Proceedings of the 2005 Winter Simulation Conference
M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.

LANGUAGE BASED SIMULATION, FLEXIBILITY, AND DEVELOPMENT SPEED IN THE JOINT INTEGRATED MISSION MODEL

David W. Mutschler

NAVAIR Air Combat Environment Test & Evaluation Facility (ACETEF)
48150 Shaw Road, Unit 5, Bldg 2109, S115,
Patuxent River, MD 20670, U.S.A.

ABSTRACT

The Joint Integrated Mission Model (JIMM) uses generic system components and a simulation language that allows developers to program specific system, platform, and player characteristics, tactics, and doctrine. This permits great flexibility in simulation design and rapid modification of system types in complex simulations. However, the time and expense of developing complex simulations can be longer than desired. These costs can be mitigated by constructing scenarios for reuse and providing example scenarios for common use. In addition, a graphics user interface (GUI) can also facilitate reuse and perform some functions faster and more easily than can be achieved directly through simulation language text editing. This paper will discuss efforts in simulation construction, simulation reuse, and GUI development currently undertaken by the JIMM Model Management Office (JMMO).

1 INTRODUCTION

The Joint Integrated Mission Model (JIMM) is a general-purpose mission-level discrete-event simulator used primary for requirement analyses and installed system test (Lattimore 2004; Nalepka 2000; Nalepka, Gump, and Kurker 2001). JIMM is a major component of the Joint Strike Fighter Program Office (JSFPO) Strike Warfare Collaborative Environment (SWCE) toolset. It is also the means for integrated operation at the NAVAIR Air Combat Environment Test & Evaluation Facility (ACETEF). In these and other capacities, it has been used for a variety of efforts including Compass Call testing, Network Centric Warfare (NCW) analyses, Analysis of Unmanned Air Vehicles (Niland and Skolnik et al. 2005), Joint Theatre Missile Defense (JTMD) testing, and operation in the JSF Virtual Strike Warfare Environment (VSWE) events.

JIMM is maintained by the JIMM Model Management Office (JMMO). Originally created as a merge of the Air Force SUPPRESSOR and the NAVAIR Simulated Warfare Environment Generator (SWEG) models, JIMM was previously managed by the Electronic Systems Center (ESC) at Hanscom Air Force Base. It has resided at the NAVAIR ACETEF site since July 2004.

The JMMO currently manages two variants of the JIMM model. One variant is used primarily by the Joint Strike Fighter (JSF) Program and boasts of an ellipsoidal earth model, phantasms and kalman filtering, and other features. The second variant is known as the JIMM Advanced Combat Environment (ACE). JIMM ACE boasts of multithreading (with parallel execution of events), improved guidance, time-to-die, and additional features. Despite different internal structure however, they process the same simulation input known as the JIMM Conflict Language (JCL). Also, the JMMO is currently working to merge these two versions. Hence, this paper is equally applicable to them both.

1.1 JIMM Data Capture

Constructively, JIMM has extensive data capture and logging capabilities. This permits a straightforward understanding of what occurred during a simulation run. It also has a very flexible and easy to use post-processing capability that allows significant filtering of data. Results of multiple simulation runs can then be combined and analyzed in conducting studies and performing evaluations.

1.2 JIMM Virtual Operation

JIMM can also be used to generate virtual threat environments. It has the ability to allow the substitution of one or more specific systems by actual systems connected via an interface (I/F) that uses a well-established shared memory protocol known as Simulated Warfare Environment Data Transfer (SWEDAT). When so interfaced, the system can act and react as if it existed in the simulated environment.

The interfaced systems can take a number of different forms. It could a piece of equipment (known as the system under test (SUT)) on a hardware bench. It could be the collection of systems in a virtual cockpit in the ACETEF

Manned Flight Simulator (MFS). It could be any number of systems represented in environments such as the High Level Architecture (HLA) or the Distributed Interoperable Simulation (DIS) protocols. Moreover, the number of interfaced systems capable of operating together in the same simulation exercise is not restricted.

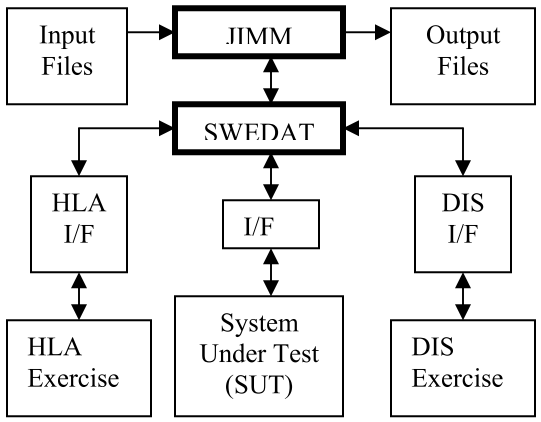

Figure 1: Integrated Operation in JIMM

1.3 JIMM Steps

Developing simulation exercises in JIMM usually consists of nine steps.

1. The Language Database (LDB) step is used to initialize the JIMM Conflict Language (JCL). This step is often run during JIMM installation and only its output is referenced thereafter. The LDB must always be the first step executed.
2. The Icon Database (IDB) step is used to generate symbols for the JIMM graphics display. IDB steps must all be executed before the RDB or CDB in which graphics are employed. This step is optional.
3. The Ground Database (GDB) is used to process Digital Terrain Elevation Data (DTED) files into an intermediate form. This step is only employed when a terrain skin is used in the exercise.
4. The Environment Database (EDB) step takes the files created by previous GDB steps and generates a terrain skin. This step is only employed when a terrain skin is used in the exercise and must follow all GDB steps.
5. The Type Data Base (TDB) step is used to define player type structures, tactics, and system characteristics.
6. The Scenario Data Base (SDB) is used to define instances of player types, provide locations and initial paths for component platforms. Initial system settings are set for specific platforms. During the initial SDB step, the use of terrain (using output from the EDB step), the earth model, and other scenario settings are established. Any change in these setting in subsequent SDB steps is ignored. The final SDB must have language for final scenario preparation in anticipation of scenario execution.
7. The Run Database (RDB) is used to execute the scenario and produce simulation output. Data captured and modes of output are specified. A random number seed can be varied when using multiple runs for analysis. The game time at the end of the previous step determines the initial game time. At the end of the final SDB step, game time is zero (0.0). Hence, subsequent RDB (or CDB) steps must have later end times than their previous steps.
8. The Configuration Database (CDB) step is similar to the Run Database step except that it also includes instruction for integrated operation with interfaced systems.
9. The Analysis Database (ADB) takes RDB (or CDB) output and filters the captured data for easier analysis. An ADB step can only be used to filter the output of a single RDB or CBD step. Multiple (different) ADB steps can be run using the same RDB or CDB step as input. ADB output can be either in text format or in a tab delimited format for importing into spreadsheets and other analysis tools.

2 PLAYER STRUCTURE

The main simulation object in JIMM is the "player". It is the entity where perceived data concerning threats and friends is managed and where all thinking operation is performed.

A JIMM scenario may contain multiple instances of players of the same type. Player types are defined in the TDB. These type definitions include tactics, platforms, and systems.

In the SDB, the specific instances of player types are defined and organized into one or more command chains. Command chains define specific relationship between players such as commander, subordinate, and peer. A single player may exist in more than one command chain. Moreover, command chains are further organized into different sides (e.g. "Blue vs. Red").

The components of a player (usually systems) may be located at different sites. Hence, these systems are grouped within a construct known as a "platform". A platform may or may not move within the scenario. A platform may also have associated shapes.

For example, a jet fighter may be modeled as a simple single platform player. In addition, a surface to air missile

(SAM) site may have two sensors and one weapon in different locations and hence be modeled as a player with three platforms. One fighter could only detect and react to a platform with a sensor while another fighter might only detect and choose not to react to the platform with the weapon. Conversely, though the SAM site exists in multiple locations, it would still have only one set of perceptions. Hence, if the two sensors each detect the different jet fighters, the player could allocate the weapon toward firing at either one of them.

Different systems within a platform may be grouped together in "elements". An element may have a susceptibility whereby it can be detected during a sensing event. In addition, elements may be destroyed during damage resolution.

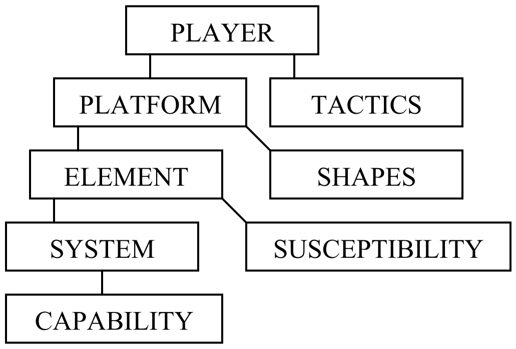

Figure 2: Player Structure

To continue the example, the jet fighter platform may have several elements. The element with weapon systems may have a susceptibility that allows it to be detected and targeted by the SAM site. When a shot strikes the target and damage resolved, the element may be destroyed. Assuming the lost element is not "critical", the fighter may survive the strike with its remaining elements intact, react, and then move away.

In JIMM, there are eight generic system types.

1. Thinker systems allow for processing of perceived data and execution of tactics (decision logic).
2. Sensor Receivers sense elements and emitting systems of target platforms.
3. Sensor Transmitters are paired with Sensor Receivers and emit energy used in detections. Transmitters do not need to be co-located with their linked sensor receivers.
4. Communication Receivers are needed for the receiving of message data from other players.
5. Communication Transmitters are needed to provide message data to other players.
6. Weapons are used to engage platforms and target their detected elements (if applicable). Weapons may be linked to sensors (trackers) if the shots are "guided" by those sensors to their targets.
7. Disruptors (Jammers) are used to inhibit the function of sensor and communication receivers. Like transmitters, the emissions of disruptors can also be detected by some sensor receivers (as programmed by the scenario developer).
8. Mover systems allow platforms to change position and orientation. Each platform may have at most one mover system.

Scenario developers program the specifications for each system type during the TDB step. In JIMM, these specifications are known as "capabilities". System capabilities may be used more than once in the same player and may also be used to define the same system type (or part of the same system type) in different player structures.

```
PLAYER-STRUCTURE abn_cmdr
  TACTIC abn_cmdr_tactics
  PLATFORM 1 abn_cmdr_a/c
    ELEMENT 11 abn_cmdr_ele
       DISCRETE QUANTITY: 1  CRITICAL
      MOVER 114 abn_cmdr_body
        CAPABILITY abn_cmdr_body_data
        FUEL jp-4 CONTINUOUS  4000. (KG)
    ELEMENT 12 abn_cmdr_comm_ele
       DISCRETE QUANTITY: 1
      COMM-RCVR 112 comm_rcvr
        CAPABILITY comm_rcvr_data
      COMM-XMTR 116 comm_xmit
        CAPABILITY comm_xmit_data
      COMM-RCVR 113 comm_rcvr
        CAPABILITY comm_rcvr_data
      COMM-XMTR 117 comm_xmit
        CAPABILITY comm_xmit_data
    ELEMENT 13 abn_cmdr_radar_ele
       DISCRETE QUANTITY: 1  NONCRITICAL
      THINKER 111 abn_cmdr_thk
        CAPABILITY abn_cmdr_thk_data
      THINKER 118 abn_cmdr_thk
        CAPABILITY abn_cmdr_thk_data
      SNR-RCVR 1110 abn_cmdr_rx
        CAPABILITY abn_cmdr_rx_data
      SNR-XMTR 1111 abn_cmdr_tx
        CAPABILITY abn_cmdr_tx_data
   LINKAGES
      111 WITH 1110   112 WITH 116
      1110 WITH 1111   113 WITH 117
END PLAYER-STRUCTURE
```

Figure 3: Sample Player Structure

2.1 Simulation of the Thinking Process

In JIMM, the thinking process is simulated on a per player basis. Each player retains "knowledge" about platforms. This knowledge is collectively known as a "perception". Information in the perception is garnered either through direct observation via its own sensor systems or through communication with other players. Moreover, information in a perception need not correspond to "ground" truth. The data can be out-of-date or entirely incorrect.

Perceptions are employed by players to make decisions (via tactics). Since perception data can be wrong, the resulting decision can also be wrong. Moreover, simulation of thinker systems can result in delays in perception updates and consequential delays in decision processes.

The modeling of thinking in JIMM is based on a mechanism known as a "pending queue". Each player has a single pending queue. When a thinking event is generated, it is not scheduled immediately. Instead, it is placed at the tail of the pending queue where it waits for a thinker system. If a system is available immediately, then it is set to "busy" and the event is scheduled for a future game time given a programmed "time to think" for that event type. Not all thinker system types can process the same type of thinking events. Also, different thinker system types can have different "time to think" settings for different thinking events.

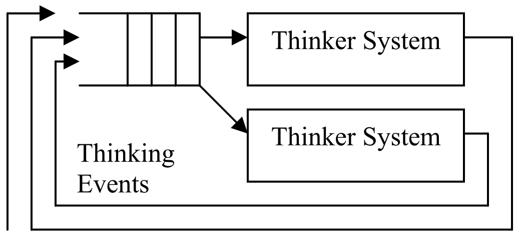

Figure 4: Pending Queue Operation

If all thinker systems are busy, then the event must wait on the pending queue until a thinker system is available. If too many thinking events occur, then the pending queue can have a "backlog". In other words, the player is overwhelmed and delays in its decisions will result.

2.2 Thinking Events

Thinking events are divided into five major categories.

1. Notice – The initial awareness of a perception or specific type of perception update. Notice events will usually result in corresponding 'digest' events.
2. Digest – The processing of a perception update. Digest events will result in 'review', 'react', and 'ponder' events given the programming in the player tactics.
3. Review – The review of perception. This includes its elimination if it hasn't been updated given a period of elapsed simulation time.
4. React – The allocation of player resources given perceived data. This is also known as "resource allocation". Actions (such as sending messages) can also be done.
5. Ponder – Also known as plan, this simulates more complex planning regarding categories of resource allocation or other player actions.

Resource Allocation is generally based upon system types where each system is a resource to be utilized 'against' a perception. Additional resources are perceived friends (commander, peers, and subordinates in the same command chains as the player), communication nets, and future players (from which new players are created).

Table 1: Resource Allocation Types

Resource Allocation Type	Resources
Absorb	Any system with expendables (e.g. fuel and ordnance)
Comm. Method Selection	Communication Nets on which transmitters operate
Emcon (Emission Control)	Sensor Transmitters (on or off)
Fill Request	Any system (usually a thinker) with expendables that can become created players
Intell Send	Perceived Friends (subordinates, commanders, and peers)
Lethal Assign	Perceived Friends
Lethal Engage	Weapons
Maneuver	Movers
Non-lethal Engage	Disruptors (Jammers)

The types of resource allocation can have one or more stages. Absorb, Emcon, Fill Request, Intell-Send, and Comm-Method-Selection each have a single stage. Lethal Assign, Maneuver, and Non-Lethal Engage have four stages (queue add, queue drop, start and stop), and Lethal Engage has six (queue add, queue drop, start, stop, firing start, and firing stop). The multiple stages better simulate stages in the thinking process. An example of one stage in the assignment logic of a player is below.

```
LETHAL-ASSIGNMENT-QUEUE-ADD normal_tactics
  TGT-TYPE drone attacker adhoc_fighter
  USE INPUT FOR FILTER 1
     SUB-TYPE close_sam
      3D-POSITION WITHIN engage_zone
        RE: PLATFORM/PT TARGET-LOC
      AND 2D-DIST < 100.0 (KM)
      AND BELIEVED-ALIVE
  USE FILTER 1 SELECTIONS FOR FILTER 2
     SUB-TYPE close_sam
      TARGET-ACTION IS shoot_to_kill
      OR ENG-CONTROL-MODE IS close_sam_cdr
  FROM FILTER 2 SELECTIONS
     CHOOSE-FROM
       close_sam
     PICK-AT-MOST 9 NOW
END LETHAL-ASSIGNMENT-QUEUE-ADD
```

Figure 5: Example of Resource Allocation

3 SCENARIO GENERATION

The ability to program player structures, system characteristics, and tactics (such as resource allocation) makes JIMM a very flexible model. System type characteristics, individual system specifications, player construction, and tactics and doctrine can be modified in the input file and processed through JIMM without modifying the JIMM executable. This can greatly facilitate analysis of different systems, platform configurations, and operational protocols, et al.

However, this flexibility also introduces a difficulty since a great number of parameters and tactics must be specified even for the most simple of systems. This can result in longer times for scenario generation.

Since reducing the time (and cost) of scenario generation is highly desirable, a number and techniques have been utilized achieve this reduction while also maintaining flexibility.

3.1 Reusing Simulation Components

In JIMM, significant savings in development time are routinely achieved through multiple utilizations of simulation components. Tactics, susceptibilities, characteristics, system types, element types, and even platform types can be used in multiple player structure specifications. This is a very basic form of reuse that operates with the same simulation.

Another method of reducing development time is to adapt existing simulations in whole or in part. Hence, a developer could 'cut' a desired component and 'paste' into a new simulation using any editing tool.

To facilitate this transferring of instructions, simulation code is often divided into "chapters" where each chapter describes a specific player. Furthermore, the top of the simulation file contains a listing of all chapters to facilitate any searching.

In addition, since JIMM can process multiple TDB and SDB files, different simulation components can be coded in different text files. Hence, no text file manipulation would be necessary for reutilization of player structures or specific sides in a scenario laydown.

Example scenarios are provided with the JIMM distribution for this very same purpose. The current JIMM distribution has the following.

1. Obruty Final Battle – A comprehensive example of JIMM features in a fictional setting. One scenario is updated with recent JIMM features. Another version dates from the earliest NAVAIR-maintained version of the Simulated Warfare Environment Generator (SWEG). This older version tests backward compatibility.
2. EIMSE – This scenario for "Enhanced Integrated Air Defense (IADS) Messaging in a Simulation / Stimulation Environment" focuses on communication. (Williams and Chapman 2001)
3. Parallel – A simple scenario used to test multi-threading.
4. EGADS – This set of seventeen scenarios for the "Enhanced Generic Air Defense System" was originally developed in the Air Force Research Laboratory (AFRL) at Wright Patterson Air Force Base (AFB). (Duquette 2003), (Duquette, Nalepka, and Luczak 2004).

3.2 Modifying Simulation Components

Even after a set of TDB files have been processed, information specific to a player type, platform type, element type, or system type may be modified. This is accomplished by a mechanism known as a "TDB overlay" wherein TDB instructions replace the specifications previously given. Since TDB overlays are normally programmed in the same file as the SDB instructions, they are often employed in analyses such as when comparing system parameters or platform configurations. The example below instructs all players using "attacker_tactics" to use the JIMM "threat avoid" capability when determining movement routes.

```
TDB
REPLACE-MODE
TACTIC attacker_tactics
  MOVE-OPTIONS
    THREAT-AVOID
  END MOVE-OPTIONS
END TACTIC
END-TDB
```

Figure 6: Example TDB Overlay

For specific players, it addition to those specifications that are required for each player (such as initial location and movement path), some parameters may be modified in the SDB step itself. This includes the availability of platforms and elements, initial tactics and contingency plans, known perceptions, criticality, et al. This permits even finer tuning of a scenario.

In the following example, the blue drone is a multiple platform player. However, since only one platform is specified, only that one platform will be instantiated in the scenario. Initial location, path points, and movement starting time are also provided for that platform. In addition, initial perception information is provided via the "TOLD ABOUT" statement and the initial parameters for contingency plans from the player tactics are modified for this one instance. Lastly, specific instructions are given for the sensor receiver. The default setting is "ON" at scenario start.

```
PLAYER: blue drone LEVEL: 1
  PLATFORM: 3 drone_a/c
   X,Y,Z: 260.0 -200.0 (KM)  350.0 (M)  AGL
   ELEMENT: 31 drone_ele DISCRETE QUANTITY: 1
      SNR-RCVR 312 drone_emit_rx OFF
       TURN ON AT TIME: 3698.0 (SEC)
   END ELEMENT
   PATH START TIME: 3675.0 (SEC)
      ALT: AGL MODE: 3-D WITH-TURNS
   X,Y,Z:   260.0  -200.0 (KM)   350.0 (M)
    SPD: 225. (M/SEC)  TURN-RADIUS: 2300. (M)
   X,Y,Z:   160.0  -260.0 (KM)   350.0 (M)
   X,Y,Z:    70.0  -210.0 (KM)   350.0 (M)
   X,Y,Z:   -85.0  -170.0 (KM)   1050.0 (M)
   X,Y,Z:  -105.0  -135.0 (KM)   1050.0 (M)
   X,Y,Z:   -30.0  -140.0 (KM)   1050.0 (M)
  END PLATFORM
  INITIAL PLAN FOR MOVEMENT    get_started
  TOLD ABOUT blue drone PLATFORM 3
     X,Y,Z: 260.0  -200.0 (KM)  350. (M)  AGL
BY blue drone
END PLAYER
```

Figure 7: SDB Player Instructions

In addition, some parameters may be modified by interface programs during integrated operation. External programs can inject players, inject perceptions, create and send messages, and take over decision logic (in whole or in part), and dynamically return that control back to JIMM.

3.3 The JIMM GUI

The JIMM Graphics User Interface (GUI) also known as “Pisces” was added to the JIMM Model Management Office (JMMO) distribution in 2005 (Briere et al. 2005). This tool currently works for the Scenario Database (SDB) and later steps. It allows reuse of scenario components specified in different TDB files and facilitates the simple and straightforward construction and modification of scenarios given existing player types. The following screenshot of the GUI above shows a “very high” level plan view.

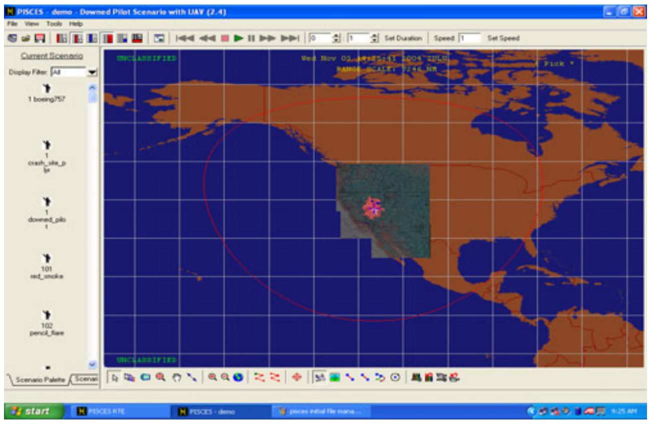

Figure 8: GUI Scenario Generation (High Level)

Once zoomed in, the GUI enables the adding of players of specific types to the scenario via a palette (icons) and the mouse to indicate a specific location. The following screen shot shows more detailed information where the path for a platform is being generated using the toolbar, menus, and the mouse.

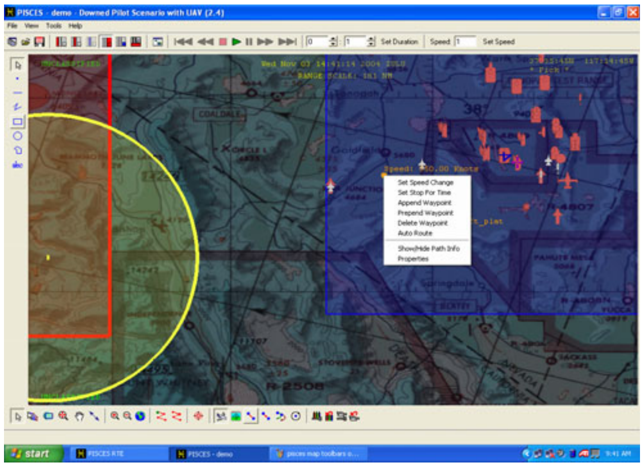

Figure 9: Path Generation with the JIMM GUI

In addition to scenario generation, the JIMM GUI provides several run-time scenario displays with loadable map options, viewers, programmable charts, and scenario monitoring features.

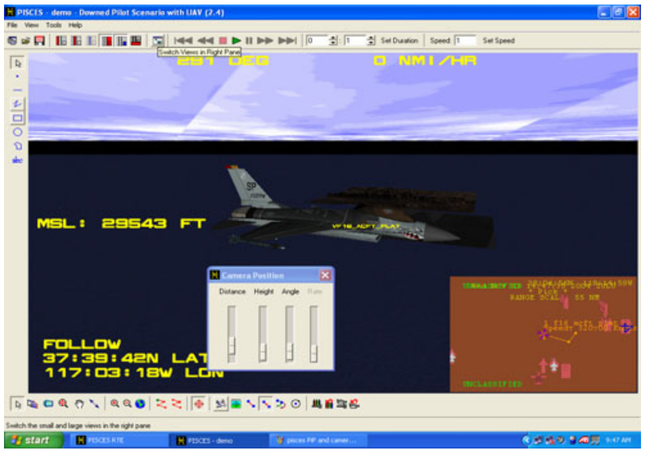

Figure 10: GUI Runtime Display of a Platform

4 SCENARIO ANALYSIS

In JIMM, the main unit of captured data is an “incident”. JIMM currently has over 150 different incident types. An incident is a time-tagged recording of an action or change that occurred during a simulation run. Incidents are roughly based on English grammar. They include from one to three players (subject, object, and indirect objects), a single action (verb), and auxiliary data specific to the incident.

During post-processing in the ADB step, filtering can be done on the players, incident types (actions), auxiliary data, and time windows. The incident filtering mechanism in JIMM is known as a "situation". Output can be either in the form of English-like syntax or a tab-delimited format suitable for incorporation into spreadsheet and database programs.

```
23:59:37.8 Day 365
  A3 aaa_gun (1 aaa_gun_veh), in envelope, intercepts
      8 gun_tgt (2 gun_tgt_area)
      weapon: uncontrolled; ordnance: gun_shell
      tgt (x,y,z): -369.0 km  -339.0 km  0.000 km; Heading 90.0 deg;
      wpn (x,y,z): -372.0 km  -339.5 km  0.0 km
      az: 80.5 deg; el: 0.0 deg; 3D dist: 3.0 km; 2D dist: 3.0 km;
      relative az: -170.5 deg; el: 0.0 deg: P9k): 0.05000
      launch time: 23:59:34.7 Day 365; commander: 11 aaa_gun_cdr
```

Figure 11: JIMM Incident Output (Text)

Situation output can be further filtered to provide only the counts of those incidents that meet the filter. This is very useful for analysis. For example, given a simulation run where a jet fighter flies a mission deep into enemy territory, a situation could be employed to determine the count of shots fired at the fighter by all "SAM players" of a given player type. Given multiple simulation runs (with different random number seeds), the counts could be easily analyzed toward mission survivability (or other purpose).

```
SITUATION: close_sam_fires_a_weapon
   THE independent close_sam
      FIRES-A-WEAPON-AT
         THE 71 vis_fighter
END SITUATION
```

Figure 12: Example JIMM Situation

Given the extensive use of JIMM for analysis, numerous examples of situation logic are provided in the example scenarios. In addition, the JMMO provide a set of tools used for analysis with each JIMM release. JIMM users are also free to submit tools to the JMMO for incorporation and subsequent reuse.

JIMM is extensively tested for each release (Chapman 2005). Most of the tests are automated and consist of the execution of small JIMM scenarios (known as 'vignettes') and their subsequent analysis for correct operation. Over 3000 automated tests are in place. The automated acceptance test suite is currently being restructured as part of an effort to leverage multiple processors during test execution. One effect of this restructuring will be to make these analysis scripts and program more readily available to the user community. Though the test suite is available to JIMM users on request, this greater ease of access should greatly facilitate reuse.

5 FUTURE WORK

As part of its ongoing efforts, the JMMO is working to both develop and acquire scenario databases for distribution to the JIMM community. This should promote better scenario development and provide a base for scenario component reuse.

The JIMM GUI is also a major focus of the JMMO. Efforts are underway to enhance its capability and increase its ease of use.

In addition, the JMMO is currently proposing the developing a specific TDB database for use with the GUI. This database would contain complex specification describing the physical characteristics of known platform types. However, tactic would be simplistic. This would allow the insertion of platforms into a scenario and provide an "80%" solution for scenario development. Simple scenarios could be developed more quickly. In addition, the databases would still be available for the development of more complex and more scenario specific tactics.

Another possibility for future work lies in the establishment of default parameters and behaviors. This would allow JIMM players and platforms to be developed in less time since data would already be known and would not need to be explicitly programmed.

6 ADDITIONAL INFORMATION

The Joint Integrated Mission Model (JIMM) Model Management Office (JMMO) provides management, development, testing (Gibson and Chapman 2001), support (including phone support), and other services pertinent to JIMM. The JMMO is currently housed in the NAVAIR Air Combat Environment Test & Evaluation Facility (ACETEF), Battlespace Modeling & Simulation Division (Code 5421). The JMMO may be reached via e-mail at <jmmo@navy.mil>.

7 CONCLUSION

This paper has described language-based simulation in the Joint Integrated Mission Model and shown the use of player structures, generic systems, and tactics. It has described several methods for component reuse as well as plans by the JIMM Model Management Office to facilitate further reuse. All in all, these efforts should reduce the time and cost for generating and analyzing JIMM scenarios while maintaining current flexibility.

8 REFERENCES

Briere, Marc et al. JIMM 2.4.2 GUI user's guide. 2005. Available at <jmmo@navy.mil>.

Chapman, Michael et al. JIMM 2.4.2 acceptance test plan. 2005 Available at <jmmo@navy.mil>.

Duquette, M, Nalepka J, & Luczak R. The enhanced generic air defense system. *AIAA Modeling and Simulation Technologies Conference and Exhibit*. Providence RI (Aug 16-19 2004). AIAA-2004-4799

Duquette, Matthew M. The enhanced generic air defense system – an unclassified threat environment for JIMM. *Nov 2003, JIMM User's Group*. Available at <jmmo@navy.mil>.

Gibson, R.D, and Chapman, M. Legacy software testing – a current methodology. *The Eleventh Annual International Council On Systems Engineering (INCOSE)*. Melbourne, Australia. (July 1-5, 2001).

Lattimore, Peter et al. JIMM 2.4.2 Users Guide. July 2005. Available at <jmmo@navy.mil>.

Nalepka, J.P. JIMM: The next step for mission level simulation models. *AIAA Modeling and Simulation Technologies Conference*, AIAA 2000-4491, AIAA, Washington D.C. 2000

Nalepka J., Gump J., Kurker R. JIMM: The next step for mission models. *2001 SPIE Aerospace/Defense Sensing and Controls Conference (#2367)*, Orlando FL (April 2001).

Niland, William; Skolnik, Brian; Rasmussan, Steve; Finley, Kevin; Allen, Kevin. Enhancing a collaborative UAV mission simulation using JIMM and the HLA. *Proceedings of the Spring 2005 Simulation Interoperability Workshop*, Simulation Interoperability Standards Organization, San Diego CA, April 2005

Williams, Stacy & Chapman, Michael. EIMSE example scenario documentation. Available at <jmmo@navy.mil>.

AUTHOR BIOGRAPHY

DAVID W. MUTSCHLER obtained his PhD in Computer and Information Science from Temple University in 1998. He has been employed by the Naval Air Systems Command (NAVAIR) for twenty years and has been working with the SWEG and JIMM models for nearly ten years. He is a member of the Association for Computing Machinery (ACM), the ACM Special Interest Group in Simulation (ACE/SIGSIM), the Institute of Electrical & Electronics Engineers (IEEE) and the IEEE Computer Society (IEEE/CS). He is currently serving as the JIMM Model Manager and head of the JIMM Model Management Office (JMMO). He is also an assistant professor at the Florida Institute of Technology (FIT) School of Extended Graduate Studies (SEGS) at Patuxent River, MD. His e-mail address is <david.mutschler@navy.mil>.