpeed>9.0</CommandedSpeed>
    </SpeedDataCollection>
  </VALUES>
  <TYPE>BasicData</TYPE>
  <TIME>408</TIME>
  <COMPONENT>BasicData</COMPONENT>
  <ID>

```

```

    <uniqueid refID="31" >
      <stringId>0c42db11-77b3-475c-a778-91c2765db299</stringId>
    </uniqueid>
  </ID>
</ENTRY>
<ENTRY refID="32" >
  <VALUES>
    <CCTTGroundVehicleMobilityModel refID="33" >
      <vehBin>highMobilityTracked</vehBin>
      <requestedLinearAcceleration>2.451675</requestedLinearAcceleration>
      <slope>-0.007060990631857278</slope>
      <brakeForce>0.0</brakeForce>
      <longWeight>0.0</longWeight>
      <currentSpeed>0.42843525442672326</currentSpeed>
      <linearAcceleration>0.0</linearAcceleration>
      <entityLocation>GCC: (-287361.86191023444, -5464905.17195063, 3265213.5139627373)</entityLocation>
      <maxSpeed>18.610000610351562</maxSpeed>
      <stgjCode>619</stgjCode>
      <longSumOfForces>87488.0034785146</longSumOfForces>
      <longFrictionForce>0.0</longFrictionForce>
      <requestedLinearVelocity>9.0</requestedLinearVelocity>
      <vertWeight>0.0</vertWeight>
      <brakeLinearFactor>8.382</brakeLinearFactor>
      <effectiveMu>0.1011556</effectiveMu>
      <brakeDecel>0.0</brakeDecel>
      <forceDriving>87488.0034785146</forceDriving>
      <mass>61326.0</mass>
    </CCTTGroundVehicleMobilityModel>
  </VALUES>
  <TYPE>tankAbramsM1A1</TYPE>
  <TIME>608</TIME>
  <COMPONENT>GROUND_MOBILITY</COMPONENT>
  <ID>
    <uniqueid refID="34" >
      <stringId>0c42db11-77b3-475c-a778-91c2765db299</stringId>
    </uniqueid>
  </ID>
</ENTRY>

```

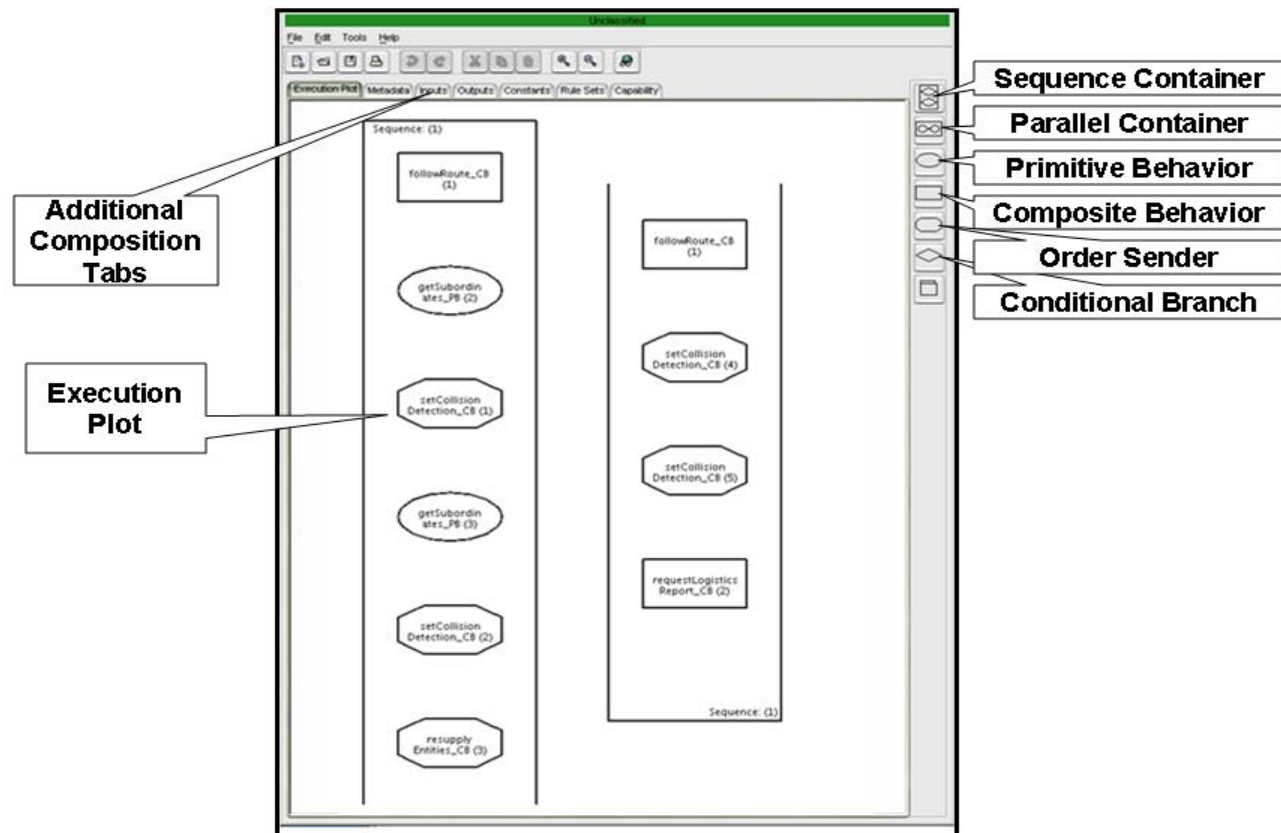
## APPENDIX J TRAC-MONTEREY VERIFICATION PROCESS METHODOLOGY

### Overview

---

- **Problem Statement:** Develop and execute quantitative and qualitative tests that verify that the orderable composite behaviors within the OneSAF Objective System (OOS) perform to their design specifications.
- **Guidance.**
  - Initial: work should be complete by mid-December in advance of the OOS government acceptance testing (GAT) in January.
  - Subsequent: work should continue at current depth of analysis until resources expire; mid-December is no longer a deadline.
- **Current progress.**
  - Developed and refined the methodology.
  - Completed verification of 3 composite behaviors; almost complete with the 4th.
  - Reporting results to PM OneSAF and OOS development team.
- **Projected endstate under current project.**
  - Verification process completed by 03 March 2006.
  - Approximately 10 composite behaviors (of 51) tested (or retested).
  - Endstate approved by PM OneSAF.

# OOS Behavior Composer



06 January 2006

OOS Behavior Model Analysis 2



## VV&A Definitions from DA Pam 5-11

---

- **Verification:** the process of determining that an M&S accurately represents the developer's conceptual description and specifications.
- **Validation:** the process of determining the extent to which an M&S is an accurate representation of the real world from the perspective of the intended use of the M&S.
- **Accreditation:** the official determination that a model, simulation, or federation of M&S is acceptable for use for a specific purpose.

## Background

---

- Behavior modeling is a relatively new concept; no established verification processes specific to behavior models.
- Behavior model verification had not received the attention during OOS development that physical model verification had.
- Primary references include:
  - Suggested verification process from PM OneSAF.
  - TRAC-WSMR primitive behavior verification methodology.
  - AMSAA physical model verification methodology.
  - VV&A Recommended Practices Guide Reference Document (DMSO website).
  - Applicable regulations.

# Constraints, Limitations, Assumptions

---

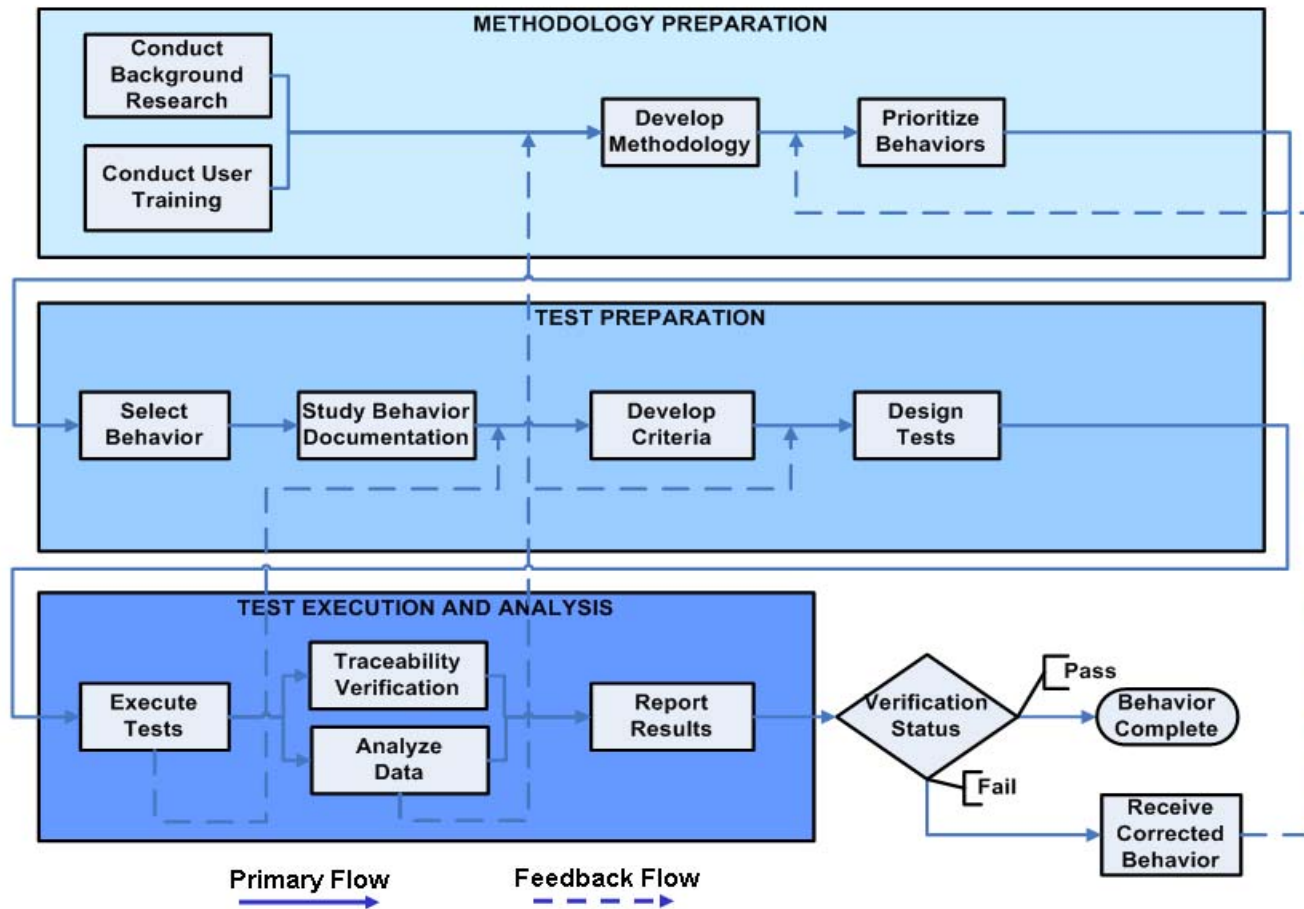
- **Constraints.**
  - We do not have enough resources (primarily manpower) to verify all 51 composite behaviors.
- **Limitations.**
  - Documentation of the behavior implementation is incomplete, limiting our ability to determine with certainty the desired behavior performance.
  - For any given behavior, there are too many potential inputs to test each possible combination.
  - Difficulty in collecting output data will affect the accuracy of our results.
- **Assumptions.**
  - Documentation in conjunction with OOS development team consultations provides enough information to evaluate behavior performance.
  - Testing a representative sample of scenarios for each composite behavior is sufficient to evaluate behavior performance.

# OOS Verification Methodology

---

- Prioritize composite behaviors for testing.
- Select composite behavior to test.
- Study the behavior documentation.
  - Focus on solution space documentation – Use Cases.
  - Refer to problem space documentation, if necessary – Task Descriptions, Process Step Descriptions, and Behavior Process Documents.
- Develop criteria (visual/qualitative and quantitative) to evaluate behavior performance.
- Design tests.
  - Develop scenarios to ensure each behavior input is tested at least once.
  - Choose a representative sampling of units and equipment for each scenario.
  - Include tests of potential points of failure and extreme cases.
- Execute scenarios and record data (visual and quantitative).
  - Conduct excursions (slight variations within a scenario to test parameter of interest further).
  - Cannot use the Data Collection Specification Tool; therefore, quantitative data taken from the Status Window.
- Analyze data, conduct traceability verification, and report results.

# OOS Verification Methodology Flowchart



## Prioritized List of Composite Behaviors (First 15)

---

- Move tactically.
- Attack by fire.
- Mount / dismount.
- Tailgate resupply.
- Occupy position.
- Clear room.
- Send call for fire.
- Move tactically (rotary wing aircraft).
- Attack by fire (rotary wing aircraft).
- Tow to location.
- Attack built up area.
- Conduct raid.
- Execute sniper mission.
- Conduct ambush.
- Conduct air reconnaissance.
- *Full list (60) in backup slides .*

# Example Behavior Test Design

## *Tailgate Resupply (1 of 3)*

---

- **General inputs.**
  - **Trigger** (4 options – on command, completion of previous task, at time, crossing of phase line).
  - **Enable reactions** for this task (yes/no).
- **Required inputs.**
  - **Logistics release point (LRP)** – location where the resupply operation is to take place.
  - **Unit to resupply.**
  - **Return location** – location to which the supplying unit moves after the resupply operation.
- **Optional input.**
  - **Formation** – formation in which the supplying unit moves enroute to the LRP (7 options – wedge, column, line, etc).
- **Rules of engagement (ROE).**
  - **Choice of “use default ROE only” or “allow asset level overrides”.**
  - **Weapons control status** (3 options – free, tight, hold).
  - **Fire control measures** (as needed).

# Example Behavior Test Design

## *Tailgate Resupply (2 of 3)*

---

- **Test design concept.**
  - **Not test engagement or reactive behaviors as part of this test set.**
    - Engagement is not intrinsic to the behavior.
    - Will examine extensively in later behaviors.
  - **Test to ensure each remaining parameter entry has the desired effect.**
    - Supplying unit moves to correct locations and in the correct formation.
    - Correct unit is resupplied.
  - **Test other potential conditions.**
    - Resupply of different supply classes (III, V, and VIII).
    - One or more supply units.
    - Resupply of one or more units, or a sub-element of a single unit.
    - Insufficient quantity of supplies.
    - Incorrect supplies.
    - No supplies.
- **Final number of scenarios – 6.**



# Example Behavior Test Design

## *Tailgate Resupply (3 of 3)*

SCENARIO #	1	2	3	4	5	6
<b>GENERAL SETTINGS</b>						
Resupply Unit Type	Armor	Infantry	Mech Infantry IFV	Military Police	Medical	Field Arty
Resupply Unit Echelon	Platoon	Fire Team	Platoon	Platoon	Section	Platoon
<b>SCENARIO CHARACTERISTICS</b>						
Classes of Supply Delivered	Class III and V	Class V	Class III	Class III & V	Class III & VIII	Classes III & V
Units Near the LRP	Multiple	Single	Multiple	Single	Multiple	Single
Units to be Resupplied	Single	Single	Single	Single	Multiple	Single
Level of Resupply	Subunit(s)	Unit(s)	Subunit(s)	Unit(s)	Unit(s)	Subunit(s)
Req'd Supplies Available?	Yes, all	Yes, some	None	Yes, some	Yes, all	Yes, all
Unreq'd Supplies Available?	Yes	Yes	Yes	No	No	No
Supply Amounts	Sufficient for All Types	Sufficient for All Types	Insufficient for All Types	Sufficient for Some Types	Sufficient for All Types	Sufficient for Some Types
Unit to Resupply	Section A, Armor Platoon 1	Fire Team	Section 2, Mechanized Infantry Platoon 1	Military Police Platoon	Medical Section to receive Class III and VIII. Transport Platoon to receive Class III only.	Section 2, Artillery Platoon
Formation	Vee	Wedge	Column	Line	EchelonLeft	EchelonRight

## **Example Behavior Test Data Collection**

### ***Tailgate Resupply***

---

- **Visual verification.**
  - Movement to the correct LRP.
  - Movement to the correct entities for resupply.
  - Movement to the correct return location.
  - Movement formation.
- **Data verification.**
  - Entities participating in the resupply operation.
  - Accuracy of:
    - Type and amount of supplies delivered.
    - Type and amount of supplies received.

# Example Behavior Test Results

## *Tailgate Resupply*

VERIFICATION RESULTS						
OVERALL VERIFICATION STATUS	Green					
VERIFICATION STATUS BY SCENARIO						
SCENARIO #	1	2	3	4	5	6
Scenario Verification Status	Green	Green	Green	Green	Green	Green
Trigger	Green (Passed)	Green (Passed)	Green (Passed)	Green (Passed)	Green (Passed)	Green (Passed)
Enable Reactions for this Task	Unverified	Unverified	Unverified	Unverified	Unverified	Unverified
LRP Location	Green (Passed)	Green (Passed)	Green (Passed)	Green (Passed)	Green (Passed)	Green (Passed)
Unit to Resupply	Green (Passed)	Green (Passed)	Green (Passed)	Green (Passed)	Green (Passed)	Green (Passed)
Return Location	Green (Passed)	Green (Passed)	Green (Passed)	Green (Passed)	Green (Passed)	Green (Passed)
Formation	Green (Passed)	Amber (Unable to Verify)	Green (Passed)	Green (Passed)	Green (Passed)	Green (Passed)
Weapon Control Status	Unverified	Unverified	Unverified	Unverified	Unverified	Unverified
Fire Control Measures	Unverified	Unverified	Unverified	Unverified	Unverified	Unverified
Supplies Delivered	Green (Passed)	Green (Passed)	Green (Passed)	Green (Passed)	Green (Passed)	Green (Passed)
Supplies Received	Green (Passed)	Green (Passed)	Green (Passed)	Green (Passed)	Green (Passed)	Green (Passed)
Supply Accuracy	Green (Passed)	Green (Passed)	Green (Passed)	Green (Passed)	Green (Passed)	Green (Passed)

## Results – Composite Behavior Functionality

---

- **General**
  - Behavior documentation is insufficient.
- **Move tactically.**
  - **Red (failed verification).**
  - Significant problems with formations, route planning, and individual combatant movement.
  - Questionable results in terms of speed and final orientation.
- **Tailgate resupply.**
  - **Green (passed verification).**
- **Attack by fire.**
  - **Amber (minor faults).**
  - Some problems with weapons control status and some individual combatant behavior execution.
  - Identified faults appear to be due primarily to non-behavior-specific errors.
- **Mount / dismount.**
  - In progress.
- **OOS development team has been taking steps to update behavior documentation and to fix errors in behavior execution.**

# OOS Behavior Model Analysis

## *Prioritized List of Composite Behaviors*

---

- Move tactically.
- Attack by fire.
- Mount / dismount.
- Tailgate resupply.
- Occupy position.
- Clear room.
- Send call for fire.
- Move tactically (rotary wing aircraft).
- Attack by fire (rotary wing aircraft).
- Tow to location.
- Attack built up area.
- Conduct raid.
- Execute sniper mission.
- Conduct ambush.
- Conduct air reconnaissance.
- Conduct ground reconnaissance.
- Platform follow route (fixed wing aircraft).
- Unit follow route (fixed wing aircraft).
- UAV conduct surveillance.
- Conduct repair.
- Conduct casualty movement.
- Conduct MEDEVAC.
- Conduct entity RWA MEDEVAC.
- Conduct entity treatment.
- Passage of lines forward.
- Passage of lines rearward.
- Provide treatment.
- Cross level supply.
- Drop cargo.
- Load/unload supply.
- FARP resupply.
- Prepare for resupply.
- Service station resupply.
- Transfer cargo to basic load.
- Conduct capture rescue.
- Conduct interview.
- Breach wall.
- Clear and mark lane.
- Construct HVID.
- Construct obstacle.
- Cue radar.
- Emplace bridge.
- Emplace minefield.
- Employ smoke.
- Fire and relocate.
- Hitch/unhitch.
- Maneuver and occupy fire support position.
- Perform river crossing.
- Prepare fighting position.
- Retrieve bridge.
- Withdraw.

THIS PAGE INTENTIONALLY LEFT BLANK

# **OneSAF Behavior Verification Automation**



**MAJ Michael Martin, USA**

**Lt John Leo, USN**

**Ms. Jane Wu, Rolands and Associates Corp.**

**9 March 2008**

## Purpose and Agenda

---

**Purpose:** To provide information about the TRAC-Monterey OneSAF Behavior Verification Automation project to the OneSAF User community.

**Agenda:**

- Team
- Verification and Implications
- Methodology
- Combinatorial Scenario Generation
- Automated Property Checking
- Example Test Output



## Team

<i>Person</i>	<i>Organization</i>
MAJ Michael Martin, USA	TRAC-Monterey
Dr. Mikhail Auguston	Naval Postgraduate School Computer Science Department
LT John Leo, USN	Naval Postgraduate School Computer Science Department
Ms. Jane Wu	Rolands and Associates Corp

# Behavior Verification Automation Overview

---

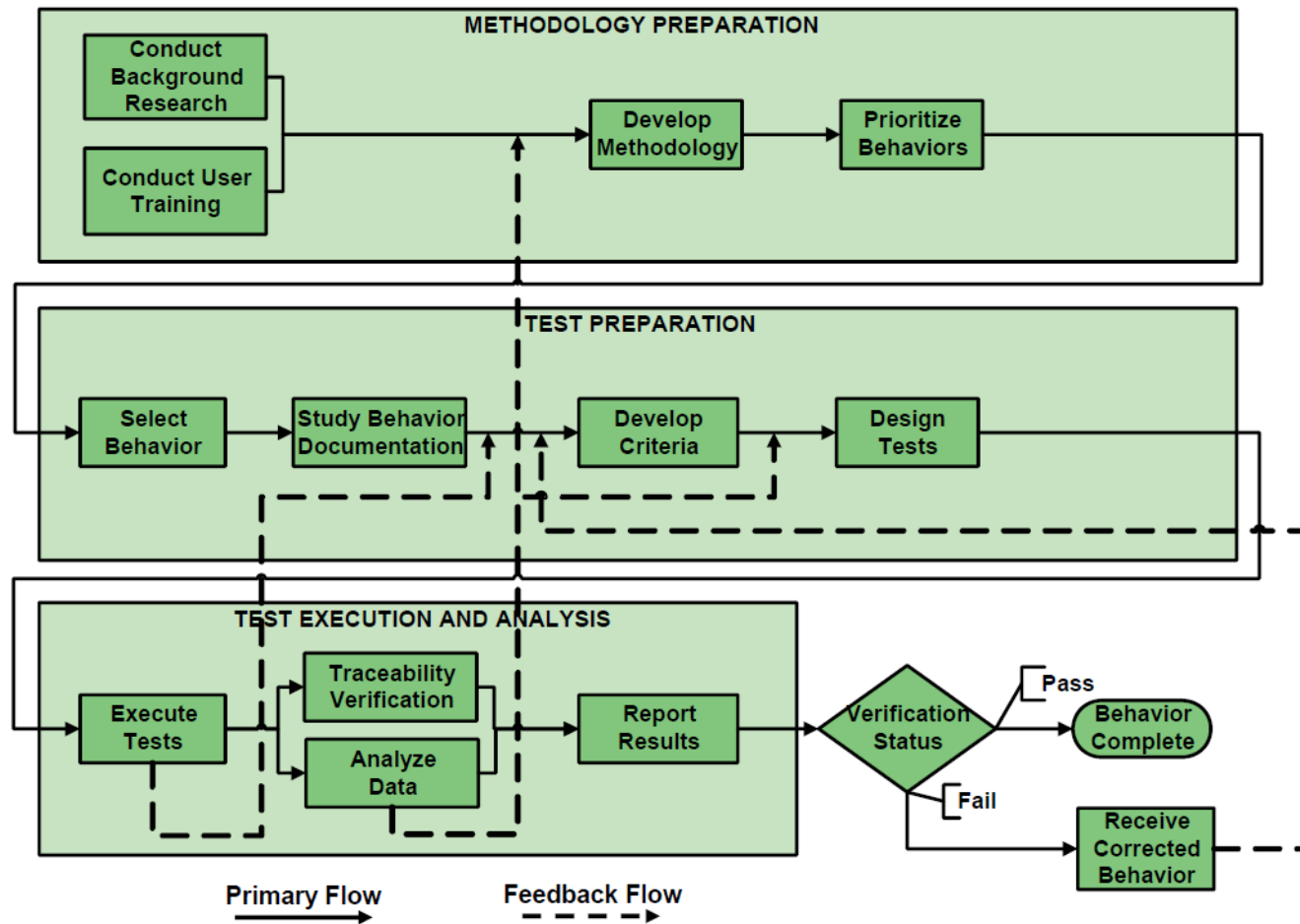
- **Project Objective Statement:** Support the development of OneSAF creating concepts and tools to automate portions of the behavior verification process developed in FY06. The focus for this project will be on automating the verification of orderable behaviors.
- **Project Intent:**
  - The automation will reduce the time and manpower requirements for this process.
  - Integrate an automated behavior verification tool into the OOS Behavior Composer.
  - Provide a generalized methodology for automating the verification of behavior, to include simulation design considerations to facilitate verification.
  - Produce results that are objective and traceable.

## Verification and Implications

---

- **Verification:** The process of determining that a model implementation and its associated data accurately represents the developer's conceptual description and specifications. (*DODI 5000.61, May 13, 2003*)
- **Strict comparison** between the conceptual model and documentation that describes a simulation, and the performance of the simulation.
- **Verification is constrained by documentation.** Checking implementation without traceability in documentation of conceptual modeling crosses the thin line to Validation.
- **Validation.** The process of determining the degree to which a model and its associated data are an accurate representation of the real world from the perspective of the intended uses of the model. (*DODI 5000.61, May 13, 2003*)

# Methodology Flowchart



## **FY 06 Results**

### ***Time Planning Factors***

---

#### **Initial Verification (37-57 hrs):**

- |                  |  |
|------------------|--|
| <b>6-8 hrs</b>   | <b>Documentation review (Use Case, BPDs, TDs, test threads, etc...) and scenario design.</b> |
| <b>24-40 hrs</b> | <b>Behavior testing.</b>   |
| <b>7-9 hrs</b>   | <b>Test results analysis/compiling.</b>  |

#### **Re-verification (19-45 hrs):**

- |                 |  |
|-----------------|--|
| <b>4-6 hrs</b>  | <b>Initial Verification results review, scenario design.</b>   |
| <b>8-40 hrs</b> | <b>Behavior testing varied greatly because only select scenarios were re-tested (anywhere from 2 – 7 scenarios).</b> |
| <b>7-9 hrs</b>  | <b>Test results analysis/compiling.</b>  |



## **FY 06 Results**

### ***Time Planning Factors***

---

#### **Initial Verification (37-57 hrs):**

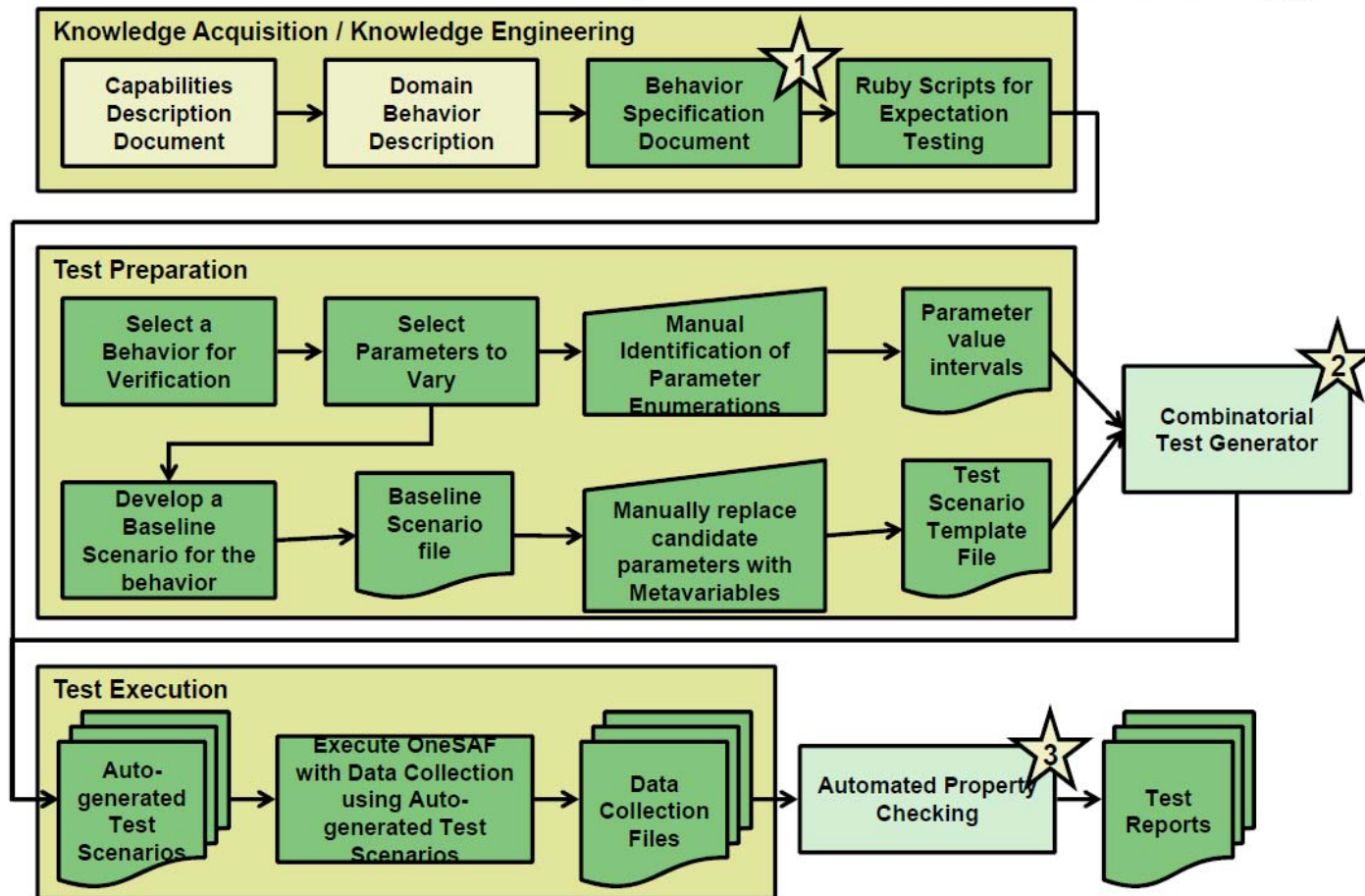
<b>6-8 hrs</b>	<b>Documentation review (Use Case, BPDs, TDs, test threads, etc...) and scenario design.</b>
<b>24-40 hrs</b>	<b>Behavior testing.</b>
<b>7-9 hrs</b>	<b>Test results analysis/compiling.</b>

#### **Re-verification (19-45 hrs):**

<b>4-6 hrs</b>	<b>Initial Verification results review, scenario design.</b>
<b>8-40 hrs</b>	<b>Behavior testing varied greatly because only select scenarios were re-tested (anywhere from 2 – 7 scenarios).</b>
<b>7-9 hrs</b>	<b>Test results analysis/compiling.</b>

# Methodology

## Automation Restructuring of Verification Methodology



## **Methodology:**

### ***Behavior Specification Document (1 of 2)***

---

- **Current expectations for behavior performance are explicitly defined in the Capabilities Definition Document (CDD). Example from “Emplace Controlled Minefield”:**
  - **Expected result 1: Life forms move to the location of the pre-conversion cache, and assume sitting postures. When the interval needed to perform conversion elapses, the designated entities move between the conversion site and the mine dump as many times as may be needed to simulate transportation of the converted mines. The on-hand supplies of mines and conversion kits are decremented from the pre-conversion cache, and incremented at the mine dump cache.**
- **Convert “Natural language” descriptions into “Atomic Expectations” by decomposing the specific wording of the Expectation.**



## **Methodology:**

### ***Behavior Specification Document (2 of 2)***

---

- **Example of Resulting Atomic Expectations:**
  1. **Entities move to the conversion cache.**
    - a. If entity is not at conversion cache, check location relative to conversion cache every 1 minute. If entity is not closer to conversion cache for 10 consecutive minutes, then fail.
    - b. Report time when entity is within 50 meters of conversion cache as “Time 1”.
  2. **At conversion cache, entities assume sitting position.**
    - a. After time 1, if entity is within 30 meters of conversion cache, and does not sit within 1 minute, then fail.
    - b. Report sitting time as “Time 2”.
  3. ...
- **Establish objective criteria and thresholds to account for unexpected situation which might delay objective performance, such as no-go terrain.**
- **Recommend adding “negative expectations” to conceptual modeling: define circumstances when a behavior should fail.**

## **Methodology:**

### ***Combinatorial Scenario Generation***

---

- **Allows rapid, automated creation of a large number of OneSAF scenarios using a baseline scenario.**
- **From baseline scenarios, meta-variables are inserted to mark candidate values to be changed.**
- **List of possible alternate variables to be substituted into each meta-variable is researched and created.**

### **Result:**

- **A set of scenarios that is pair-wise combinatorial exhaustive for identified variables, and are ready to execute in OneSAF.**

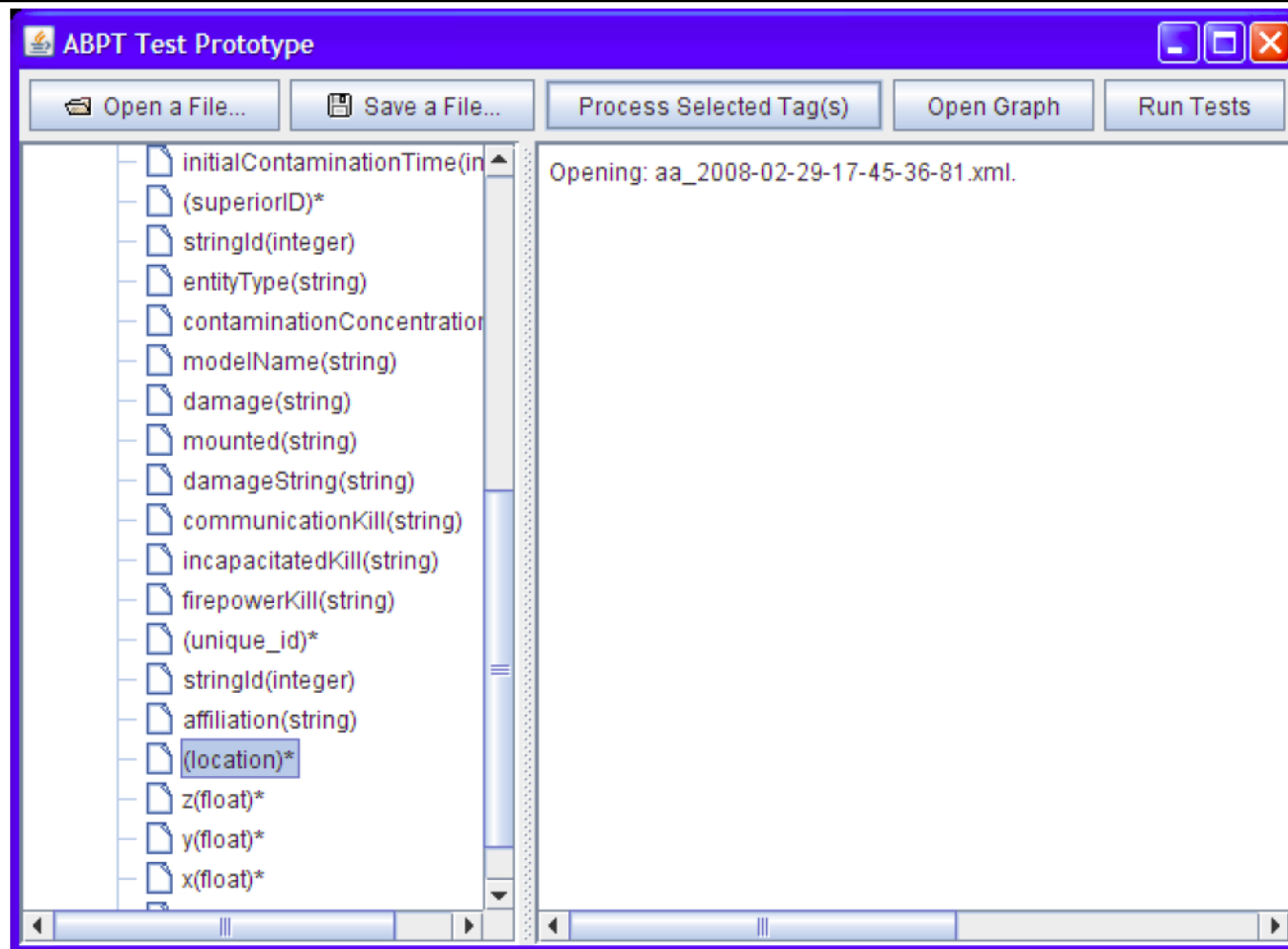
## **Methodology:** ***Automated Property Checking***

---

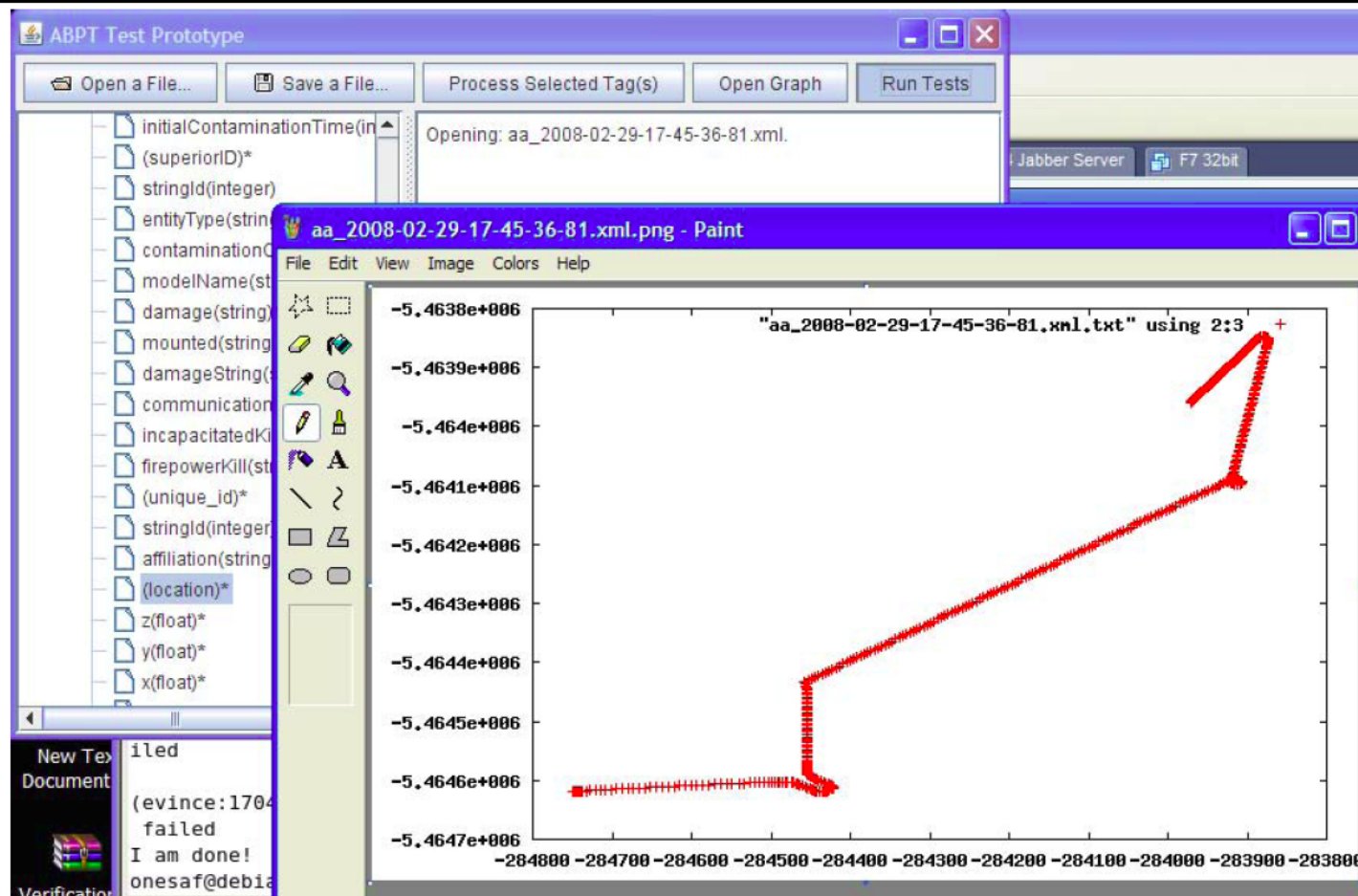
- **Input scenarios from Combinatorial Scenario Generation.**
- **The results of the scenario execution are captured using the built in OneSAF data collection tools.**
- **Quantitative measures of behavior performance evaluated using Ruby script tests.**
- **Tests informed by behavior documentation, providing traceability for expectations.**
- **Ruby is able to evaluate approximately 40 GB of collected data in less than 5 seconds, and renders objective pass or fail evaluations of behavior performance as specified.**
- **Output data format is malleable, and contributes to creating graphs for rapid Examination.**

# Example of Test Execution

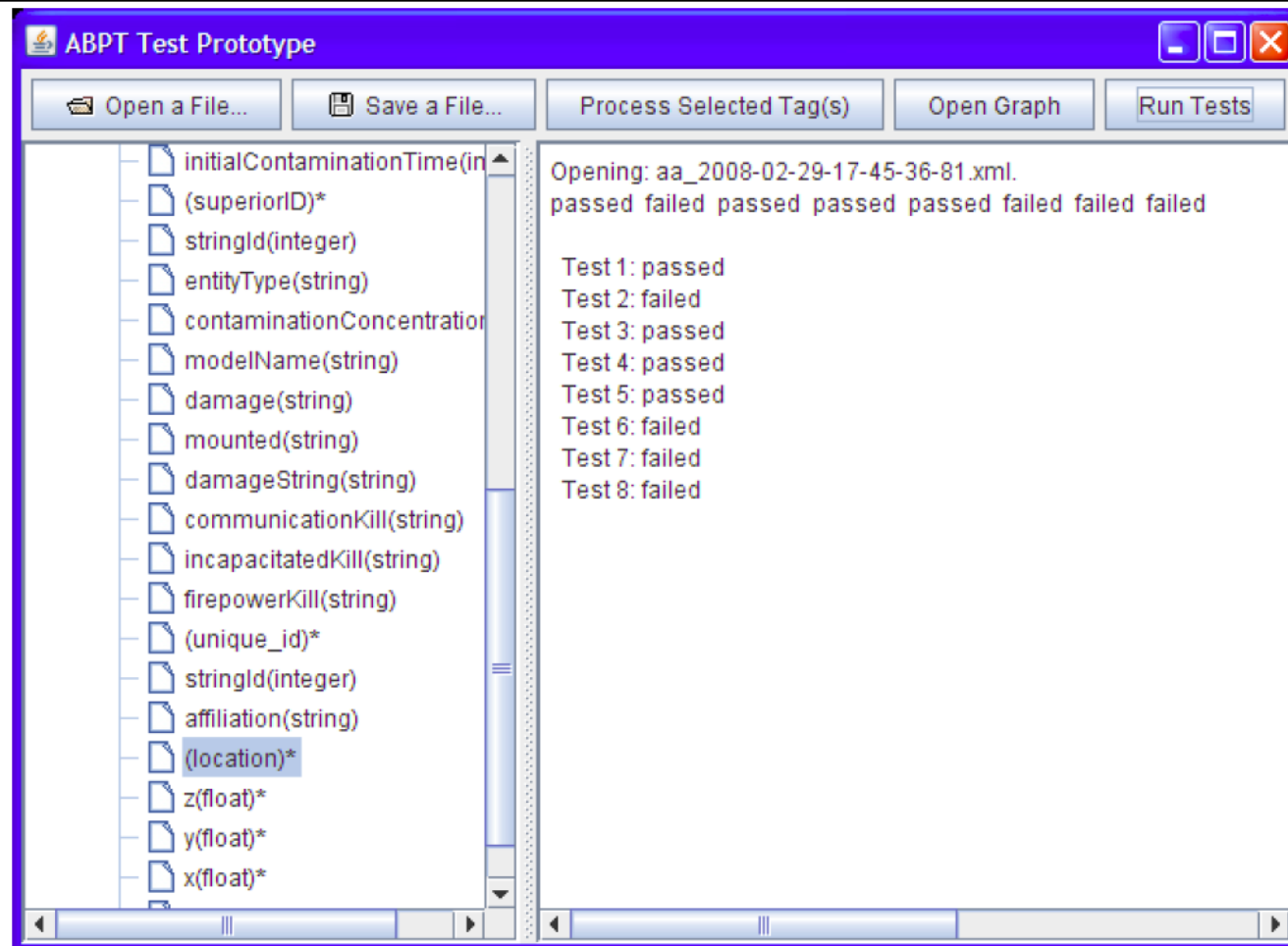
## (1 of 4)



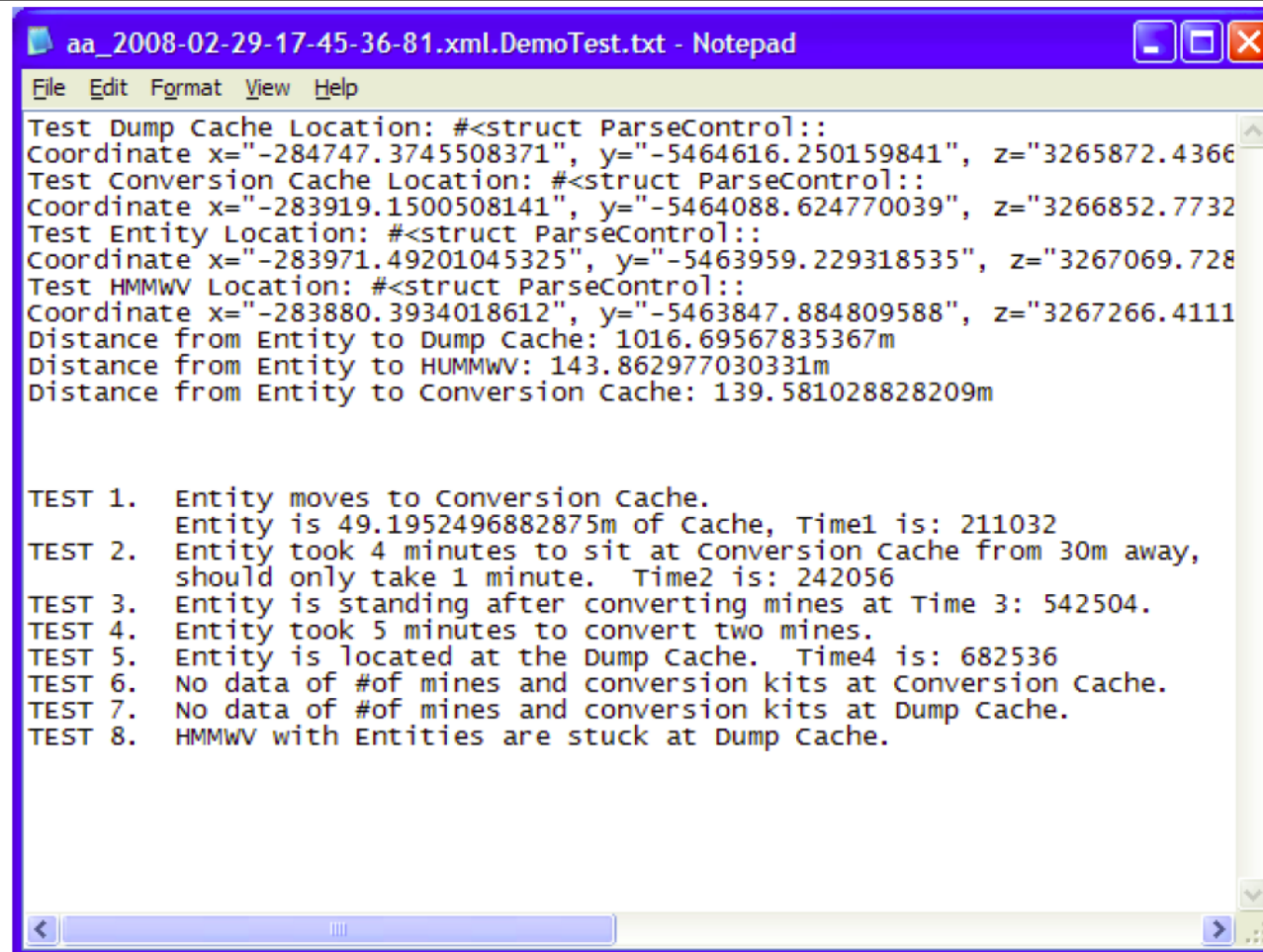
## Example of Test Execution (2 of 4)



## Example of Test Execution (3 of 4)



## Example of Test Execution (4 of 4)



The screenshot shows a Notepad window titled "aa\_2008-02-29-17-45-36-81.xml.DemoTest.txt - Notepad". The window contains the following text:

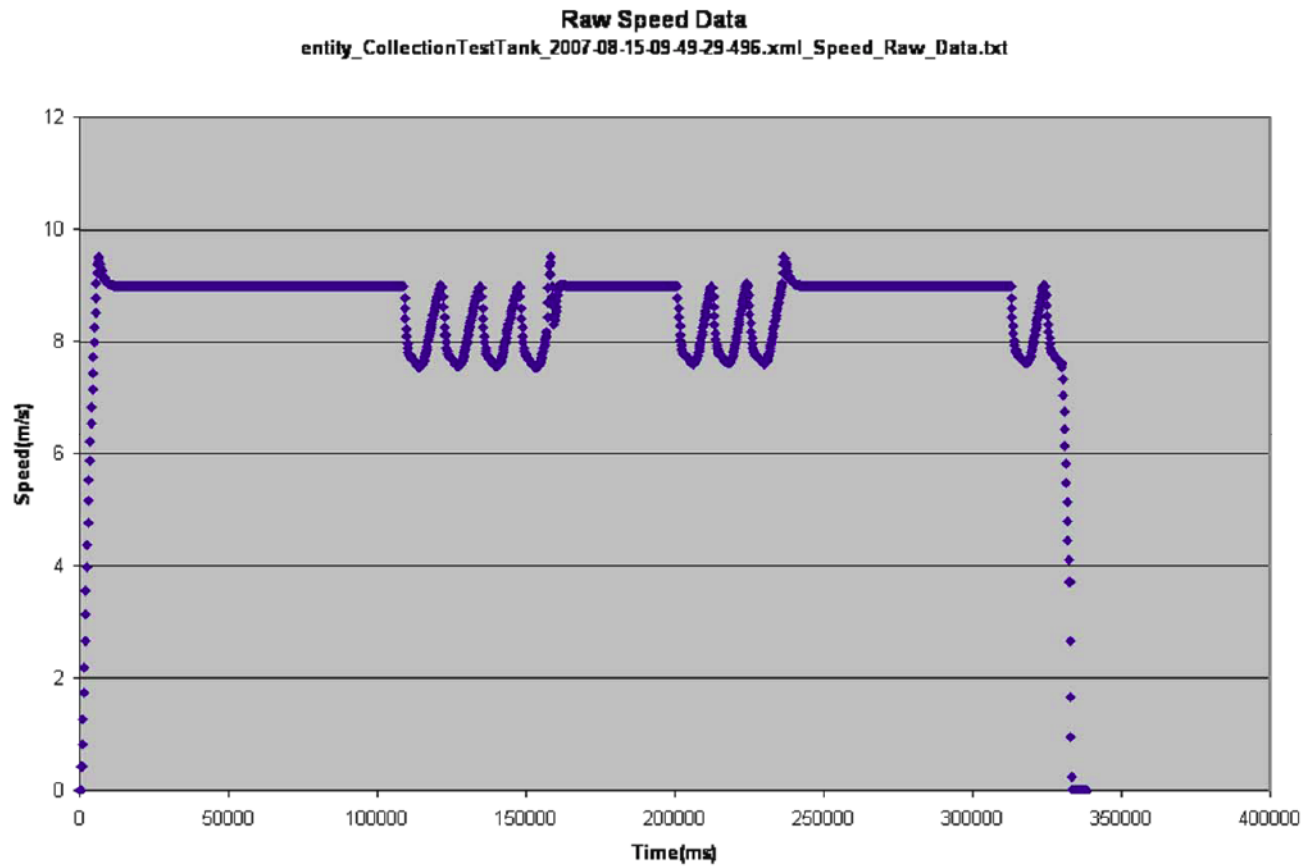
```
Test Dump Cache Location: #<struct ParseControl::
Coordinate x="-284747.3745508371", y="-5464616.250159841", z="3265872.4366
Test Conversion Cache Location: #<struct ParseControl::
Coordinate x="-283919.1500508141", y="-5464088.624770039", z="3266852.7732
Test Entity Location: #<struct ParseControl::
Coordinate x="-283971.49201045325", y="-5463959.229318535", z="3267069.728
Test HMMWV Location: #<struct ParseControl::
Coordinate x="-283880.3934018612", y="-5463847.884809588", z="3267266.4111
Distance from Entity to Dump Cache: 1016.69567835367m
Distance from Entity to HUMMWV: 143.862977030331m
Distance from Entity to Conversion Cache: 139.581028828209m

TEST 1. Entity moves to Conversion Cache.
Entity is 49.1952496882875m of Cache, Time1 is: 211032
TEST 2. Entity took 4 minutes to sit at Conversion Cache from 30m away,
should only take 1 minute. Time2 is: 242056
TEST 3. Entity is standing after converting mines at Time 3: 542504.
TEST 4. Entity took 5 minutes to convert two mines.
TEST 5. Entity is located at the Dump Cache. Time4 is: 682536
TEST 6. No data of #of mines and conversion kits at Conversion Cache.
TEST 7. No data of #of mines and conversion kits at Dump Cache.
TEST 8. HUMMWV with Entities are stuck at Dump Cache.
```

# Examples of Output Data Visualization

## (1 of 3)

---

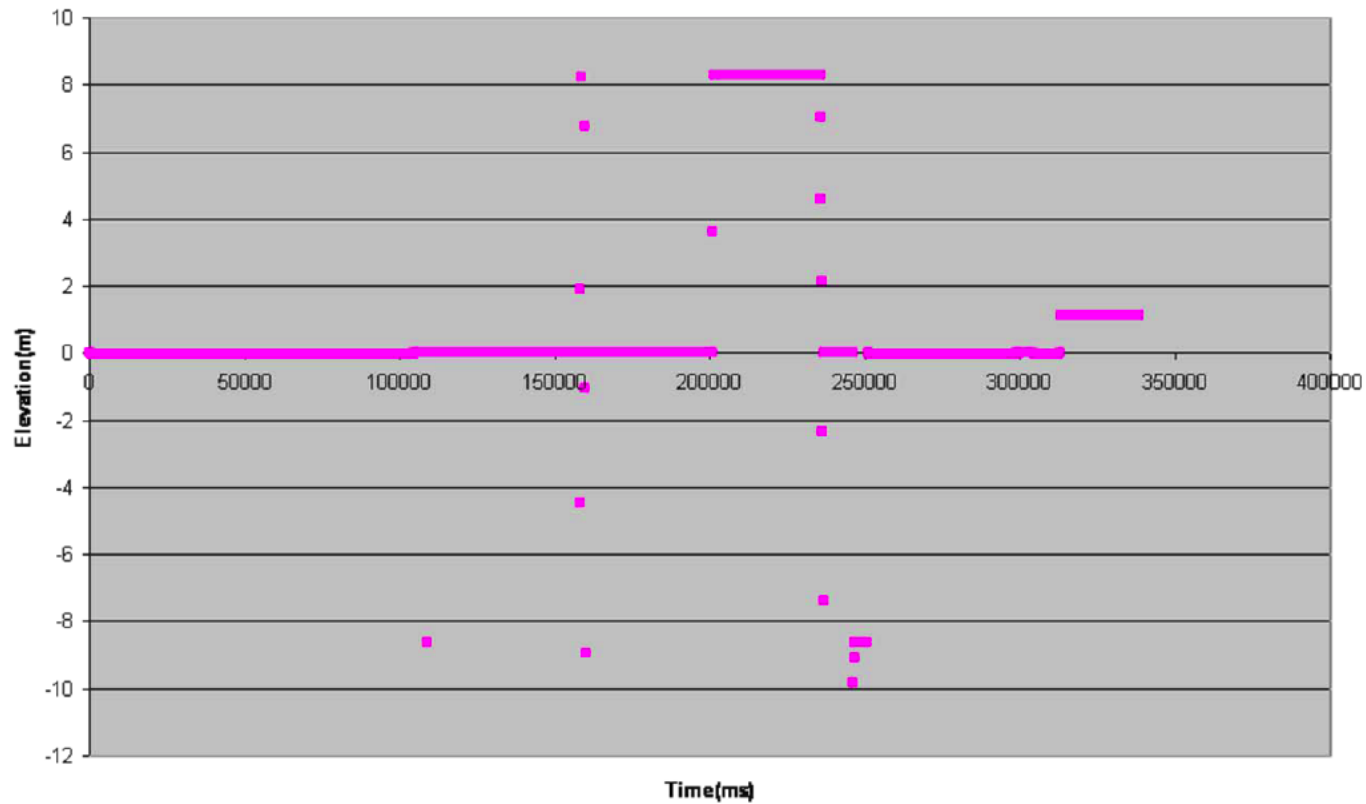




# Examples of Output Data Visualization (2 of 3)

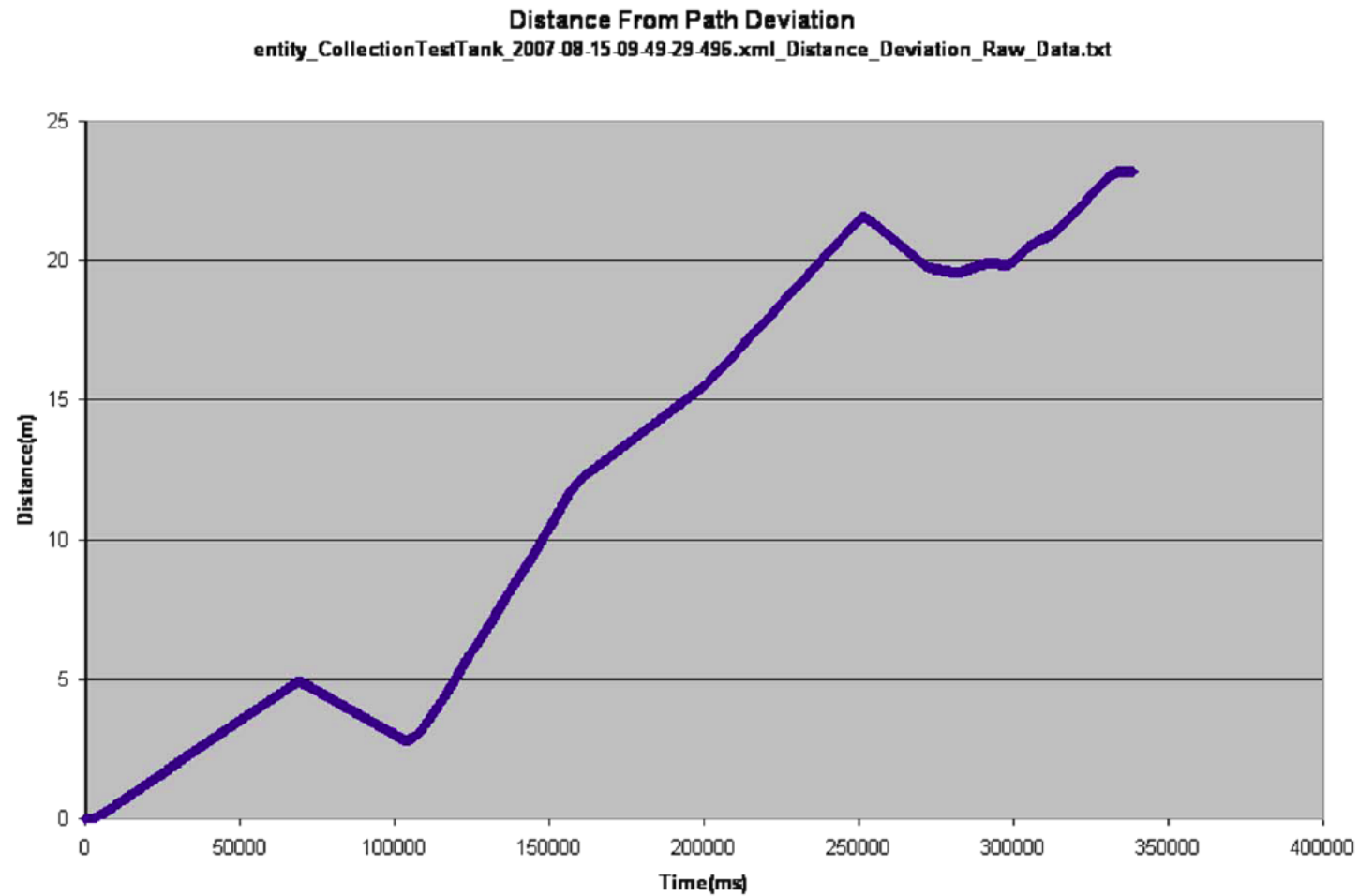
---

Raw Slope Data (Matches Speed Points)  
entity\_CollectionTestTank\_2007-08-15-09-49-29-496.xml\_Speed\_Raw\_Data.xls



# Examples of Output Data Visualization (3 of 3)

---



## Project End State

---

- **OneSAF Behavior Verification Automation methodology addressing OOS specific architecture.**
- **Generalized Behavior Verification Automation methodology.**
- **Recommendations regarding expanding CM\KE process products.**
- **Software to create combinatorial (pair wise) OneSAF scenarios for testing of behaviors.**
- **Software to test data output from OneSAF data collections files against Ruby Scripts.**
- **Feedback on OneSAF data collection tools.**
- **Design documentation of all software products and automation concepts:**
  - **Data collection of behavior inputs.**
  - **Data collection of simulation run outputs.**

# Questions?



**MAJ Michael Martin, USA**

**LT John Leo, USN**

**Ms. Jane Wu, Rolands and Associates**

**9 March 2008**

## **INITIAL DISTRIBUTION LIST**

1. Defense Technical Information Center  
Ft. Belvoir, Virginia
2. Dudley Knox Library  
Naval Postgraduate School  
Monterey, California
3. Dr. Mikhail Auguston  
Naval Postgraduate School  
Monterey, California
4. Dr. Man-Tak Shing  
Naval Postgraduate School  
Monterey, California
5. John K. Leo  
Naval Postgraduate School  
Monterey, California