AFRL-IF-RS-TR-2001-149
Final Technical Report
July 2001


# ABSTRACT MODELING DEMONSTRATION

**RAM Laboratories, Inc.**

**Robert McGraw, Joseph Clark, Allyn Treshansky**

*APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.*

**AIR FORCE RESEARCH LABORATORY**
**INFORMATION DIRECTORATE**
**ROME RESEARCH SITE**
**ROME, NEW YORK**

**Reproduced From**
**Best Available Copy**

20011109 078

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2001-148 has been reviewed and is approved for publication.

APPROVED: DAWN A. TREVISANI
Project Engineer

FOR THE DIRECTOR: JAMES A. CUSACK, Chief
Information Systems Division
Information Directorate

If your address has changed or if you wish to be removed from the Air Force Research Laboratory Rome Research Site mailing list, or if the addressee is no longer employed by your organization, please notify AFRL/IFSB, 525 Brooks Rd, Rome, NY 13441-4505. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, 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.

| 1. AGENCY USE ONLY *(Leave blank)* | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | Jul 01 | Final May 00 - Feb 01 |

**4. TITLE AND SUBTITLE**

ABSTRACT MODELING DEMONSTRATION

**5. FUNDING NUMBERS**

C - F30602-00-C-0083
PE - 62702F
PR - 459S
TA - BA
WU - 02

**6. AUTHOR(S)**

Robert McGraw, Joseph E. Clark, Allyn Treshansky

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

RAM Laboratories, Inc.
6540 Lusk Blvd
Suite C200
San Diego, CA 92121

**8. PERFORMING ORGANIZATION REPORT NUMBER**

BAA 98-01-IJFKPA

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

AFRL/IFSB
525 Brooks Rd
Rome NY 13441

**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

AFRL-IF-RS-TR-2001-149

**11. SUPPLEMENTARY NOTES**

AFRL Project Engineer: Dawn A. Trevisani, IFSB, 315-330-7311

**12a. DISTRIBUTION AVAILABILITY STATEMENT**

Approved for public release; distribution unlimited.

**12b. DISTRIBUTION CODE**

**13. ABSTRACT** *(Maximum 200 words)*

Report developed under BAA contract for CI-98-4-01. This effort demonstrated the feasibility of using model abstraction methods to develop high level representations of complex designs. Abstract models are used to reduce complexity, and thus simulation time, while maintaining the essence of the model. This effort demonstrated the effectiveness of abstract modeling by developing abstract models of high-fidelity JMASS-compliant models. Several model abstraction techniques were examined throughout the course of this effort. Sensitivity analyses were used and modified to identify key simulation parameters within the context of the simulation application. Various abstraction techniques were used to develop the abstract model. The differences in accuracy between the abstract models was noted with respect to the original model set. The difference were illustrated through the use of the JMASS Simview Visualization tool. These differences in model accuracy were examined with respect to simulation time to illustrate the types of tradeoffs that system modelers and analysts face when sacrificing fidelity for timely simulations. A demonstration was developed that illustrated the development of the abstract models, the simulation of these abstractions, a graphical comparison of the accuracy of the models, and an analysis of the tradeoffs between accuracy with respect to simulation time.

**14. SUBJECT TERMS**

Model Abstraction, Simulation Abstraction, C4ISR, ISR, Modeling and Simulation, JMASS

**15. NUMBER OF PAGES**

48

**16. PRICE CODE**

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| UNCLASSIFIED | UNCLASSIFIED | UNCLASSIFIED | UL |

Standard Form 298 (Rev. 2-89) (EG)
Prescribed by ANSI Std. 239.18
Designed using Perform Pro, WHS/DIOR, Oct 94

# Table of Contents

# Table of Figures

## List of Acronyms

| | |
|---|---|
| ASIC | Application Specific Integrated Circuit |
| CAD | Computer Aided Design |
| CBE | Control Byte Enable |
| FCFS | First Come First Served |
| FIFO | First In First Out |
| FPGA | Field Programmable Gate Array |
| HW | Hardware Element |
| I/O | Input/Output |
| IP | Intellectual Property |
| Proc | Processor |
| RTL | Register Transfer Level |
| SW | Software Element |
| VHDL | VHSIC Hardware Description Language |
| VHSIC | Very High Speed Integrated Circuit |

# 1 Introduction

This is the final report for the BAA 98-01-IFKPA project entitled "Model Abstraction Demonstration". This report documents the progress, problems, and solutions encountered during this nine-month demonstration development effort. This report is broken down into the following sections:

Section 2: Overview of the BAA
Section 3: Tasks Accomplished
Section 4: Future Work

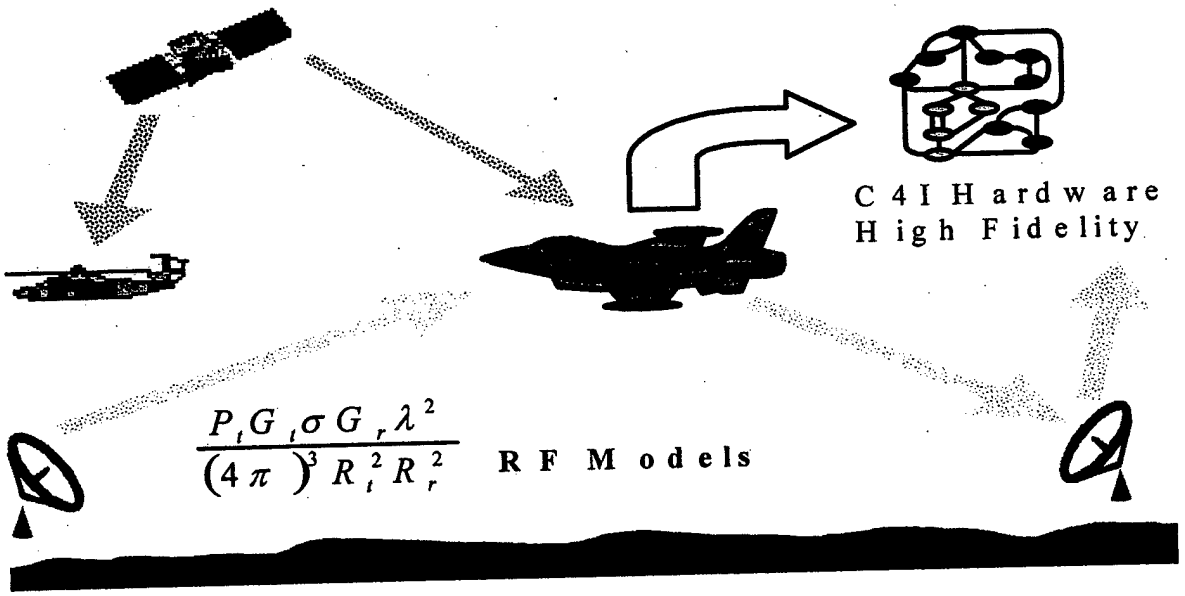



**Figure 1: Complex Simulation Environments**

# 2 Overview of the BAA

This BAA project, entitled "Model Abstraction Demonstration" seeks to demonstrate to the modeling and simulation community the effects of various model abstraction techniques on simulation accuracy. This BAA covers methods for identifying candidate parameters for abstraction, developing abstract model representations, simulating abstract models, and visualizing the results of abstract models with respect to baseline simulations.

## 2.1 Significance of the Problem

Simulations for both the military and commercial industry are geared toward the development of high-fidelity component models that accurately predict system performance, scenario and damage assessments, and mission effectiveness across a variety of paradigms as shown in Figure 1. While yielding very accurate results, fielding such simulation systems does not come without a steep price. These simulations are often costly in terms of not only development time but also simulation time. In order to address the developmental cost issues, simulation efforts have examined interoperability and model re-use as possible solutions. Other efforts have addressed simulation time cost by applying parallel and distributed architectures and frameworks to the simulation systems. One way to address both simulation and development cost issues is to employ model abstraction techniques. Model abstraction techniques allow for re-use of legacy or off-the-shelf models, while reducing model complexity.

Model abstraction "captures the essence of the behavior of a model without all the details of how that behavior is implemented in code." Simulations using abstracted models are more concerned with the qualitative results of a simulation rather than the quantitative results of that simulation. Fidelity that is necessary at some levels of modeling may not be necessary to meet modeling and analysis goals at others. For example, system models representing pollution in an environment may involve models dealing with component models down to the molecular level. However, models dealing with the effects of pollution on sensor performance need not be modeled to that degree of detail. For this reason, detail that may not be needed to meet analysis goals may be dropped from the model. The process of dropping this unneeded information is known as abstract modeling. *A key concern about the effectiveness of abstract modeling lies with whether the savings in simulation time will offset the amount of model accuracy that has been compromised by dropping this information.*

## 2.2 Research Goals

The objective of this effort was to develop a demonstration that illustrates this simulation time versus model accuracy tradeoff. This effort demonstrates the value of model abstraction by developing abstract versions of DOD component models. The resultant models were simulated within an overall simulation system/framework and compared to the original detailed baseline models that they replaced. The differences between the abstracted and original detailed simulation models were demonstrated via defense and/or commercial simulation tools.

# 3 Tasks Accomplished

In order to achieve the research goals of the BAA effort, the following tasks were addressed.

- **Task 1: Identify target JMASS models**
  - Review existing JMASS-compliant models
  - Work with AFRL to identify additional models

- **Task 2: Determine the key model parameters that require abstraction**
  - Heuristics
  - Sensitivity Analysis

- **Task 3: Develop Abstract Models**:

- **Task 4: Simulate both the high fidelity and abstracted JMASS models**

- **Task 5: Demonstration**

- **Task 6: Reporting**

These tasks are detailed in the following sub-sections. The task description is presented along with the accomplishments, research performed, problems encountered, and subsequent direction of that task.

## 3.1 Task 1: Identify Target JMASS Models.

This effort was initiated by identifying a target modeling set on which to demonstrate the usefulness of model abstraction. This task examined various target modeling sets to determine if their application to this effort was feasible. This task reviewed the use of JMASS-compliant models from various JMASS Modeling Repositories for use in this effort. These models included radar, missile and ISR models that have been used to develop engagement-level and engineering-level system models. The model sets examined included the JTEST Model Suite, The CoreSim Model Suite, The JMASS98 Training Examples, and a set of Generic Helicopter Models. Many of these model suites were obtained through other JMASS-related efforts that RAM Laboratories has been involved in. Each of these Model Suites is unclassified.

### 3.1.1 JMASS98 Training Examples

The first model set examined was the JMASS98 Training Examples. The JMASS98 Training Examples are provided with the software release of JMASS98 to instruct users on the operation of JMASS98. For the purpose of this project,

the JMASS98 Training Examples were viewed as not containing enough detail to suitably demonstrate the effects of model abstraction.

### 3.1.2 CoreSim

The second model set examined was the suite of CoreSim simulation models. The CoreSim simulation suite supports the modeling of engagement scenarios using JMASS.

### 3.1.3 JTEST

The third model set examined was the suite of JTEST simulation models. The JTEST simulation suite supports the modeling of Air-to-Air and Surface-to-Air engagement scenarios using JMASS. JTEST includes RF environment, Airborne Target, Airborne Interceptor, Surface Platform, and Missile models with both geometric and RF based seekers.

### 3.1.4 Generic Helicopter Models

A fourth model set included a set of generic helicopter models used for Airborne Targets and Airborne Interceptor Platforms.

### 3.1.5 Model Suite Selection

The JTEST Model Suite was selected as the Model Suite of choice by this project. The JTEST Model Suite was selected for two reasons:

- Due to their use in other projects, the staff at RAM Laboratories had experience working with the JTEST Model Suite
- The JTEST Model Suite was accompanied by "canned" scenarios. These canned scenarios and calibration data could be used as a simulation baseline against which the abstracted simulation scenarios could be compared.

## 3.2 Task 2: Determine the key model parameters that require abstraction.

The second key task for this effort was to determine the candidate parameters for abstraction. This task involved examining the target models and identifying the key parameters that affected performance of the modeled component in the system. Dropping unneeded information allows the model to spend its simulation time on criteria that is deemed important to the system's operation. However, dropping any information may compromise a model's accuracy. "Key" information can be determined using a variety of methods. These methods may range from using a modeler's or analysts' "rule-of-thumb" (or heuristics) to deterministically determining key parameters using a sensitivity analysis. The second task involved determining these "key" parameters for a given simulation model with respect to a baseline simulation scenario.

### 3.2.1 Baseline Scenario

The baseline scenario used for this example was an air-to-air engagement scenario comprised of an Airborne Target, an Air-to-Air missile with a geometric seeker, an

Airborne Interceptor, and an RF environment. For a baseline scenario, the JTEST Model Suite provided a "canned" simulation, along with calibration data, and results. For the purpose of this "canned" simulation, the MOE (Measure of Effectiveness) was determined to be Point of Closest Approach (PCA) between the Air-to-Air Missile and the Airborne Target. For the baseline scenario, the PCA was identified as being 1.211 meters. The simulation of this scenario using the baseline simulation configuration and calibration data is shown in Figure 2.

```
..  Team: AAMSuite
.. Study: AAMScenario1
..  Date: Tue Jan 25 13:16:12 2000
.Class JTESTAAMPlayer
 UpdateInterval = 0.01 seconds
.Component_Instance Platform
                        BurnAccel = 130
 meters/second/second                BurnTime =
 5 seconds
           DeccelAfterBurnout = 20
 meters/second/second          GuideDelay = 1
 seconds

           LaunchRailPathName =
 "AI/LaunchRail[0]"          MaxFlightTime = 20
 seconds
 MinMach = 0.6
 RefLatitude = 0 radians
 RefLongitude = 0 radians
.Component_Instance NEDReference
.End NEDReference
.End Platform
.Component_Instance Umbilical
.End Umbilical
.Component_Instance Seeker

 TargetPathName = "Tgt/Platform"
 TrackRange = 5000 meters
.End Seeker
.End JTESTAAMPlayer
```

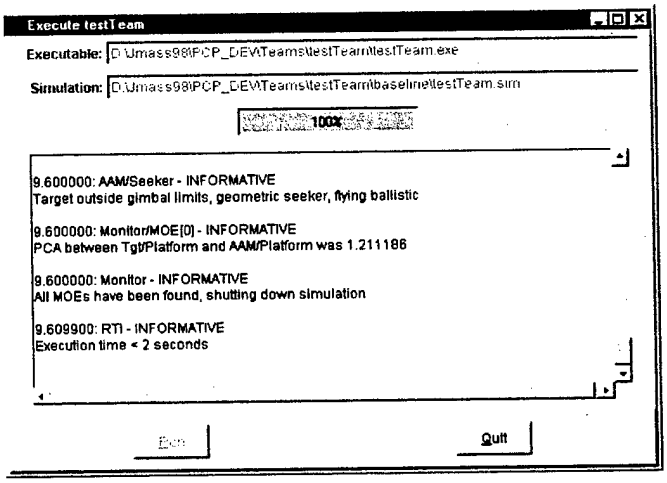



**Figure 2: Baseline Simulation Execution**

### 3.2.2 Illustration of Sensitive Parameters.

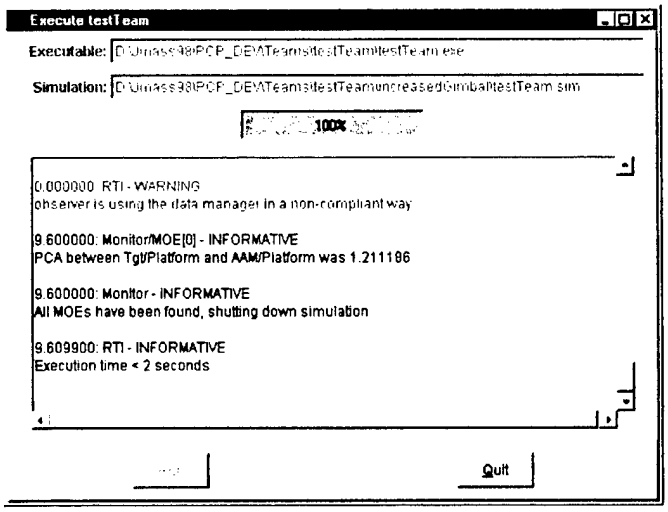
In order to illustrate the process of identifying "sensitive" simulation parameters with respect to the simulation baseline, the following example is presented. In Figure 2, the Air-to-Air Missile is examined with respect to two of its sub-components: the AAM platform and the AAM seeker. Two parameters, one from each component, are highlighted. These are the GuideFactor and the MaxGimbleAngle. For the purpose of demonstrating sensitivity, each of these parameters was altered with respect to the simulation baseline. For the simulation baseline, the GuideFactor was calibrated at 0.3, and the MaxGimbleAngle was calibrated at 0.8 radians. For the first example of parameter sensitivity, as shown in Figure 3, the MaxGimbleAngle parameter was increased to 3.15 radians. This value, however, did not affect the simulation result in that the PCA was the same for the altered simulation as the baseline version (1.211 m).

In the next example, the second parameter, GuideFactor was examined. For this scenario, the GuideFactor was increased to 1.0 from 0.3. The simulated scenario, as depicted in Figure 4, shows that the PCA for the new scenario had changed to 0.3392 m. As such, the simulation was sensitive with respect to this particular parameter

```
..  Team: AAMSuite
.. Study: AAMScenario1
..  Date: Tue Jan 25 13:16:12 2000
.Class JTESTAAMPlayer
 UpdateInterval = 0.01 seconds
.Component_Instance Platform
 BurnAccel = 130 meters/second/second
           BurnTime = 5 seconds
           DeccelAfterBurnout = 20
 meters/second/second        GuideDelay = 1
 seconds

           LaunchRailPathName =
 "AI/LaunchRail[0]"        MaxFlightTime = 20
 seconds
 MinMach = 0.6
 RefLatitude = 0 radians
 RefLongitude = 0 radians
.Component_Instance NEDReference
.End NEDReference
.End Platform
.Component_Instance Umbilical
.End Umbilical
.Component_Instance Seeker

 TargetPathName = "Tgt/Platform"
 TrackRange = 5000 meters
.End Seeker
.End JTESTAAMPlayer
```




(GuideFactor).

**Figure 3: Changing MaxGimbalAngle in Simulation Execution**

```
..  Team: AAMSuite
.. Study: AAMScenario1
..  Date: Tue Jan 25 13:16:12 2000
.Class JTESTAAMPlayer
 UpdateInterval = 0.01 seconds
.Component_Instance Platform
 BurnAccel = 130 meters/second/second
           BurnTime = 5 seconds
           DeccelAfterBurnout = 20
 meters/second/second        GuideDelay = 1
 seconds

           LaunchRailPathName =
 "AI/LaunchRail[0]"        MaxFlightTime = 20
 seconds
 MinMach = 0.6
 RefLatitude = 0 radians
 RefLongitude = 0 radians
.Component_Instance NEDReference
.End NEDReference
.End Platform
.Component_Instance Umbilical
.End Umbilical
.Component_Instance Seeker

 TargetPathName = "Tgt/Platform"
 TrackRange = 5000 meters
.End Seeker
.End JTESTAAMPlayer
```

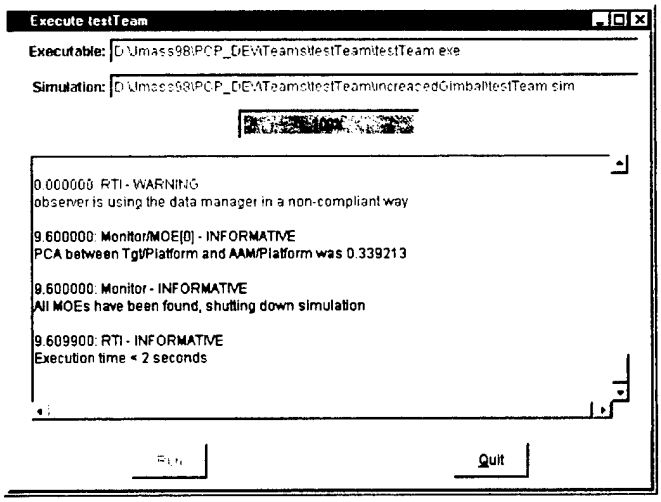



**Figure 4: Changing GuideFactor in Simulation Execution**

*Note: It must be noted that, for the purpose of this project, the goal of the simulation is not to hit the target with the abstract model, but rather to come as close as possible to estimating the PCA found in the baseline simulation. Thus, the abstract simulation must*

*use the baseline simulation as its "target" (i.e. the simulated view of reality) rather than trying to hit the airborne target of the baseline scenario.*

### 3.2.3 Sensitivity Analysis

Sensitivity analysis is the process of identifying a system's inputs or parameters that have the greatest effects on a model's output information. For each component model and selected outputs, a sensitivity function can be defined that covers both the model input variables and the internal model state (if applicable). This sensitivity function is a result of statistical analyses that identify the variables (input or state) that have the greatest effect on a given output value. Sensitivity analysis is generally deterministically or statistically performed over the range of values for the system's state or inputs. The parameters that typically result in the greatest disparity of output values are deemed to have the greatest effect on system performance. Parameters that are determined to have little effect on system performance may be abstracted away. Methods of employing sensitivity analysis include full factorial or fractional factorial analysis methods. These methods, however, consider parameter sensitivity over all possible input values and parameters. Given that the number of possible inputs, even with respect to the baseline scenario, is infinite in number over an infinite range (due to parameters such as altitude and initial velocity), some method of pruning the solution must be employed. One such method is the use of heuristics as a pre-processing step.

#### 3.2.3.1 Heuristics

This project examined the use of heuristics as a pre-processing step. Heuristics involved the use of *a priori* knowledge about a system or model to identify that system/model's key parameters. Heuristics can be used to reduce the computational complexity of an experiment. For example, for a full factorial sensitivity analysis, all parameters may be varied over the entire range of possible values. For this scenario, this may mean that a given parameter may be required to be varied over an infinite range of values (for example, varying an initial altitude parameter from 0 to infinity) or varying parameters with an undefined degree of precision (for example, specifying the precision of the altitude parameter down to the centimeter, millimeter, etc). For a fractional factorial analysis, this would also apply (after, a fraction of infinity can still be infinity). Thus, for the purpose of reducing the computation required for the sensitivity analysis of this project, heuristics were used to limit the range of input and parameter values to within +-20% of the baseline simulation scenario input and parameter values. In addition, heuristics were also applied to limit the range of possible values to +-20%, +-15%, +-10%, and +-5% of the baseline parameter values. Thus, the levels of possible parameter values were also clearly defined. For example, for MaxGimbalAngle (baseline value of 0.8 radians) the range of values consisted of {0.96, 0.92, 0.88, 0.84, 0.8, 0.76, 0.72, 0.68, 0.64}. Once all possible input and parameter values were defined for this experiment, the sensitivity analysis was then employed with respect to the simulation output value, PCA.

### 3.2.3.2 Sensitivity Analysis

Once the range of values was determined for each parameter, the sensitivity analysis was performed. This process used for applying this sensitivity analysis is described in section 3.2.3.2.1.

#### 3.2.3.2.1 Sensitivity Analysis Process

For the purpose of this project, only the Air-to-Air missile was examined. The parameter values for this component model included the following:

- BurnAccel
- BurnTime
- DecelAfterBurnout
- GuideDelay
- GuideFactor
- MaxFlightTime
- MinMach
- RefLatitude
- RefLongitude
- MaxGimbalAngle
- TrackRange

Each of the simulation parameters was varied over the range of +-20%, +-15%, +-10%, and +-5%. The output simulation value, PCA, was then compared to the baseline value. The differences in output value were then statistically analyzed by identifying their variance and standard deviation with respect to the baseline simulation scenario. The sensitivity analysis process was inserted into the JMASS98 Simulation Development flow using JMASS98's FAST Tool as shown in Figure 5.

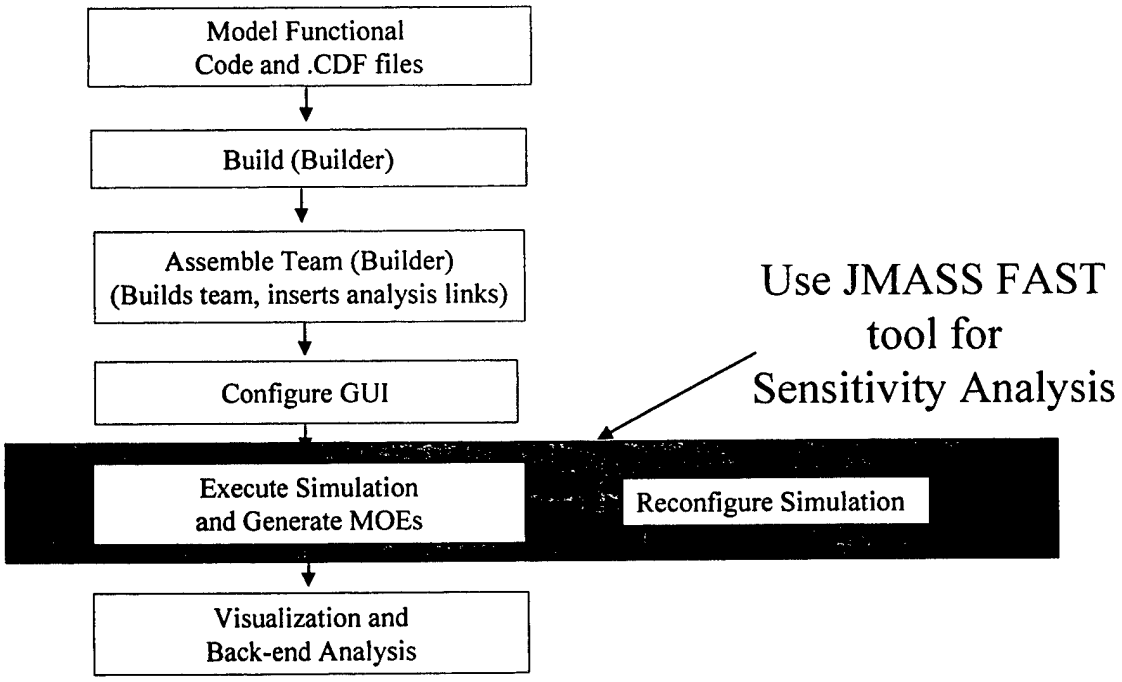


**Figure 5: JMASS98 Simulation Development Flow**

The JMASS FAST tool was used to implement the sensitivity analysis process as shown in Figure 6.

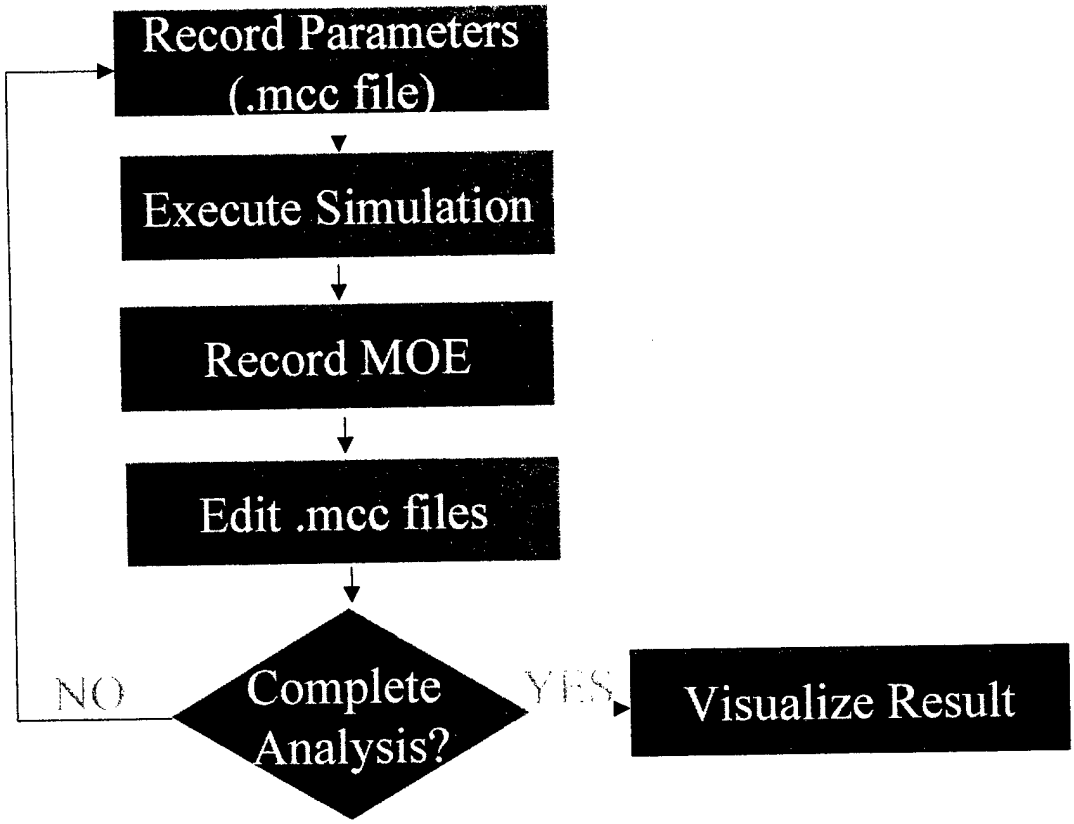



**Figure 6: Process for Performing Sensitivity Analysis**

The sensitivity analysis was performed for all parameters varied independently (1st order) as well as varying all parameters in conjunction with one other parameter simultaneously (2nd order). The analysis was conducted on a Dell Dimension 4100 PC (850 Mhz) running Win2K.

#### 3.2.3.2.2 Sensitivity Analysis Goals

The goals of the sensitivity analysis process were as follows:

- Identify all independent, highly sensitive parameters with respect to baseline data
- Identify all second order highly sensitive parameters
- Record all simulation time
- Assess analysis/simulation time tradeoffs

#### 3.2.3.2.3 Sensitivity Analysis Results

Sensitivity analysis was performed to identify all first order and second order sensitive parameters. For the purpose of analyzing data, the variance of all PCA values with respect to the simulation baseline was summed for each experiment.

##### 3.2.3.2.3.1 First Order Sensitivity Analysis

The results for the first order experiment are shown in Figure 7. As shown in this figure, the TrackRange and MaxGimbalAngle parameters showed a much higher sensitivity than the other parameters (so much so that, by comparison, the effects of the other parameters were negligible).

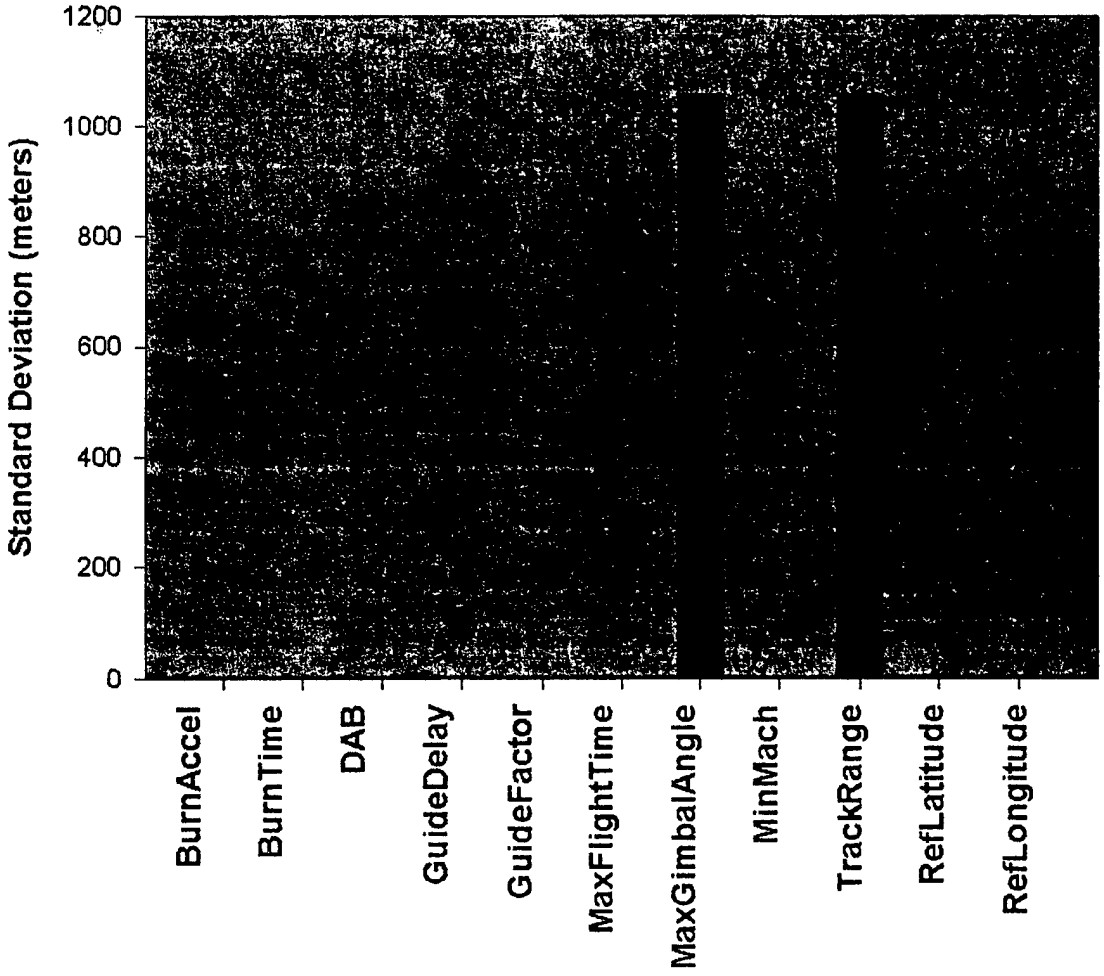



**Figure 7: First Order Sensitivity Analysis**

In order to further examine the effect of the other parameters, the TrackRange and MaxGimbleAngle standard deviation values were removed from the graph. The effects

for the remainder of the parameters from the first order experiment are shown in Figure 8. As shown, the MaxFlightTime, MinMach, RefLatitude, and RefLongitude values all had no effect on the baseline scenario with respect to PCA.

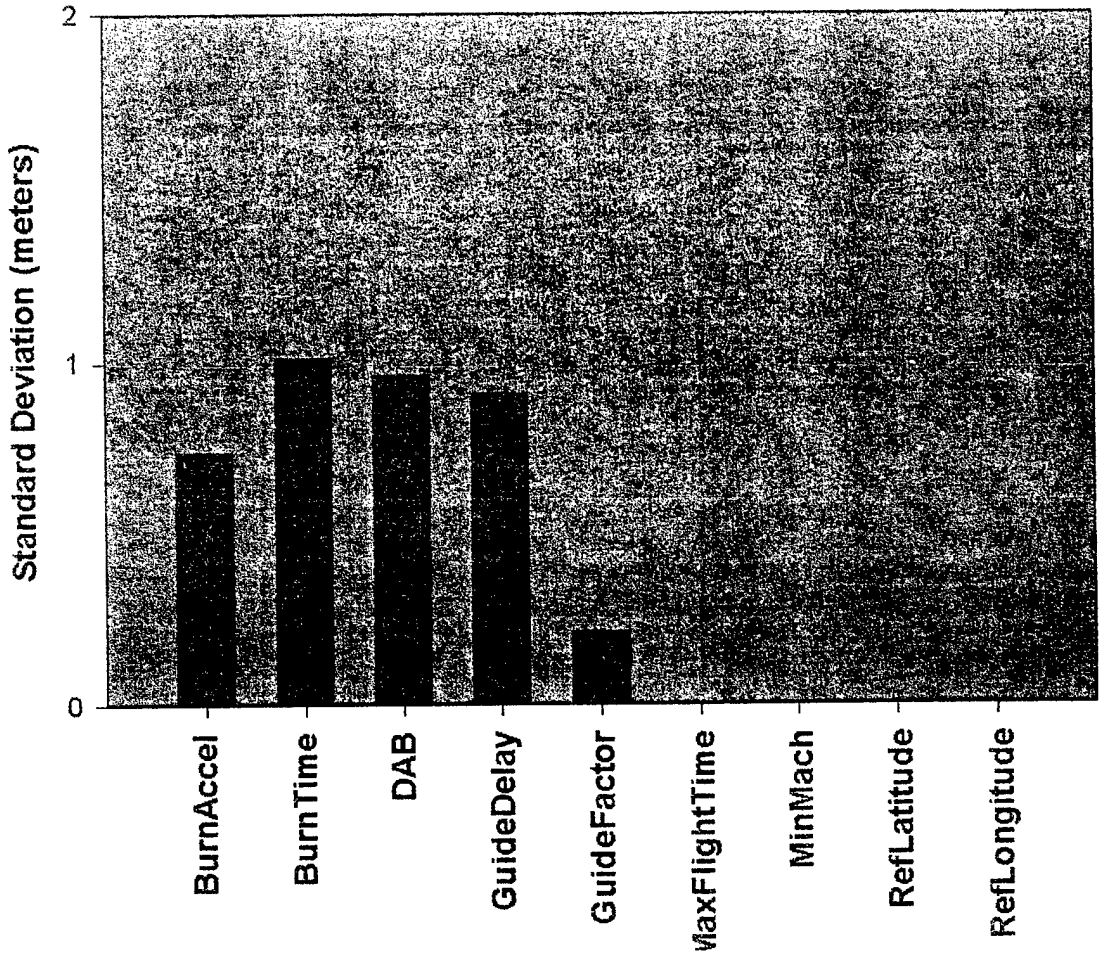



**Figure 8: First Order Sensitivity Analysis - Less Sensitive Parameters**

### 3.2.3.2.3.2 Second Order Sensitivity Analysis

The results for the second order experiment are shown in Figure 9. As shown in this figure, the experiments involving all combinations that varied, the TrackRange and MaxGimbalAngle parameters showed a much higher sensitivity than the other parameters (so much so that, by comparison, the effects of the other parameters were negligible).

In order to further examine the effect of the other parameters, the combinations that varied the TrackRange and MaxGimbleAngle values were removed from the graph. The effects on the remainder of the parameters from the first order experiment are shown in

Figure 10. As shown, the MaxFlightTime, MinMach, RefLattitude, and RefLongitude values all had no effect on the baseline scenario with respect to PCA.

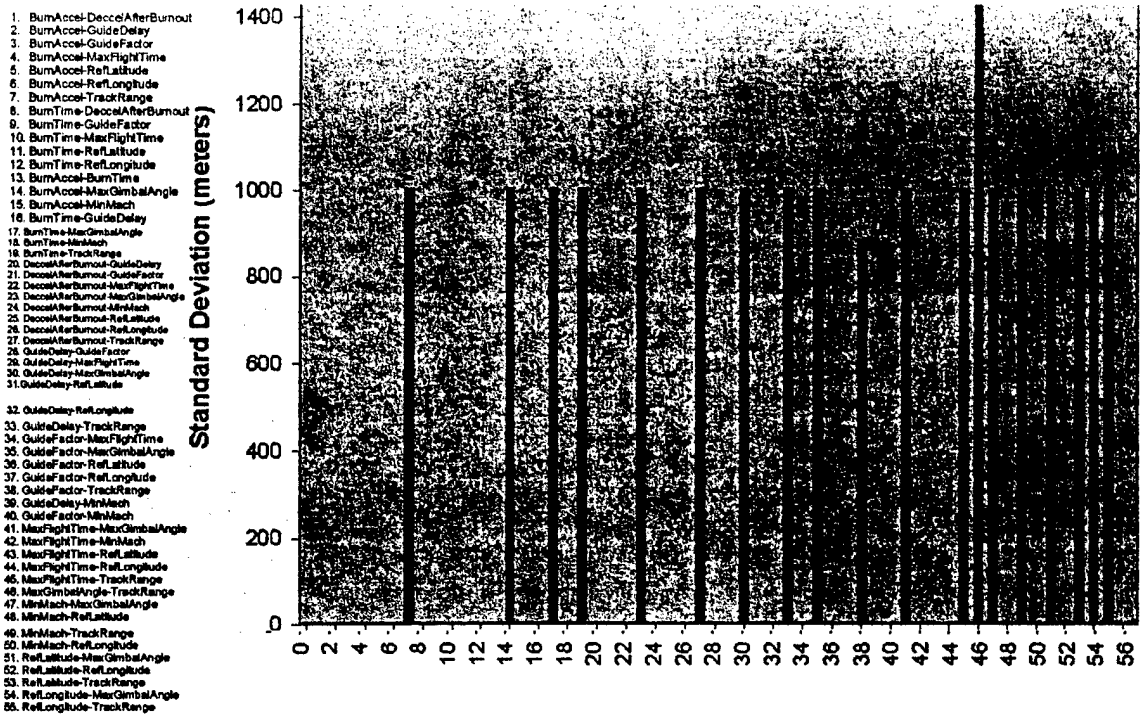



**Figure 9: Second Order Sensitivity Analysis**

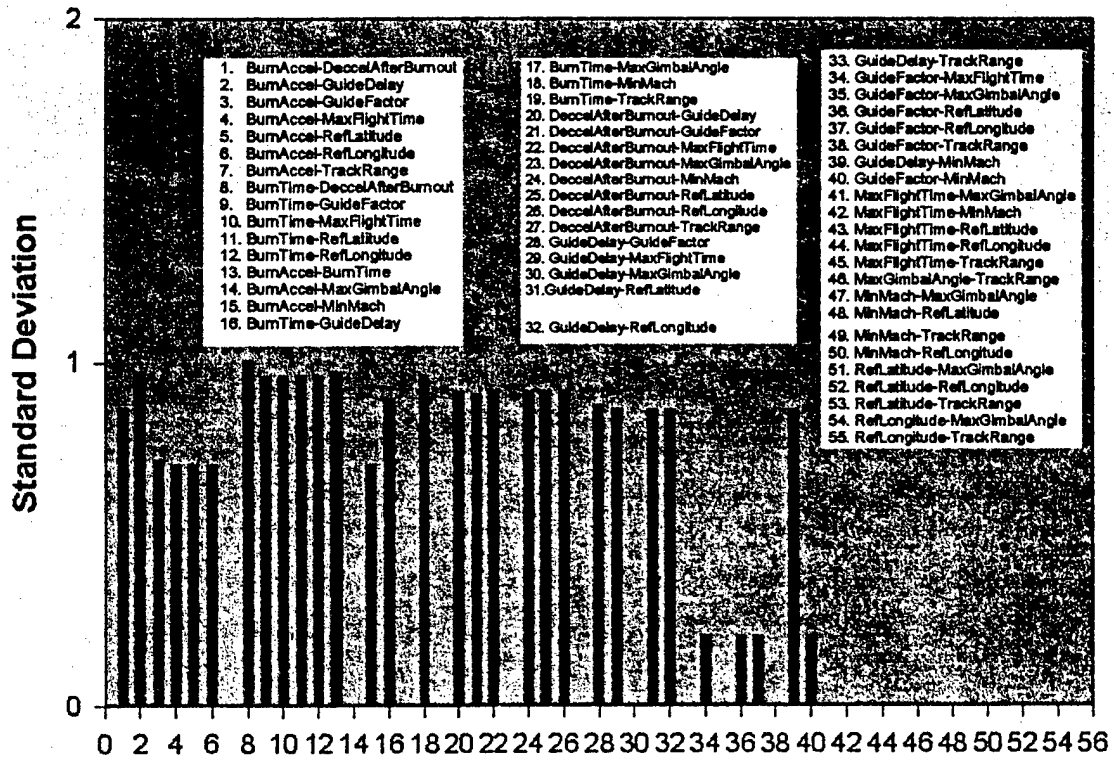



**Figure 10: Second Order Sensitivity Analysis - Less Sensitive Parameters**

### 3.2.4 Sensitivity Analysis Lessons Learned

The following lessons were learned with respect to the sensitivity analysis process.

- We cannot perform a sensitivity analysis over the entire simulation space
  - Explosively complex
- Must use some heuristics
  - Keep simulation space within the context of simulation
    - Use baseline scenarios as a guide
  - Must quantize parameters' values within some range
    - Keep simulation space manageable

## 3.3 Task 3 and Task 4 Developing and Simulating Abstract Models

Once some of the key modeling parameters were identified, abstract versions of the models were constructed. Several different abstract modeling techniques were examined for use in developing these abstract representations. Some of these methods are described in Section 3.3.1.

### 3.3.1 Abstract Modeling Methods

Simple model abstraction techniques can take several forms: statistical distributions, omission, aggregation, look-up tables, and approximations. These abstractions can then be fed into the higher fidelity model to resolve disparate models. These techniques are typically used to develop hard-coded wrappers that are employed for specific simulations. Transferring these wrapped models to other simulations requires a significant amount of new coding. Some techniques are illustrated in Figure 11.

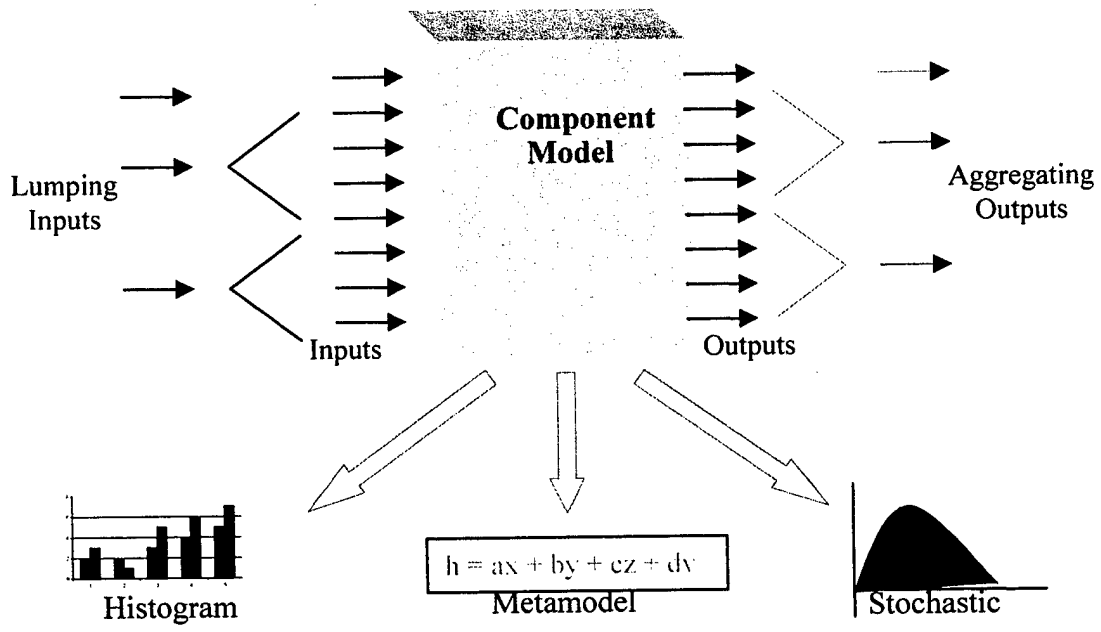



**Figure 11: Abstract Modeling Methods**

#### 3.3.1.1 Intuitive Methods

Many developers advocate using general heuristics or subjective methods for abstracting parameters, in addition to using heuristics as a sensitivity analysis step. An example can be found in the case where some subset of parameters could be marked as unimportant and not used in future abstraction functions. However, in general, these methods are difficult to formalize, highly subjective, and require specialized knowledge in the domain being modeled.

#### 3.3.1.2 Approximation

One technique for abstracting a set of parameters is to simply approximate some or all of those parameters. This can be done by rounding off values or by breaking up continuous values into discrete chunks.

#### 3.3.1.3 Equations

Mathematical equations can be substituted for running an actual simulation to generate parameters. If the behavior of one or more parameters in the simulation over time is understood, then the partial derivative of that parameter may be used. While an effective method, this does require a fairly complete understanding of the internals of the model and this is something that the current work strives to avoid.

#### 3.3.1.4 Omission

This technique removes or replaces one or more parameters. These parameters can be either removed entirely or else replaced by constant values or values that give no information to the other components of the model. For instance, it might make sense to replace a parameter with its mean or modal value in a simulation if it is determined that the value does not significantly influence the behavior of the simulation (or, rather, the behavior of the subset of parameters of the simulation that a developer is interested in).

#### 3.3.1.5 Aggregation

This technique involves grouping similar components and aggregating their inputs together. This aggregation can take the form of a weighted average of the component parameters or some other compounding function can be used.

#### 3.3.1.6 Statistical Distributions

This abstraction technique refers to replacing parameter values with random variables based on a probability distribution that describes the range of values for the parameters in question. It seems to be the most prolific of the data abstraction techniques used for MRM. Perhaps this is because it does not require a detailed understanding of the model functionality. It does, however, require a data set (i.e.: recorded parameter values for simulation runs) of known values. This need be nothing more than a table of values.

#### 3.3.1.7 Clustering Algorithms

Clustering algorithms are useful whenever one needs to classify an excessive amount of information into a set of manageable and meaningful smaller amounts. There

are several techniques to perform such a classification. Clustering algorithms, and modeling and simulation in general, can be thought of as dealing with vectors in state space. These vectors can be defined as the application of input parameters with respect to time. For example, given a model component with N parameters, we can think of an N-dimensional vector moving through N-dimensional state space over the time of the simulation. Clustering algorithms divide up that state space into different regions where the vectors cluster with respect to output values or measures of effectiveness.

### 3.3.1.8 Tree Clustering

Tree Clustering refers to the class of clustering algorithms that join together elements successively based on some measure of similarity. It is easiest to think of this process as resulting in a hierarchical tree, with the most detailed level corresponding to the roots of the tree (where each element is in its own cluster), and the most abstract level corresponding to a single node at the top of the tree (where each element is in the same cluster). The measure of similarity is often simply the distance among vectors whose dimensionality corresponds to the number of traits per element being considered. The easiest and most general measure of distance is Euclidean Distance:

$$distance(x,y) = \sqrt{\square\square(x_i-y_i)^2}$$

A somewhat more versatile measure of distance is the Power Distance:

$$distance(x,y) = (\square(x_i-y_i)^p)^{1/r}$$

Where p equals the progressive weight that is placed on differences on individual dimensions and r represents the progressive weight that is placed on larger differences between elements. Note that if p=r=2, then this is the same measure as Euclidean Distance.

In order to join together individual clusters, a developer typically measures the distance of either the nearest or furthest elements in a set of clusters. Alternatively, he can measure the average or weighted average (weighted according to the size of each cluster) distance of all elements in a set of clusters.

### 3.3.1.9 K-means Clustering

K-means clustering is used when a developer already has certain hypotheses about how the vectors he is dealing with will cluster. It is therefore poorly suited for MRM work. The K-means clustering algorithm creates K clusters with maximum distance between them, where K is a user-supplied parameter. It starts with K random clusters and then progressively moves elements between clusters to maximize the variance between clusters.

#### 3.3.1.10 Vector Quantization

Vector Quantization has been typically used for data compression. Originally multiple variable integration, it can now be used in conjunction with training algorithms to facilitate use. Vector quantization generates prototypes that represent the centers of the clusters they belong to. Often an excess of prototypes are used to prevent algorithms from 'running out' of clusters. However, there is no mechanism within this technique for adding new prototypes. This method would therefore require knowledge about the domain being clustered in order to accurately predict the number of prototypes.

### 3.3.2 Development of Abstract Models Using Abstraction Techniques

Several of the abstraction methods described in the above subsection were examined for use in this project. Not all of these methods, however, were identified as being suitable for developing abstraction representations. For example, using histograms to replace model behavior was viewed as having too much overhead and would adversely affect simulation time. The use of look-up tables was likewise thrown out for having too much overhead. In addition, the use of certain clustering algorithms, such as the Distributed Clustering Algorithms was thrown out as being too complex for this experiment. The methods chosen included:

- Timing
- Omission
- Value-based
- Clustering
  - Quantization
  - Stochastic-based
- Stochastic-based

#### 3.3.2.1 Timing Abstraction

The first case of developing abstract models involved the use of timing abstraction. For this method of abstraction, the update_time interval was varied from 0.01 s to 0.5 s (0.01 s was the baseline value). This process caused the simulation models to update their values less often. Theoretically, fewer updates meant less iterations and, thus, fewer calculations. In addition, however, fewer updates and iteration would mean less accurate simulation results. This experiment was performed for 4 different engagement scenarios ranging from one-on-one to twelve-on-twelve. These scenarios, shown in Figures 12 to 15 depict the variance of PCA with respect to the update interval. In each of these scenarios, the PCA worsened as the update interval increased. It should be noted that for multiple engagements, the max difference depicts the largest difference in PCA while the average difference depicts the average difference in PCA across all engagements.

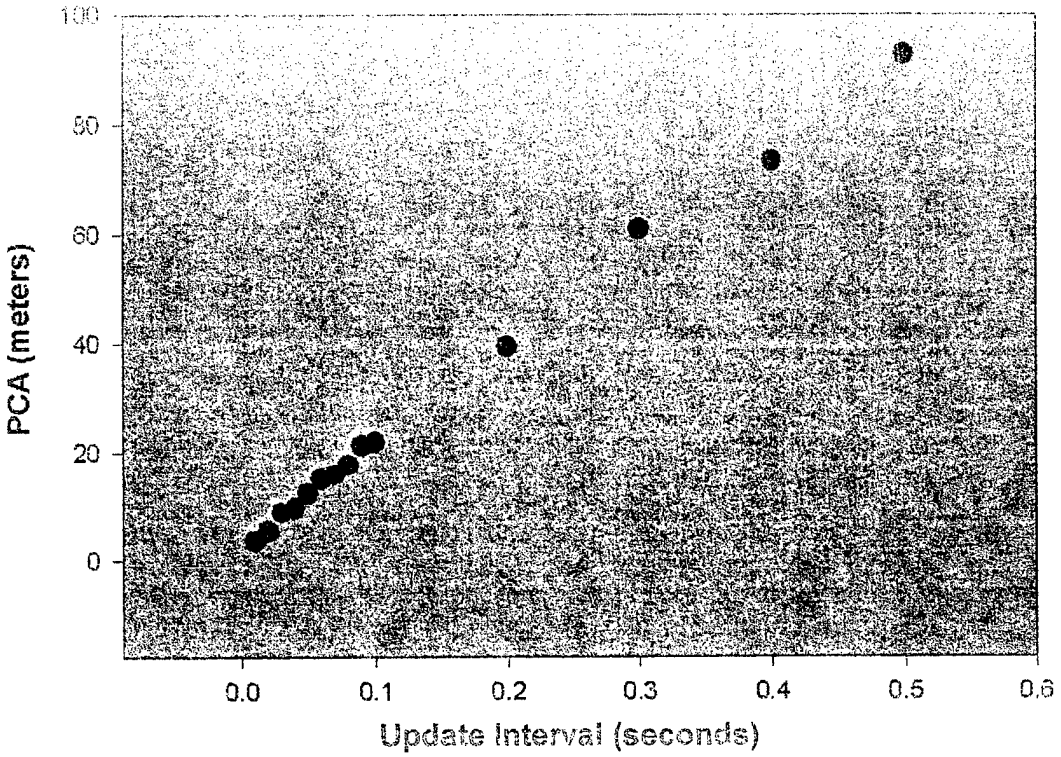



Figure 12: PCA vs. Update Interval – 1 on 1 Engagement Scenario

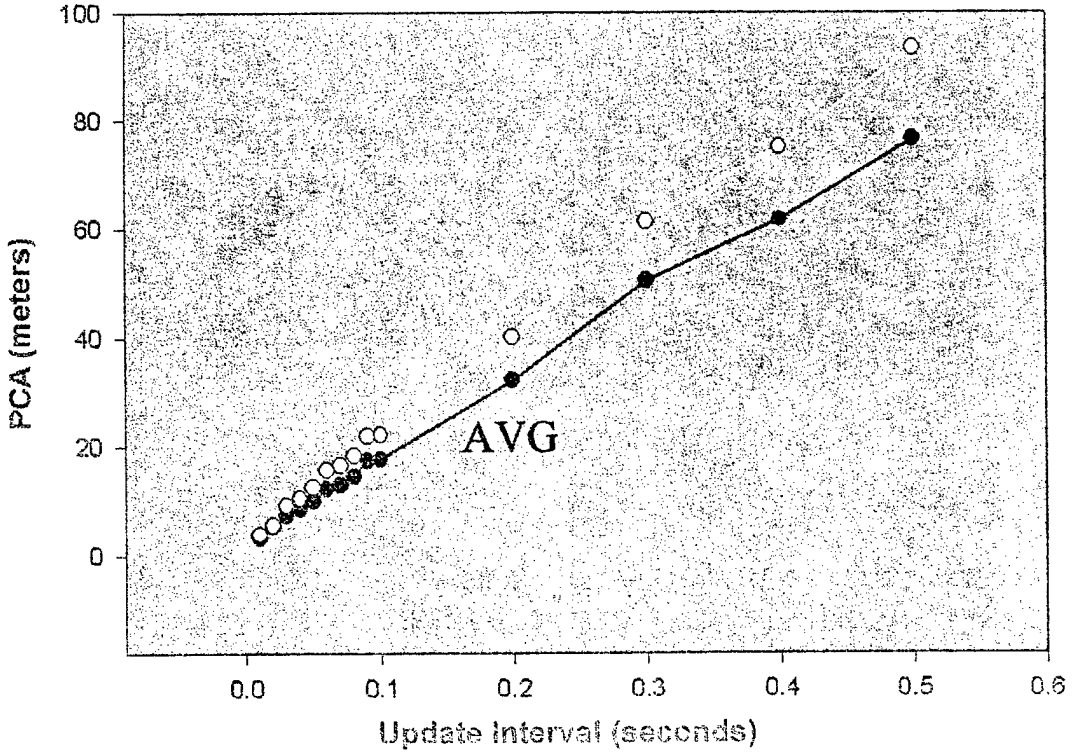



Figure 13: PCA vs. Update Interval – 4 on 4 Engagement Scenario

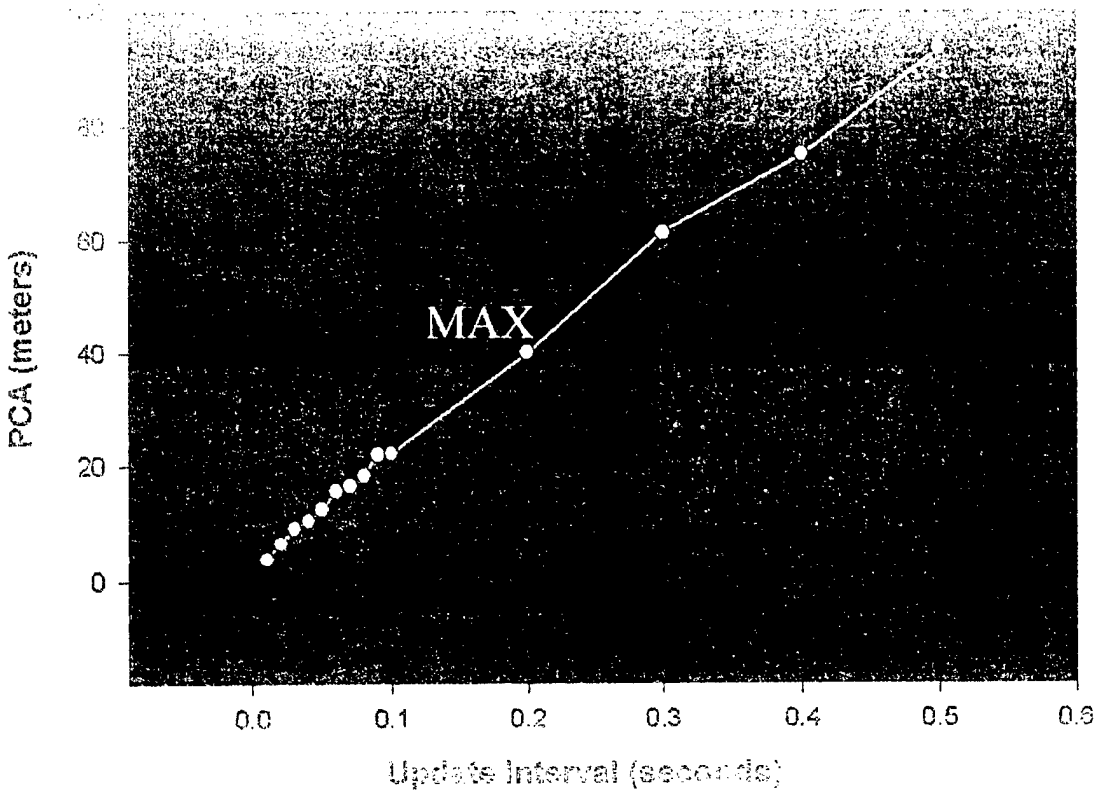



Figure 14: PCA vs. Update Interval - 8 on 8 Engagement Scenario

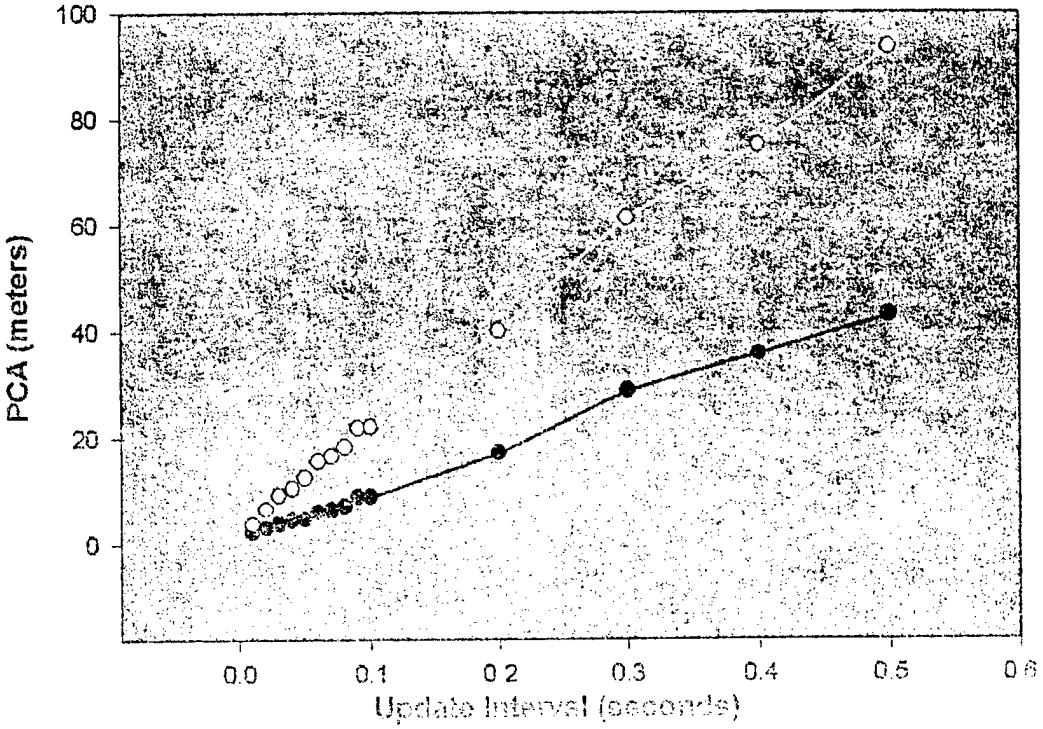



Figure 15: PCA vs. Update Interval - 8 on 8 Engagement Scenario

One of the theories in requiring fewer updates for each model in the engagement scenario is that fewer computations would require less simulation time. However, for this experiment, fewer updates showed basically no difference in simulation time. Figures 16 to 19 depict the Wall clock simulation time versus update time and Figures 20 to 23 depict the CPU simulation time versus update time for each group of engagement scenario simulations.

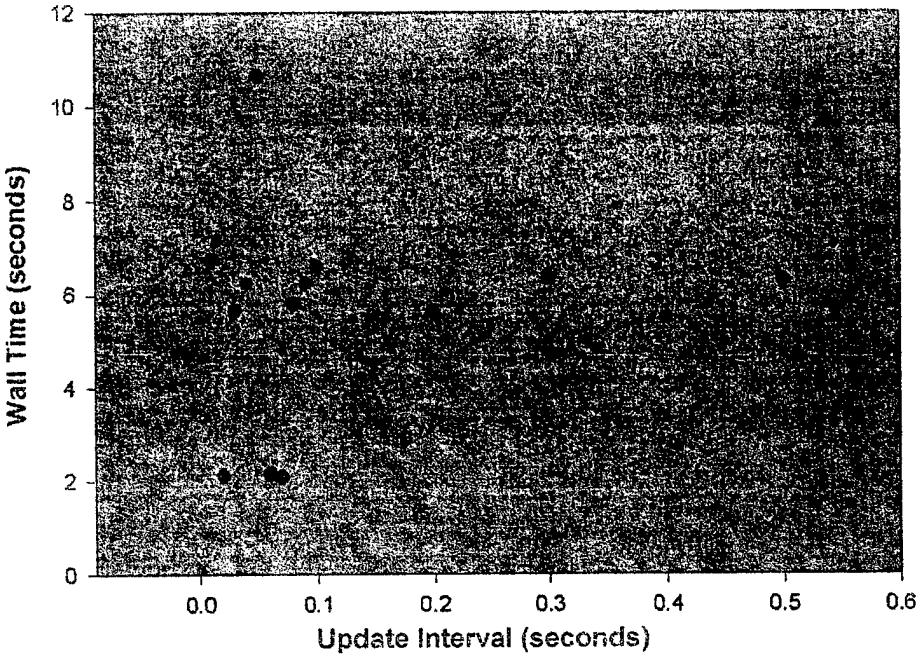



**Figure 16: Wall Clock Time vs. Update Interval (NT) – 1 on 1 Engagement Scenario**

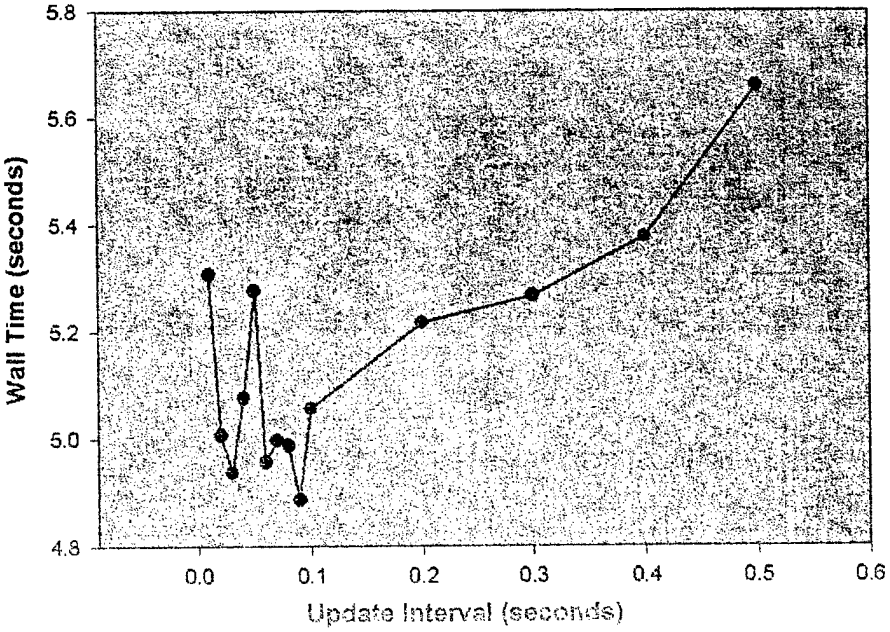



Figure 17: Wall Clock Time vs. Update Interval (NT) - 4 on 4 Engagment Scenario

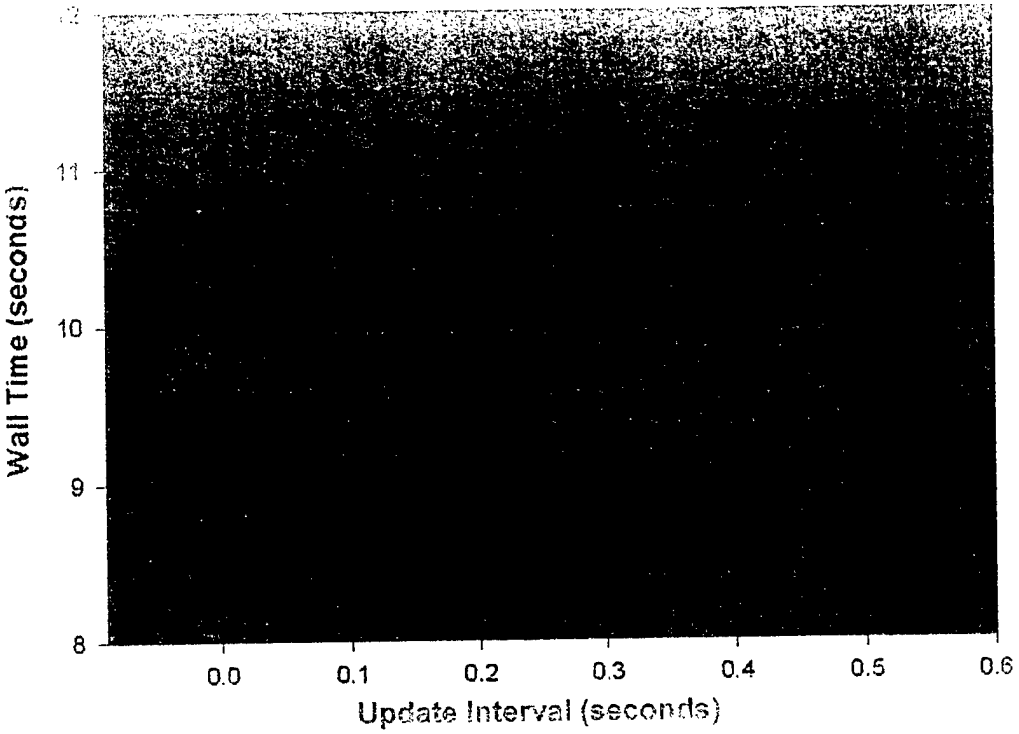



Figure 18: Wall Clock Time vs. Update Interval (NT) - 8 on 8 Engagement Scenario

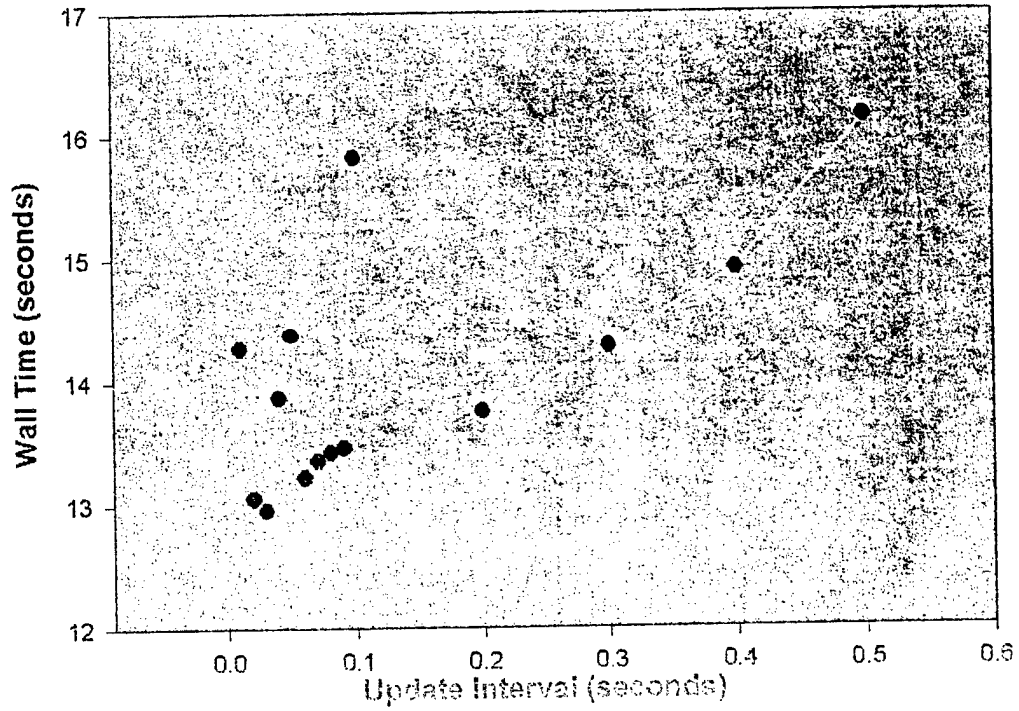



Figure 19: Wall Clock Time vs. Update Interval (NT) - 12 on 12 Engagement Scenario

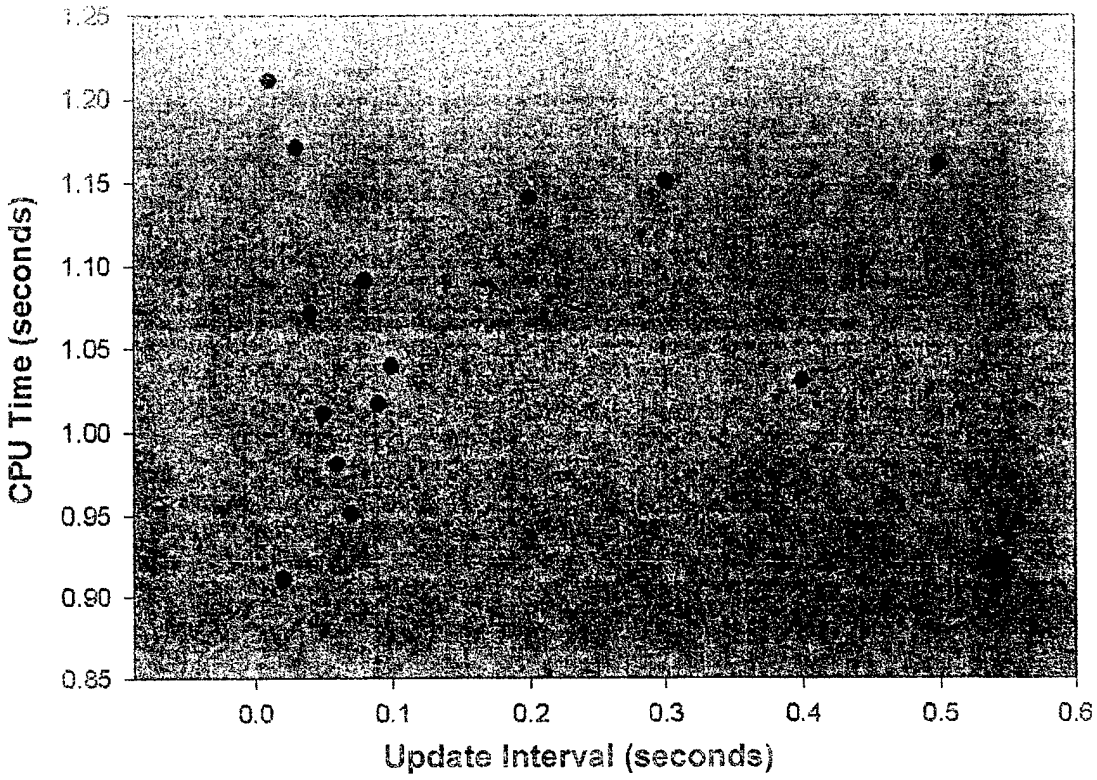



Figure 20: CPU Time vs. Update Interval (NT) - 1 on 1 Engagement Scenario

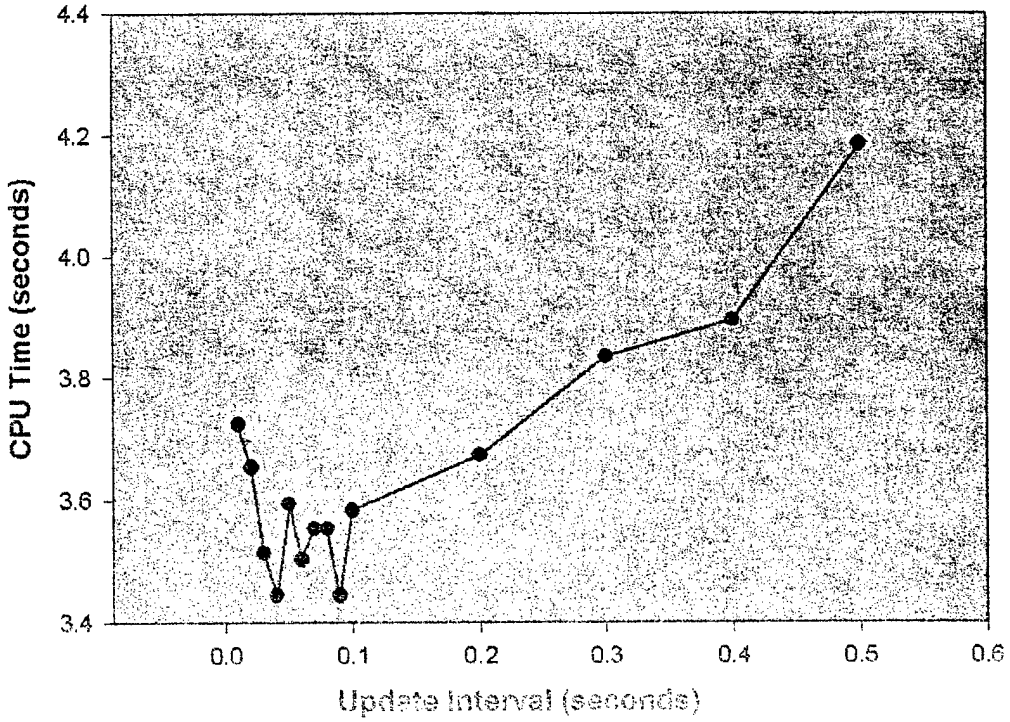



Figure 21: CPU Time vs. Update Interval (NT) - 4 on 4 Engagement Scenario

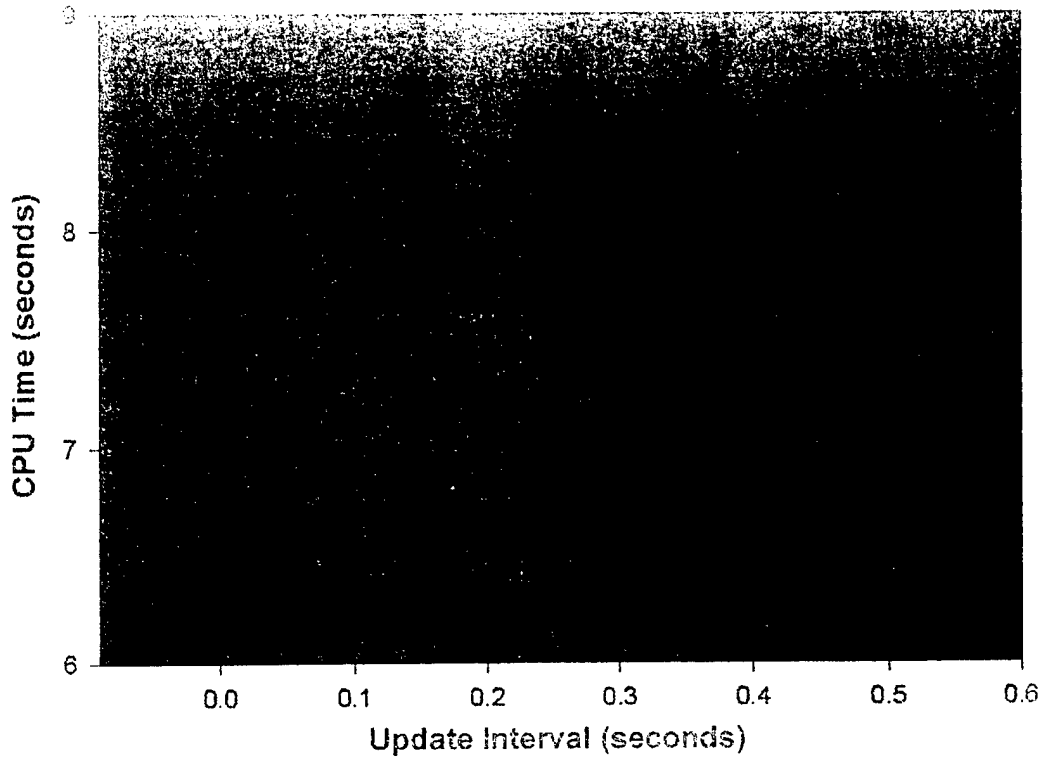



Figure 22: CPU Time vs. Update Interval (NT) - 8 on 8 Engagement Scenario

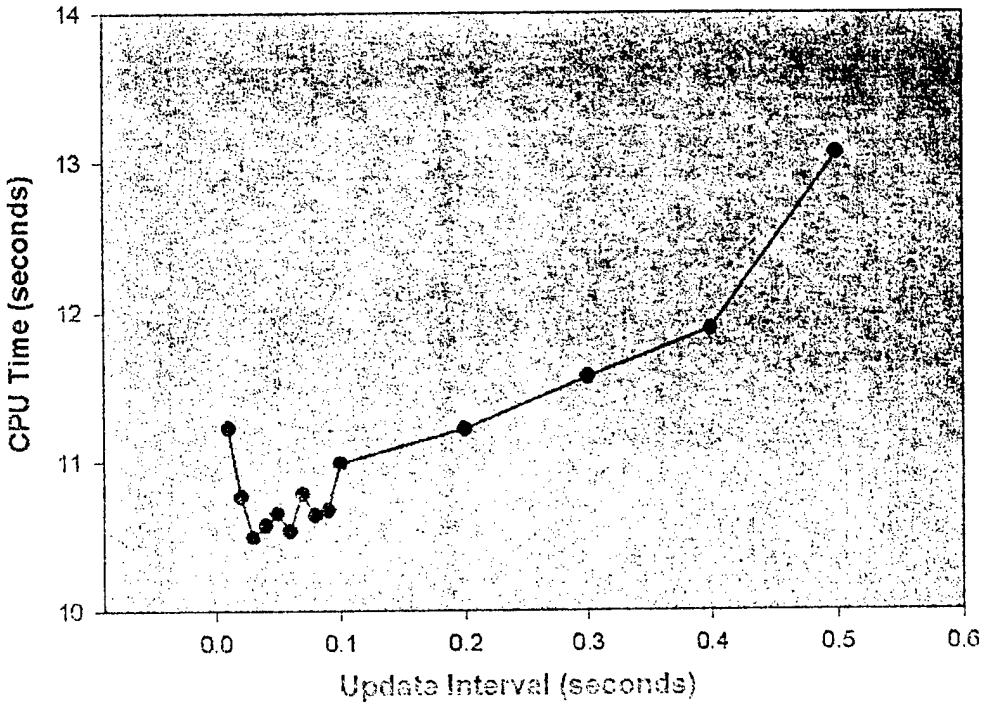



Figure 23: CPU Time vs. Update Interval (NT) - 12 on 12 Engagement Scenario

In order to ensure that the timing artifacts found from these experiments were not platform dependent, additional comparisons were made using SGI Indigo2 platforms. The Wall clock simulation time versus Update Interval comparison is shown in Figure 24 for a 12 on 12 engagement scenario. The CPU simulation time versus the update interval comparison is shown in Figure 25 for a 12 on 12 engagement scenario.

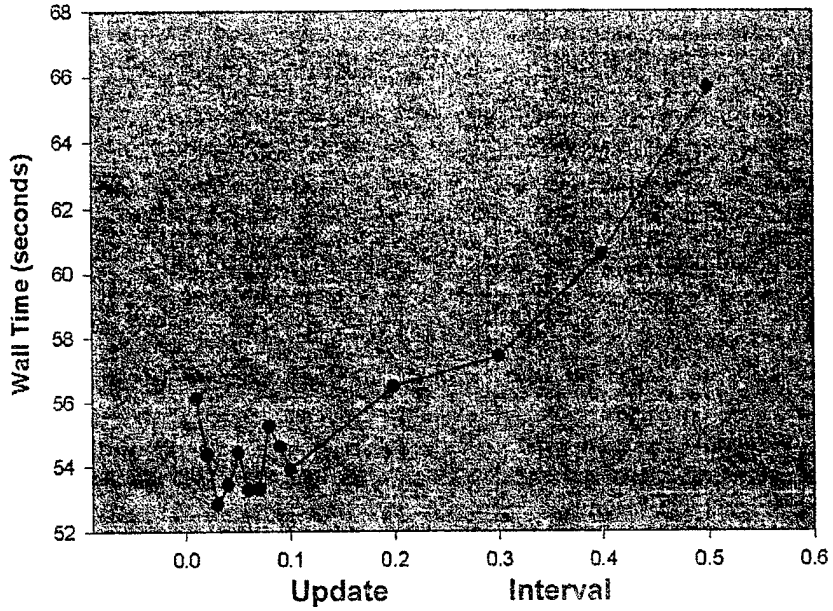



**Figure 24: Wall Clock Time vs. Update Interval (SGI) - 12 on 12 Engagement Scenario**

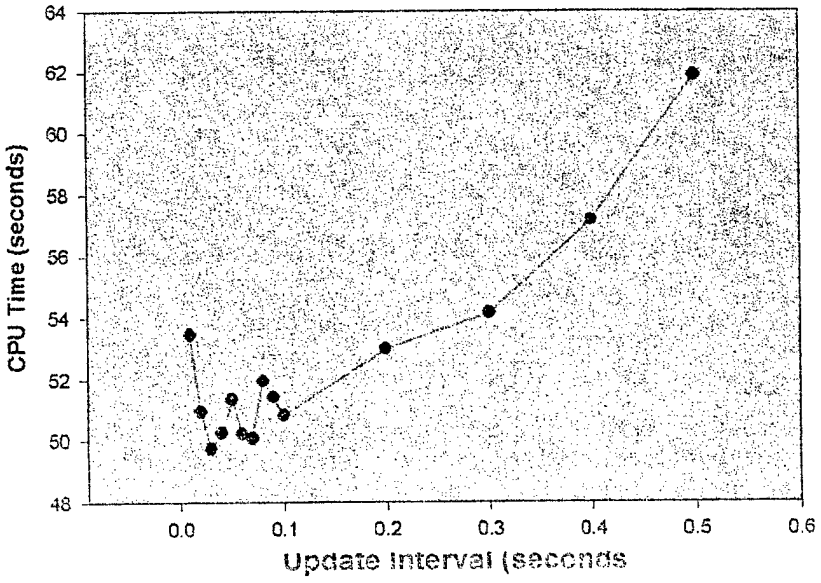



Figure 25: CPU Time vs. Update Interval (SGI) - 12 on 12 Engagement Scenario

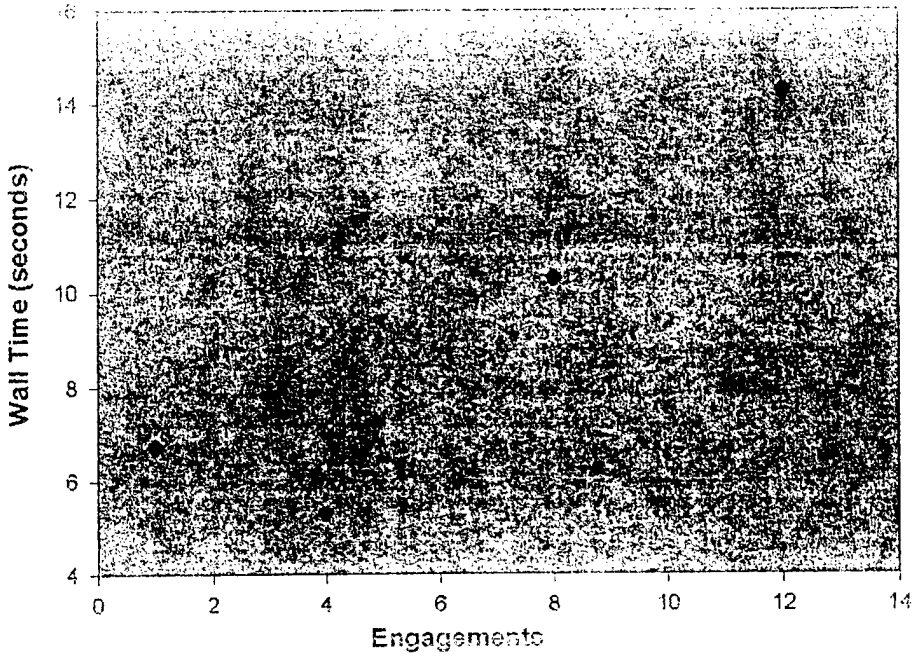



Figure 26: Wall Clock Time vs. Number of Engagements

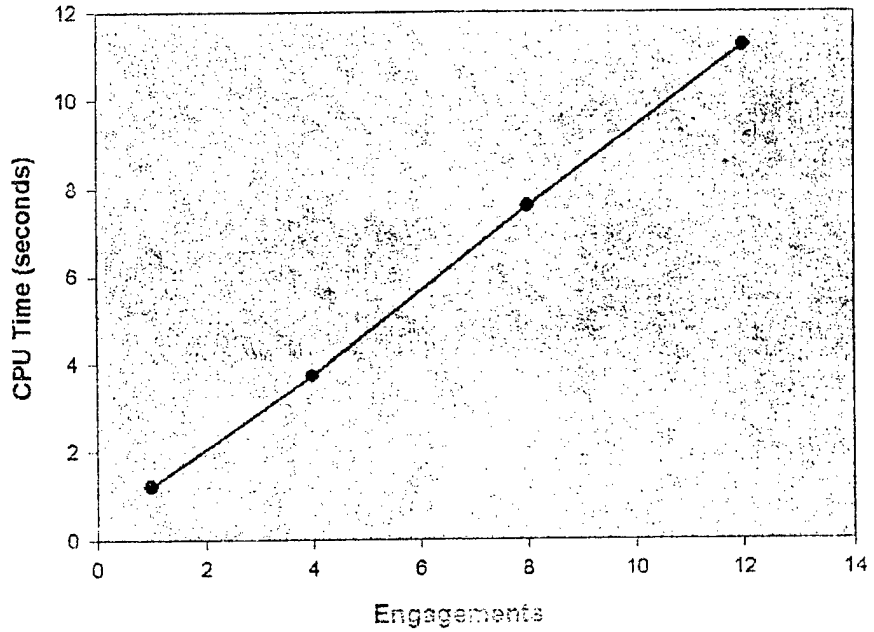



Figure 27: CPU Time vs. Number of Engagements

### 3.3.2.2 Omission

The second method of developing abstract models involved the use of omission abstraction techniques. Employing the omission abstraction methods focused on the calculation of the pitch and yaw values by the Missile Platform model. The pitch and yaw values are dependent upon the Bse_az and the Bse_el input parameters to the Missile's Platform sub-component as shown in Figure 28. The pitch and yaw values affect the missile's spatial orientation and spatial velocities. For this project, the omission technique was applied by simply omitting the calculation of the missile's spatial orientation from the model. The result of the omission technique was that the missile was not oriented properly (as later viewed using JMASS98's Simview visualization tool). However, the missile's orientation did not affect the PCA output of the simulation.

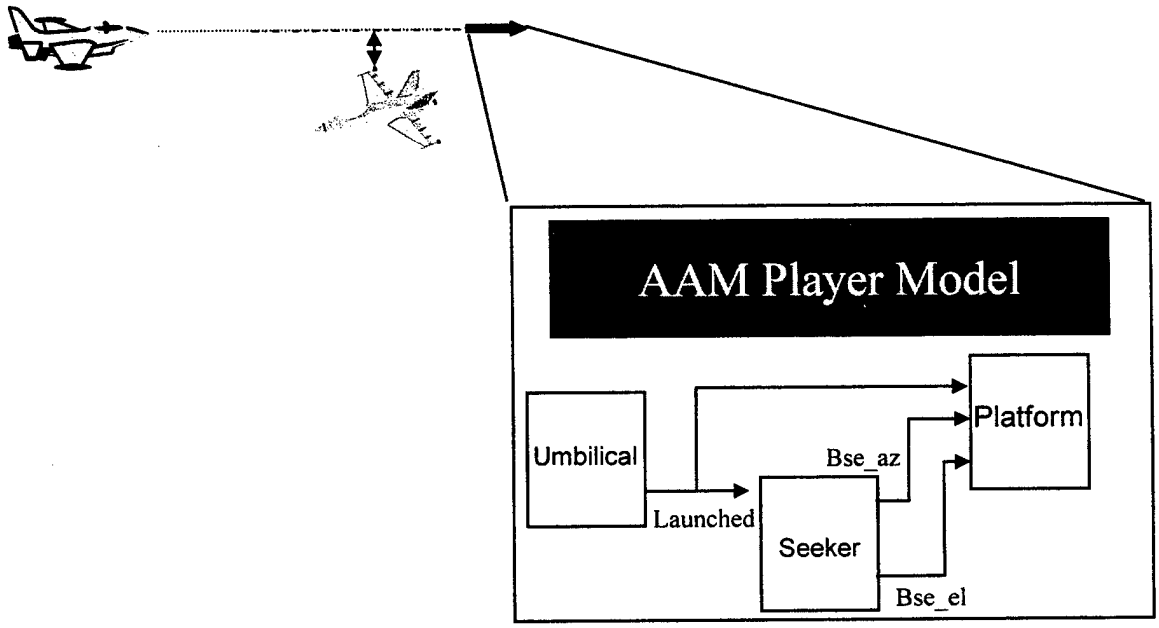



**Figure 28: Air-to-Air Missile Platform Sub-component Model**

### 3.3.2.3 Value-based Abstraction

The next method involved value-based abstraction. This experiment involved developing an abstraction of the actual pitch and yaw guidance calculations performed by the Missile's Platform Sub-component. Using value-based abstraction, the pitch and yaw guidance calculations could be replaced by an abstract representation for the purpose of determining the missile's spatial orientation and spatial velocities.

#### 3.3.2.3.1 Pre-processing step

The first step in developing a value-based abstraction for the missile's platform sub-component was to develop a representation of the pitch and yaw calculations within the context of the baseline simulation scenario. A preprocessing step was used to record the calculated pitch and yaw values for each iteration step of the sub-component model

during simulation of the baseline model. A graph of the pitch values versus time is shown in Figure 29. A graph of the yaw values versus time is shown in Figure 30.

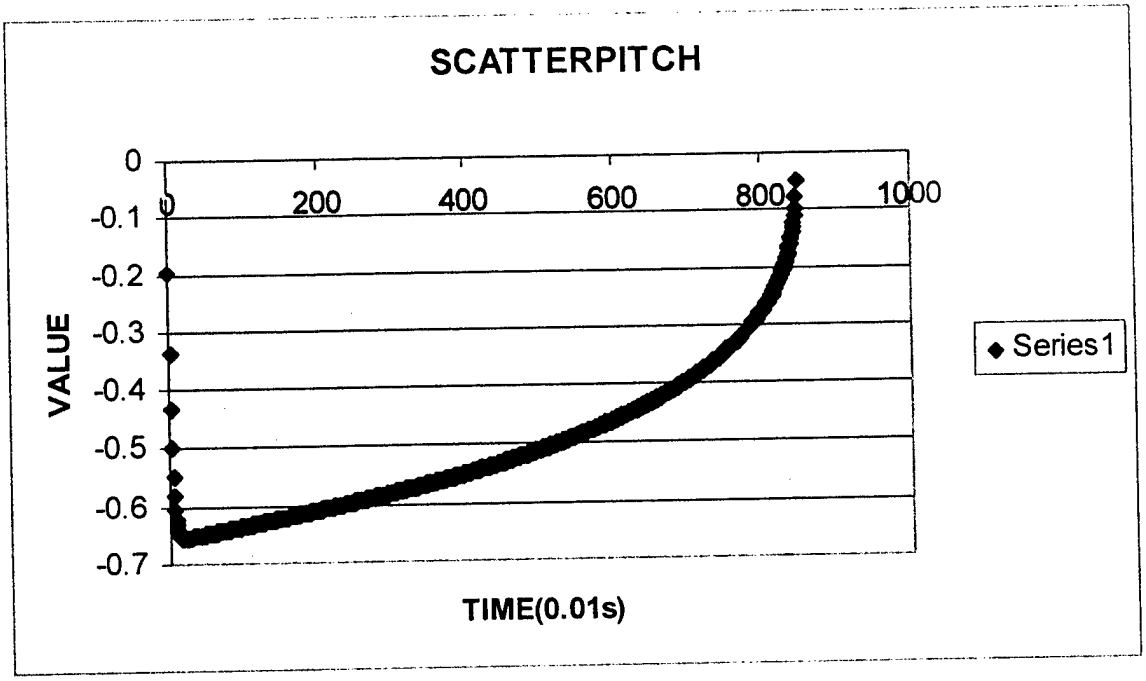



**Figure 29: Pitch Scatter Plot**

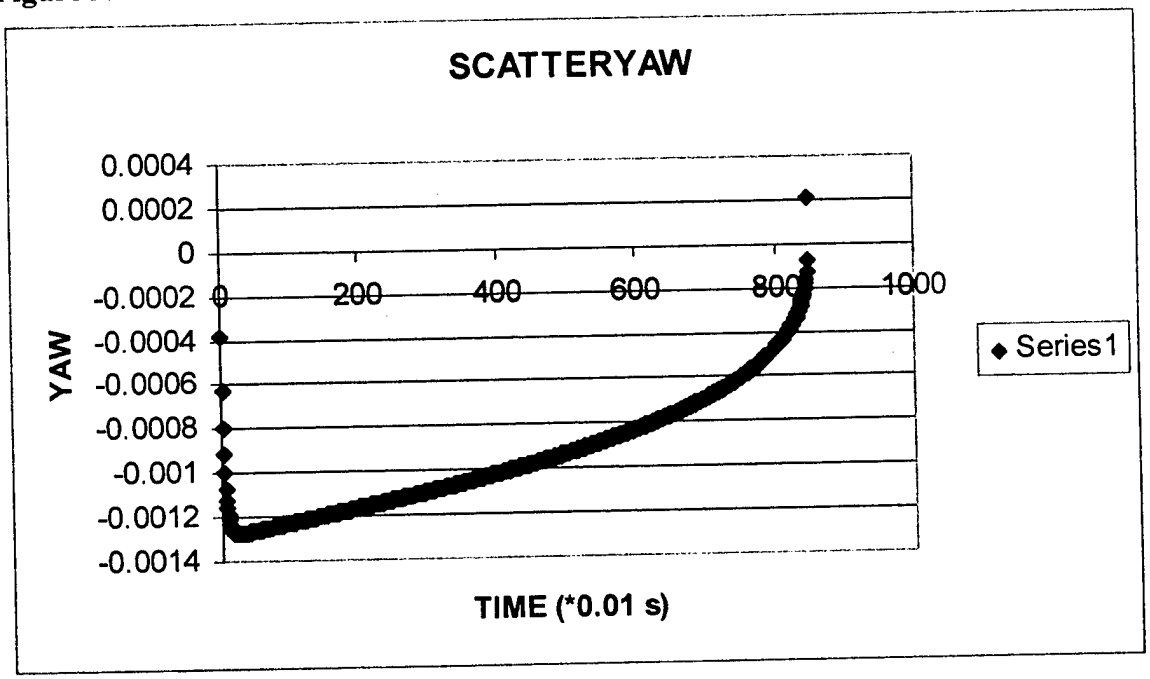



**Figure 30: Yaw Scatter Plot**

This preprocessing step was used not only for the value-based abstraction method, but also for the clustering abstraction methods demonstrated later.

### 3.3.2.3.2 Developing the Value-based Abstract Models

Once the average pitch and yaw values were identified during the pre-processing step, those values were used in developing the abstract representations of the pitch and yaw guidance calculations (and thus the spatial velocities). The average values used by this abstract missile platform sub-component model were pitch = -0.5055 and yaw = -0.00093878. Figure 31 shows the difference between the original model code and the abstracted model code. The abstract simulation scenario resulted in a PCA of 69.898 m or a difference of 68.687 m from the baseline results (which was 1.211 m).

```
 if(CurrentTime > LaunchTime + GuideDelay ) {
//  pitch = spatial_state.orientation[1] += GuideFactor*bse_el;
//  yaw   = spatial_state.orientation[2] += GuideFactor*bse_az;
// based on the new orientation, update the velocity
spatial_state.linear_velocity[0] = Velocity*cos(-0.000939)*cos(-0.5055);
spatial_state.linear_velocity[1] = Velocity*sin(-0.000939)*cos(-0.5055);
spatial_state.linear_velocity[2] = -Velocity*sin(-0.5055);
```

**Figure 31: Developing the Abstract Value-based Model**

### 3.3.2.4 Clustering Techniques

As reported, the value based abstract models did not prove to be very accurate. The next method of abstraction examined involved using clustering techniques. Most clustering techniques, such as Adaptive Resonance Theory Networks and Distributed Clustering Algorithms are too complex for this particular simulation and would result in simulation overhead that would actually increase simulation time. However, the concepts presented in clustering theory could still be used, albeit in a more rudimentary form. To apply clustering concepts, the preprocessing steps that identified the pitch and yaw scatter

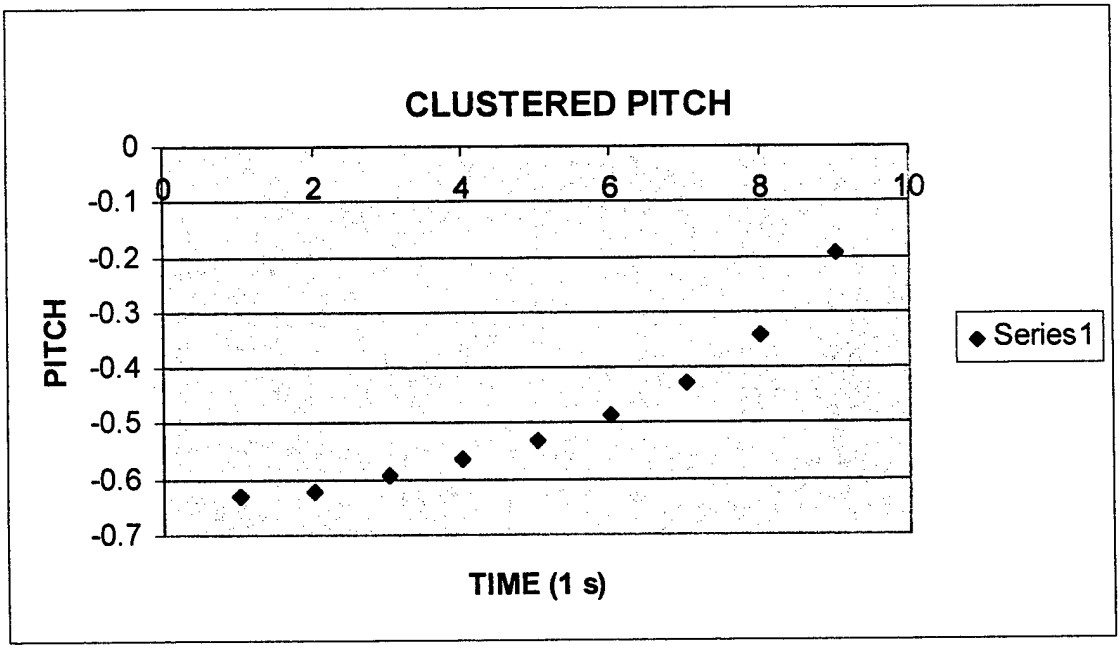



**Figure 32: 10 Cluster Pitch Values**

values with respect to the baseline simulation scenario were used. These plots were used to "cluster" the pitch and yaw values with respect to their value and time. Pitch and Yaw values grouped into nine distinct clusters are shown in Figures 32 and 33.

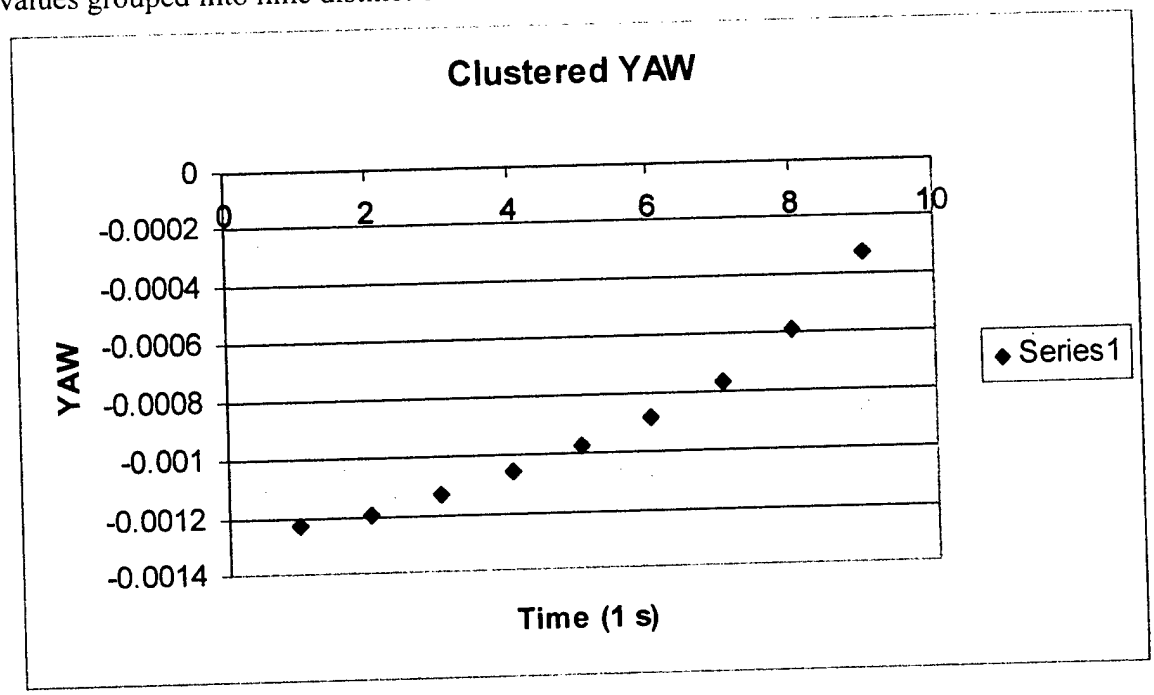



**Figure 33: 10 Cluster Yaw Values**

### 3.3.2.4.1 Quantization Techniques

The first type of clustered abstract model that was developed involved the use of quantization techniques. For each of the clusters, the average value for the pitch or yaw was calculated. All cluster values with respect to time were then quantized to that average value within the cluster. Four experiments (abstract representations) were conducted using 2, 3, 5, and 10 clusters. The average pitch and yaw values for each cluster are shown in Table 1.

| PITCH | | | | YAW | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | | -0.632021 | | | | -0.001224 |
| | | -0.625726 | -0.621706 | | | -0.001206 | -0.001195 |
| | | | -0.594418 | | | | -0.001127 |
| | -0.599615 | -0.573624 | -0.564348 | | -0.001142 | -0.001077 | -0.001055 |
| | | | -0.529364 | | | | -0.000975 |
| | | -0.493555 | -0.486281 | | | -0.000898 | -0.000880 |
| | | | -0.429492 | | | | -0.000764 |
| | | | -0.342341 | | | | -0.000594 |
| -0.5055 | -0.411376 | -0.329584 | -0.193486 | -0.000939 | -0.000736 | -0.000574 | -0.000326 |

**Table 1: Pitch and Yaw Cluster Values**

The graph of PCA versus number of clusters with respect to the simulation baseline is shown in Figure 34. As shown, in general, the more clusters that are used for abstracting the model, the more accurate the model becomes.

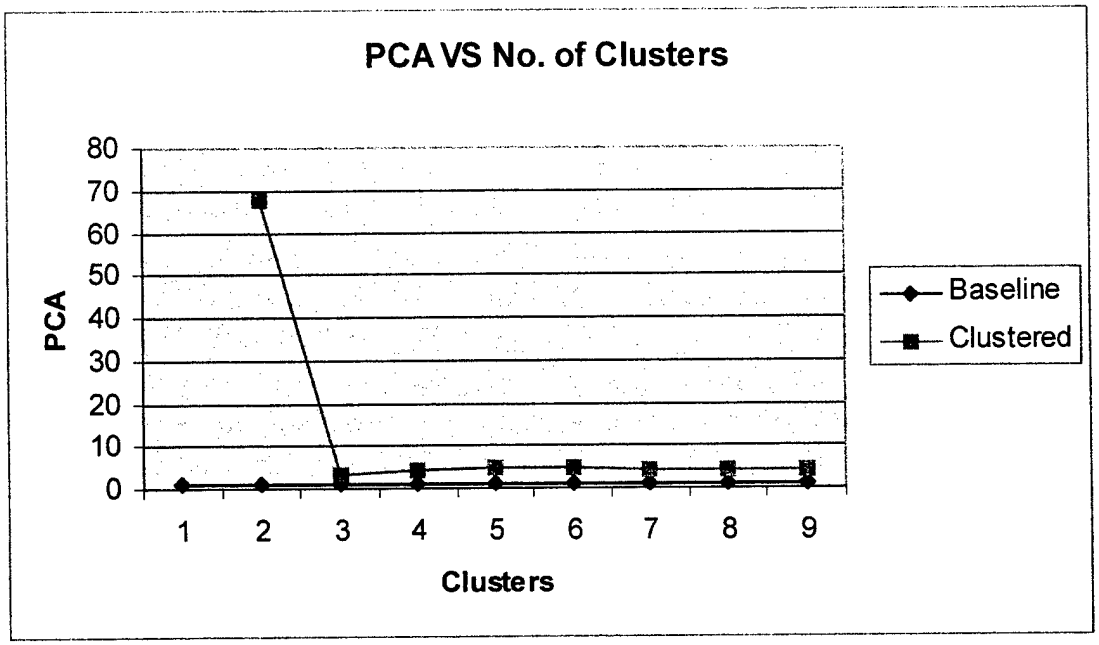



**Figure 34: PCA vs. Number of Clusters with Respect to Simulation Baseline**

### 3.3.2.4.2 Stochastic-Based Clustering

The second type of clustered abstract model that was developed involved the use of stochastic distribution techniques. For each of the clusters, the average value for the pitch or yaw was calculated using the preprocessing step. In addition, the standard deviation for each cluster was also calculated with respect to the average value. The average values and the standard deviations were used to specify a Gaussian distribution centered around the average value for each cluster. Four experiments (abstract

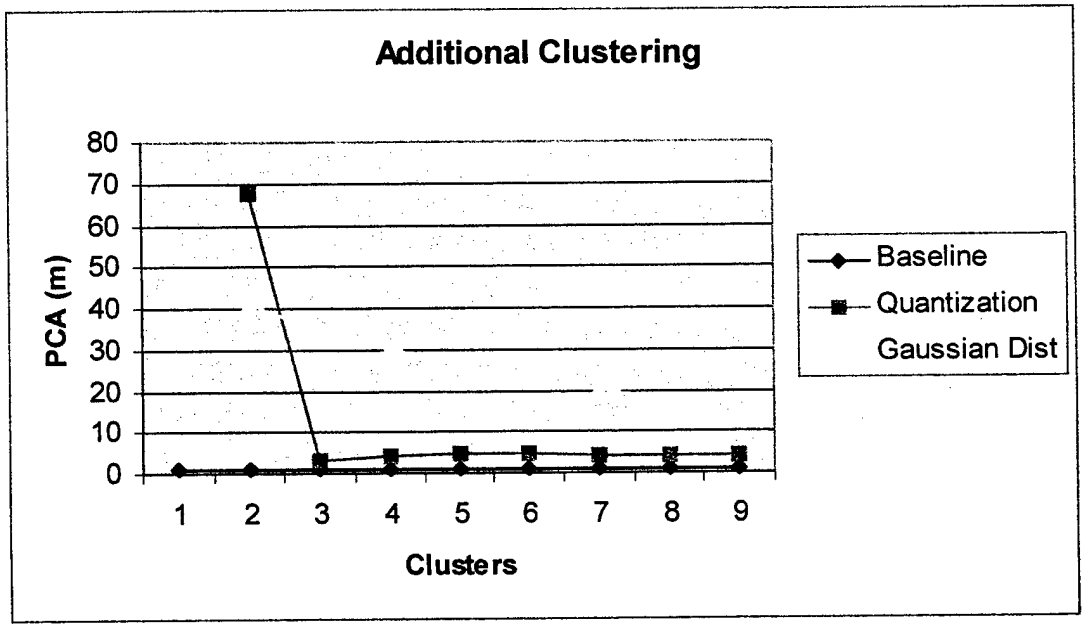



**Figure 35: PCA vs. Number of Clusters with Respect to Simulation Baseline - Gaussian-based Abstractions**

representations) were conducted using 2, 3, 5, and 10 clusters. The graph of PCA versus number of clusters with respect to the simulation baseline is shown in Figure 35. As shown, once again, the more clusters that were used for abstracting the model, the more accurate the model becomes. The stochastic based clustering method, however, did not prove to be as accurate as the quantization method.

#### 3.3.2.4.3 Performance of Abstract Clustering-based Simulations

The performance of the various abstract clustering-based simulations was determined for both quantization and stochastic based approaches by obtaining the wall clock and CPU simulation time from the simulation execution. The performance metrics were obtained for simulations using both 2 and 10 clusters for both the quantized and stochastic-based abstraction methods. The performance results from these simulations are shown in Table 2 in comparison with the simulated baseline scenario. The improvement in simulation performance ranged for 2-7% for the Dell 1 GHz. Platform running Windows NT.

| Scenario | Abstaction Method | Wall Time (s) | CPU Time (s) |
|---|---|---|---|
| Baseline | N/A | 1.712 | 0.44 |
| 2 Clusters | Quantized | 1.622 | 0.41 |
| 2 Clusters | Stochastic | 1.672 | 0.41 |
| 10 Clusters | Quantized | 1.662 | 0.41 |
| 10 Clusters | Stochastic | 1.692 | 0.43 |

**Table 2: Abstract Model Performance Measurements**

#### 3.3.2.5 Stochastic-based Abstraction

The next abstract modeling method examined dealt strictly with stochastic-based abstractions. The scenario used to demonstrate this method of data abstraction was a 2 on 2 Air-to-Air engagement scenario where each Airborne Interceptor fired two missiles on a specific Airborne target. The missiles were RF-based guided seekers. Upon examining the Missile's Seeker sub-component code, it was determined that a candidate for abstraction would be the calculation of signal power. For the baseline scenario, the Minimum Detectable Wattage was set to a very low threshold. Running the baseline simulation (in the same manner that the pitch and yaw values were calculated for the previous scenario) it was determined that the signal could be detected 99.3% of the time. The signal power calculations were replaced by a Gaussian distribution. The threshold was set to the area under the distribution curve where 99% of the values for the Gaussian distribution would fall. The results of this abstracted simulation scenario were exactly the same as the baseline results. The simulation performance results for the baseline and abstract version of this scenario is shown in Table 3.

| Scenario | Wall Time (s) | CPU Time (s) |
|---|---|---|
| Baseline | 3.172 | 1.922 |
| Abstract | 3.164 | 1.912 |

**Table 3: Performance Measurements for 2 On 2 Scenario**

### 3.4 Task 5 Demonstration Development

The fifth task for this effort involved the development of a demonstration that could illustrate the value and the effects of model abstraction techniques on a simulation. The scenario chosen for this demonstration was a JMASS JTEST One-on-One Air-to-Air Engagement Scenario using a Geometric Seeker. The demonstration used a visualization tool to view the differences between the detailed baseline and abstracted simulations. The goal of the demonstration was to overlay the abstract scenario on top of the baseline scenario in order to visualize the effects of abstraction by direct comparison.

#### 3.4.1 Selection of Visualization Tool

Three visualization tools were under consideration for use in this demonstration: AFRL's JVIEW, JMASS 5.0's Simview Pro, and JMASS98's Simview. AFRL's JVIEW required translating the JMASS98 simulation results into the proper format. Simview Pro did not require any translation, however, Simview Pro was not available at the time of the demonstration. JMASS98's Simview was chosen because it was the visualization tool inherently working with JMASS98 and did not require any file translation.

#### 3.4.2 Visualization Using Simview

Simview can be used to visualize JMASS98 simulations using a drag and drop process on NT-based platforms. JMASS98 simulation executions automatically create .SV files for each simulated team. The .SV file specifies all players involved in a given simulation, as well as the motion data file that maps that player's motion throughout the simulation execution. For example, in a given one-on-one engagement scenario (such as the baseline simulation scenario), the flight of the missile is mapped using the Missile.dat file. When the team's .SV is dragged to the Simview executable, the visualization tool reads from this Missile.dat data file in order to re-create the path of the missile on the visualization screen. In order to create a visualization overlay between the baseline simulation execution and the abstracted simulation execution, the missile data file from the abstracted scenario is imported into the baseline scenario by modifying the .SV file as

```
Player {
  ObjectName   = MissileAbstract
  Object = Sidewinder
  Key = d
  MotionFile     = missileAbstract.dat
  Trace      = on
}
```

shown in Figure 36.

**Figure 36: Modifying the Simview .SV file**

The .SV file specifies a new player in the baseline scenario, a MissileAbstract player, that points to the simulated motion data file obtained from the abstract simulation execution.

### 3.4.3 Demonstration Composition

Four visualization demonstrations were constructed for this project's demonstration. These demonstrations included:

- Overlaying an abstract simulation using 2 clusters against the baseline scenario
- Overlaying an abstract simulation using 9 clusters against the baseline scenario
- Overlying an abstract simulation using 2 clusters based on a Gaussian distribution against a quantized abstract model and the baseline scenario
- Overlaying an abstract simulation using 9 clusters based on a Gaussian distribution against a quantized abstract model and the baseline scenario

The visualization of the simulation execution showed how the increased number of clusters better tracked the baseline simulation execution. The visualization process also showed the effects of the removal of the spatial orientation calculation from the guidance calculations and the course correction of abstract model at each cluster boundary.

### 3.4.4 Demonstration Operation

The operation of the demonstration is described in Appendix A for each of the four simulation visualization demonstrations.

## 3.5 Task 6: Final Report

This report is the final report for the BAA effort. This report documents the progress, problems, and solutions encountered during this nine-month research effort.

# 4 Future Research and Final Results

RAM Laboratories successfully accomplished the tasks it set forth at the beginning of this BAA effort. This effort employed various model abstraction techniques to JMASS engagement simulation models and assessed the accuracy of those techniques with respect to the simulation baseline results and simulation performance. Several model abstract techniques were examined throughout the course of this project. Sensitivity analyses were used and modified to identify key simulation parameters within the context of the simulation application. Various techniques were used to abstract away model information in order to simplify the engagement level simulation. Techniques examined included value replacement, omission, stochastic processes, histograms, tables and clustering methods using quantization and stochastic-based replacement strategies. Results of this research were demonstrated through the use of the JMASS Simview Visualization tool.

For futurue work, it is recommended, three improvements to this project stand out:

- Use of a better tool for visualizing the simulation. The Simview tool is a very simplistic viewer. Much better visulization tools exist with Simview 5.0 and Jview.

- Use of more complicated models. This effort focused on unclassified models. As such, the advanced model abstraction techniques that could be examined were somewhat limited by the scope of the effort. More advanced and even higher fidelity models can be used to even better illustrate the advantages of model abstraction.
- Use of additional simulation environments. This particular environment has been rooted to the JMASS system and focused on engagement level simulation. It may be helpful to apply these techniques to a campaign or theater level simulation where other model abstraction techniques may be a better fit.

# 5 BIBLIOGRAPHY


Cassandras, C.G., Panayiotou, C.G., Gond, W.B., Liu, Z., & Zou, C. Clustering Methods for Multiresolution Simulation Modeling. Enabling Technology for Simulation Science IV, pages 37-49, April 2000.

Caughlin, D. An Evaluation of Simulation Annealing for Modeling Air Combat Simulations. Proc. 1994 IEEE Dual-Use Technologies and Application Conference, May 1994.

Caughlin, D. & Sisti, A. Summary of Model Abstraction Techniques. Enabling Technology for Simulation Science I, pp 2-13, April 1997.

Davis, P.K. An Introduction to Variable-Resolution Modeling: Model and Cross-Resolution Model Connection. Proceedings of Conference on Variable Resolution Modeling, May 1992.

Davis, P.K. Multiresolution Multiperspective Modeling (MRMPM) as an Enabler of Exploratory Analysis. Enabling Technology for Simulation Science IV, pg. 2-15, April 2000.

Fall, T.C. Dynamic Focusing Approach to Mixed Level Simulation. Enable Technology for Simulation Science I, 1997.

Frantz, F.K. Analyzing Models for Abstraction. Enabling Technology for Simulation Science I, 1997.

Heins, L.G. & Tauritz, D.R. Adaptive Resonance Theory (ART): An Introduction.

Hulth, N. & Grenholm, P. A Distributed Clustering Algorithm. Technical Report 74, Lund University Cognitive Studies, 1998.

Lee, K. & Fishwick, P.A. A Semi-Automated Method for Dynamic Model Abstraction. Enabling Technology for Simulation Science I, 1997.

MacDonald, Richard A. & McGraw, Robert M. Assessing Candidates for Model Abstraction. Enabling Technology for Simulation Science IV, 2000.

Paulus, Rich. Advanced Propagation Model (APM), 2000.

Sisti, A. & Farr, S. Model Abstraction Techniques: An Intuitive Overview. Proceedings of SCSC, 1998.

# APPENDIX A

## Demonstration User Instructions

The following instructions describe the process for running the four model abstraction demonstrations developed under this effort. These model abstraction demonstrations work in conjunction with the JMASS98 Simview visualization tool. These instructions target PC-based Microsoft Windows NT and 2000 platforms, however the demonstrations can also be run on a UNIX-based Workstation that supports JMASS98. It should be noted that the demonstrations do not require the execution of any simulation, but rather visualize the results of pre-compiled and simulated simulation scenarios that use the JTEST Model Suite of Air-to-Air Engagement scenarios.

1. Ensure that the target platform has the following software components installed:
   - PC-based Windows NT or Win2k based platforms
   - Microsoft Visual Studio Version 5.0 or higher
   - JMASS98 Version 1.3 or higher

2. Using Microsoft Explorer or any other navigation tool, CD to the following directory:
   <JMASS98 directory>/Pcp_dev/Teams

3. Copy all file folders from the Model Abstraction Demonstration CD to the <JMASS98 directory>/Pcp_dev/Teams directory. These folders include the following:
   - AAMSCENARIO1_ABSTRACT_CLUSTER2

   This folder contains a Simview simulation that compares an abstract air-to-air missile with a detailed air-to-air missile from the baseline scenario. The abstract air-to-air missile was developed by breaking the pitch and yaw calculations internal to the missile's platform sub-component into two clustered values that were quantized to the mean value for the cluster.
   - AAMSCENARIO1_ABSTRACT_CLUSTER9

   This folder contains a Simview simulation that compares an abstract air-to-air missile with a detailed air-to-air missile from the baseline scenario. The abstract air-to-air missile was developed by breaking the pitch and yaw calculations internal to the missile's platform sub-component into nine clustered values that were quantized to the mean value for the cluster.
   - AAMSCENARIO1_ABSTRACT_CLUSTER2DIST

   This folder contains a Simview simulation that compares two abstract air-to-air missiles with a detailed air-to-air missile from the baseline scenario. The abstract air-to-air missile was developed by breaking the pitch and yaw calculations internal to the missile's platform sub-component into two clustered values. One abstract missile used values that were quantized to the mean for each cluster. The second abstract missile used values represented by a Gaussian distributed centered around the mean value for the cluster.
   - AAMSCENARIO1_ABSTRACT_CLUSTER9DIST

This folder contains a simview simulation that compares two abstract air-to-air missiles with a detailed air-to-air missile from the baseline scenario. The abstract air-to-air missile was developed by breaking the pitch and yaw calculations internal to the missile's platform sub-component into nine clustered values. One abstract missile used values that were quantized to the mean for each cluster. The second abstract missile used values represented by a Gaussian distributed centered around the mean value for the cluster.

4. Use the Navigator to bring up the following directory:
<JMASS98 Directory>/PCP_dev/Tools/NT4.0intelVC5.0/Analysis
In this window there should be a "Simview" icon.

5. In order to visualize the simulation results, use the Navigator to bring up another window for the demonstration folder you wish to visualize. The path for this folder is:
<JMASS98 Directory>/PCP_dev/Teams/<Folder name>

6. Drag the .SV file for the corresponding visualization folder to the "Simview" Icon previously identified in Step 4. The corresponding visualization files for each folder are the following:

-AAMSCENARIO1_ABSTRACT_CLUSTER2
    -SIMVIEW_OVERLAY.sv
-AAMSCENARIO1_ABSTRACT_CLUSTER9
    -SIMVIEW_OVERLAY.sv
-AAMSCENARIO1_ABSTRACT_CLUSTER2DIST
    -SIMVIEW_OVERLAY.sv
-AAMSCENARIO1_ABSTRACT_CLUSTER9DIST
    -SIMVIEW_OVERLAY.sv

7. Dragging the .SV to the "Simview" Icon should bring up the Simview GUI and intialize the GUI for visualization of the target scenario. Once this GUI appears on screen, make the following adjustments for better viewing of the scenario.

-Set the "From" value to MissileReal
-Set the "Mode" value to Around
-Set the "Azimuth" value to 80
-Set the "Time Inc" value to 10
-For the Cluster9 demonstrations set the "Distance" value to 35
-For the Cluster2 demonstrations set the "Distance" value to 200

8. Using Simview, the user can view the projected missile path for the baseline and abstract missile based simulations.

***MISSION***
***OF***
***AFRL/INFORMATION DIRECTORATE (IF)***

*The advancement and application of Information Systems Science and Technology to meet Air Force unique requirements for Information Dominance and its transition to aerospace systems to meet Air Force needs.*