Proceedings of the 1996 Winter Simulation Conference
ed. J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain

A SIMULATION OF THE EVACUATION OF AMERICAN CITIZENS WITH AN OBJECT-ORIENTED, ANIMATED MODEL

Jeffrey E. Sumner
Eric A. Zahn

TASC, Inc.
2555 University Boulevard
Fairborn, OH 45342, U.S.A.

ABSTRACT

This paper highlights a model developed by TASC, Inc. under contract to the Defense Advanced Research Projects Agency (DARPA). This project models and simulates an evacuation of American citizens and other important foreign nationals from a destabilized foreign country by utilizing available U.S. military resources. This project demonstrates how different evacuation plans can quickly be compared analytically through computer simulation. The software chosen for the simulation and analysis is the Integrated Model Development Environment (IMDE), an object-oriented discrete-event simulation package. The implementation of the model features reconfigurable mission plans and detailed simulation-independent animations. This model was demonstrated at the 1995 Joint Warfare Interoperability Demonstration (JWID).

1 INTRODUCTION

The Department of Defense (DoD) has an ongoing goal to improve the quality and accuracy of crisis action planning in an increasingly constrained global environment. With decreasing manpower and resources, greater emphasis has been placed on tools to aid the high level military planners. When a crisis situation arises, and the National Command Authority (NCA) has determined that military intervention is required, a Joint Task Force (JTF) is assembled to cope with the situation. This paper highlights research in the application of simulation as a JTF planning tool.

Much of the current research and development in military support systems is presented each year at the Joint Warfare Interoperability Demonstration (JWID), which features the application of new technologies to a hypothetical scenario indicative of modern military engagements. For JWID 1995, the scenario was partly focused on a humanitarian assistance situation arising from a typhoon in the southwestern Pacific. During the relief efforts, civil unrest occurs in the fictional nation of Gompin, a nation with strong ties to the United States and home to a large number of U.S. citizens. The unrest forces the NCA to authorize a Non-Combatant Evacuation Operation (NEO), with the goal being to safely evacuate all U.S. citizens with minimal loss of life.

A JTF is assembled to plan and execute the NEO operation. To help the JTF with this planning process for the NEO, a simulation was created using IMDE, the Integrated Model Development Environment. The simulation takes into account all possible resources that could be used in the operation, and allows the JTF planners to enter a number of different evacuation plans. These plans include information about what resources to utilize, what times the evacuation order should be given, and other pertinent data. The simulation of the plan is animated so the JTF planners can visualize plan execution and identify any problems before a Course of Action (COA) is actually selected.

Section 2 of this paper describes the model in-depth, including the different requirements that need to be incorporated. Section 3 describes IMDE, the simulation package chosen for the project, how the model was implemented, and the animation capabilities. Section 4 narrates how the final project comes together, explaining what is involved in the final version of the project and what was seen at JWID '95. Section 5 lists recent improvements to the model; finally, Section 6 concludes the paper by summarizing the project and proposing enhancements.

2 MODEL REQUIREMENTS

JWID '95 focused on providing joint service commanders with the technological resources to deal with a disaster scenario. In this case, an island nation in the South Pacific has been hit by Typhoon Jennifer, an intense tropical storm. This nation, fictionally named

Gompin, was devastated; electrical and telephone services were eliminated for almost all of the island.

At the request of the Gompin government, the United States immediately begins mobilizing disaster relief efforts. The already strong rebel factions in Gompin seize the opportunity to challenge the strained Gompin government in the aftermath of the storm.

At this point in the disaster scenario, U.S. advisors anticipate that matters will worsen on the island and that it will become unsafe for any U.S. citizens to remain in Gompin. Intelligence reports indicate that riots and violence, directed towards anyone connected with the current government, may break out at any time. The decision is made by the NCA to assemble a JTF to evacuate all American citizens and designated foreign nationals from Gompin as quickly as possible with minimal civilian casualties.

2.1 Specifics of a NEO

The NEO involves many steps in its execution. First, a decision must be made as to how many planes and ships will be used in the operation. There are two major U.S. Air Bases in the region available for the operation, Kadena AB at Okinawa, and Yokota AB just west of Tokyo, Japan. Both have C-130s and C-141s ready to transport citizens from Gompin to Guam, another island which is a U.S. protectorate that has escaped major damage from Typhoon Jennifer.

There is also the issue of where the evacuation point(s) on Gompin will be. The Gompin government has guaranteed the use of an air strip to the U.S.; however, it is away from the majority of American citizens living in the capital city area. There is a commercial airport available for the capital city, but the JTF should not rely on that airport to be open due to the civil unrest. In addition, a smaller airport is available near the northwest part of the island; but intelligence suggests it has a short runway and cannot handle many of the larger transports. The one major Gompin seaport will also be considered.

Transportation of the citizens to the evacuation points is also another issue. The island contains a network of highways which the citizens will utilize to get to the evacuation points. The actual speed of the citizens will vary based on terrain and road conditions. Also, some highways may be impassable due to the rebel activities on the island; these roadblocks will also have to be modeled.

Once the American evacuees arrive at the evacuation points, they will either load onto an American plane or ship already present, or will wait for one to arrive. The times of departure for the planes and/or ships will be determined by an external plan file, which will be read in by the model. More specifics and requirements of the plan file will be detailed in section 3.5. The planes, once they arrive, will load the evacuees and leave for Guam. Each evacuee has a certain loading time that must be factored in; also, the issue of how long a non-full plane will wait at a port for other evacuees to arrive must also be addressed in the model.

The final result of the model will enable the JTF to enter different plans into the simulation and analyze the effectiveness of each plan. For instance, if the plan is to utilize only half of the planes available, keeping half of the force in reserve for other contingency operations, how much longer will the evacuation process take? The JTF can receive this information from the NEO simulation and then make a knowledgeable decision.

2.2 The Object Model Working Group

The DARPA JTF Advanced Technology Demonstration (ATD) is a demonstration of applied technology for the next generation command and control environment. This system includes an open architecture, infrastructure, various servers, and a wide array of applications. In order for the servers and applications to share information, they must also share the internal representations of the data. For instance, the situation server needs to pass the current situation to the planning application so the JTF planner can review the situation and act accordingly. The situation would be contained in a collection of objects, each with its own states and behaviors.

To facilitate widespread intercommunication, there needs to be a standard for the objects so each application knows the structure and capabilities of the objects. The Object Model Working Group (OMWG) is a DARPA subgroup in charge of developing a common schema that defines the common objects for use in the JTF-ATD. A skeleton schema is already available, with naming conventions and concept documentation. Figure 1 shows a simplified extraction of the OMWG hierarchy for two classes representing military aircraft, the C-130H and the F-18. Both descend from the Platform class, one of the key classes in the hierarchy (Carrico 1995).

One of the major objectives of the NEO model was to evaluate the OMWG for application in an object-oriented analysis model. The objects of the NEO simulation represent a small subset of the OMWG schema, but have provided valuable insight into the use and complexity of such a schema in modeling and simulation. The following section details the software that was chosen for the model and how the model was built to fulfill the requirements listed in this section.

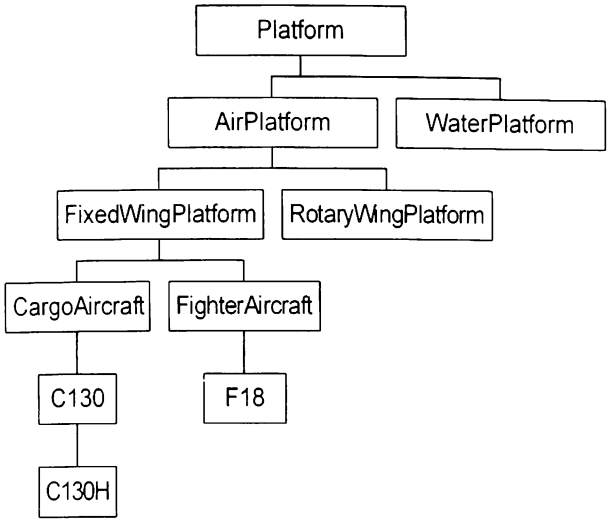

Figure 1: The OMWG Schema for the C-130H and the F-18

3 NEO Implementation

The development of the NEO model was performed by integrating several different software engineering tools and methodologies. This section briefly reviews several of these. First, a discussion of the Integrated Model Development Environment (IMDE) is presented. Following this is a description of the class selection and design process used in the architecture of the NEO model. Third is a discussion of how the Model-View-Controller (MVC) methodology was used, as well as the benefits received. Next is a discussion of the animation capabilities that were incorporated into the model. Last is a presentation of the plan file, which provides reconfigurable mission plans for rapidly developing and simulating various COAs.

3.1 An Overview of IMDE

IMDE is a domain-independent CASE tool used to develop and analyze object-oriented discrete-event simulation models. The USAF Logistics Research Division of Armstrong Laboratory has funded the development of IMDE at TASC, Inc. since 1990. It is a single application running under either UNIX or Windows NT, and directly interfaces with the Versant Object-oriented Database Management System (ODBMS). IMDE was chosen as the simulation package for this project due to its flexibility in modeling detailed situations, its object-oriented, modular development environment, and its graphical descriptions of the behavior of objects in the simulation.

IMDE is strongly based in object-oriented technology. All model parts in the simulation are defined as "objects" that have both variables that describe the state of an object, which are called attributes, and functions that specify its behavior, which are termed methods. By storing all objects in a distributed ODBMS, simulation objects can be shared and re-used in other simulations without recoding.

Objects from any field of study can be defined in IMDE. Models from the aerospace, business, education, and manufacturing fields have all been simulated and analyzed with equal success. The construction of simulation projects in IMDE can be done in a group environment, which allows multiple developers to create objects simultaneously and link them together. Finally, IMDE contains a comprehensive data analysis environment, where simple statistics, confidence intervals, hypothesis testing, and sensitivity analysis can be performed. Time trace plots of desired values can be shown quickly and easily.

The construction of models within IMDE is based on building objects and graphically assembling them to create complete simulation models. Templates for the design of these objects are called classes. For example, a C-130 class would be created that would contain all of the attributes and methods necessary to model an actual C-130 aircraft in the simulation. Several "instances" of this single class can be created in the simulation, each of which would model a specific C-130 aircraft. The definition of a class in IMDE includes detailing its attributes and methods. A graphical editor is employed to visually define all methods of any class. The editor allows a developer to define different behaviors of a single class with a separate flow chart that sequentially details the behavior logic. Flow charts are created by dropping nodes that represent simulation actions onto a canvas and connecting them in the correct order. IMDE uses these networks to generate source code in MODSIM II, an object-oriented simulation language, or C++ (Clark 1994).

Once all classes for a simulation have been defined, they can be linked together to form a model in IMDE's project editor. This editor allows the user to specify which entities will be present in the model and at what quantities. The project diagram illustrates the "has a" relationship of model parts, thereby representing the aggregation relationships of the model. With this framework in place, IMDE allows the user to further configure the model. Class substitution can also be performed, replacing objects in the model with other objects sharing the same inheritance hierarchy. For instance, if a modeler wants to determine the impact of using C-141 aircraft instead of C-130s, then a new C-141

class would be developed and substituted for the C-130 class in the simulation.

Also, lists of objects can be shortened or expanded in the project diagram. For example, one of the attributes of the Resource class in NEO is a list of aircraft. The initial number of aircraft that the Resource contains can be changed at this level, and a new simulation could be run without recompilation. Each component of the project can have input sets and statistics collection sets attached, detailing which attributes are parameters to be specified by the user, and which are noted for statistics collection.

Once the project diagram is complete, IMDE generates source code in MODSIM II or C++. This source code is then compiled and linked to create an executable simulation. All of this, including the running of the simulation and subsequent data collection, can be performed within IMDE.

At the conclusion of the simulation run(s), the data generated by the statistics collector is loaded into the database and can be analyzed within IMDE. As stated above, IMDE provides a complete data analysis package, calculating standard statistics as means, deviations, skews, and medians, as well as generating histograms, time traces, and scatterplots from the data.

3.2 Class Selection and Design

A main requirement is to have the class selection for the NEO model adhere to the OMWG schema. Therefore, a large number of classes were developed in order to match the OMWG schema and satisfy the needs of the NEO model. Since the number of classes involved is so large, this section briefly describes only the base level classes.

At the root level of the simulation is the JTF class. This class contains a list of Resources, a list of SafeHavens, and a list of HotSpots. The JTF class is responsible for reading in the plan and mission leg files, discussed later, and starting up the entire operation. A HotSpot is any area where an evacuation needs to occur. A HotSpot contains a road network that is used by the evacuees to travel to the evacuation points.

The OMWG schema defines a class called a Platform with serves as a parent to most vehicle classes. Therefore, all aircraft and ship classes defined in the NEO model descend from the Platform class. This allows certain operations to be performed by Platforms, such as traveling to specific locations, loading and unloading evacuees, etc., where it is irrelevant whether the actual vehicle is an aircraft or a ship.

The OMWG also defines a class called Port, which serves as a place of departure or arrival for Platforms. This Port class was used to model the actual air and sea ports that American citizens travel to in order to be evacuated. The SafeHaven class is descended from Port because it can not only serve as a port, it also serves as a location where evacuees can be transported. The Resource class models an air or sea base that houses Platforms, and is descended from SafeHaven. Each Resource contains a list of Platforms that may be used in the evacuation operation.

A class called PeopleGroup is used to model a generic group of people. An instance of this class represents a group of evacuees that travels together to the nearest Port and waits to be evacuated. Descended from the PeopleGroup class is a class named RebelGroup. This class models actions such as rebels taking control of a Port and destroying links in a road network.

3.3 MVC Architecture

NEO was developed under a powerful object-oriented methodology called Model-View-Controller (MVC). This architecture separates the responsibilities of software components into three different layers. The model layer handles all of an application's functionality. The view layer implements all feedback mechanisms used to present information to the user. Lastly, the controller layer handles all interaction that a user has with the application (Barkakati 1991).

NEO makes extensive use of the model and view layers. Less attention was allocated to the controller layer because all user interaction is handled inside of IMDE rather than during the actual execution of the simulation. The model layer was implemented through the development of the classes described in the last section. These classes provide the necessary functionality by modeling all of the real-world objects, such as planes, ports, evacuees, etc., that have a part in this simulation. These classes, and more importantly the interaction between them, provide for the accurate simulation of an actual evacuation operation.

The view layer of the NEO simulation provides the mechanism for displaying all information to the user during the execution of the evacuation operation. Several windows are displayed showing maps of the locale(s) where the evacuation is taking place as well as an expanded overview map which depicts the entire operation. Each object in the model layer contains a list of views that will need to display a graphical representation of the object. As the state of the object changes, the views are updated. For example, the current NEO simulation displays two windows, the evacuation location (Gompin) and an overview map of southwest Asia. A particular instance of a C-130 object would have a list of views that contains these windows.

As the C-130 travels, each view is updated. The C-130 will always appear on the map of southwest Asia, but it will also appear on the map of Gompin as it approaches that location. This provides a true synchronous "zoom-in, zoom-out" display. Since the different views operate independently of each other, the C-130 object does not have to coordinate how it is displayed in the different windows. Instead, the C-130 simply iterates through its list of views telling each one to update the C-130's representation given its new position.

The power of the object-oriented design and the MVC architecture is evidenced by the high flexibility of the system. Additional windows could be added as separate views. Appropriate backgrounds such as maps would have to be loaded, and coordinate systems would need to be set. Since the functionality and display responsibilities are separated, objects such as the C-130 would not need to be modified to display itself on the new views. Instead, the C-130 would simply have the new views added to its view list so that the new views could be told to update themselves as the state of the C-130 changes.

3.4 Animation

The animation capabilities of the NEO simulation provide the user with a top-down view of all activities occurring during the model's execution. The main window shows a scaled-out view of Southeast Asia. This map shows several areas of interest: United States air bases (Yokota and Kadena), a United States carrier fleet (Comphibron Eleven), an evacuation "safe haven" (Guam), and an evacuation site, or "hot spot" (Gompin). Air and sea networks show the routes that planes and ships will take during the simulation. This window shows the evacuation operation at a high level as planes and ships travel to the evacuation points, load evacuees, travel to the safe havens to drop off the evacuees, and return to base. Figure 2 shows this window.

For each hot spot, a "zoomed-in" window displays a map of the area, including locations of air/sea ports and road networks on which evacuees travel. These windows show several events occurring. At the beginning of the simulation, blue icons in the shape of a person appear at random places on the map, with each icon representing a group of approximately 100 evacuees. As the planes and ships approach the hot spot on the main window, they also appear on the hot spot window when they get close enough to display. This allows the user to see the behavior of the ships, planes and people groups in much greater detail as they arrive at the ports. An example of a hot spot window is shown in Figure 3.

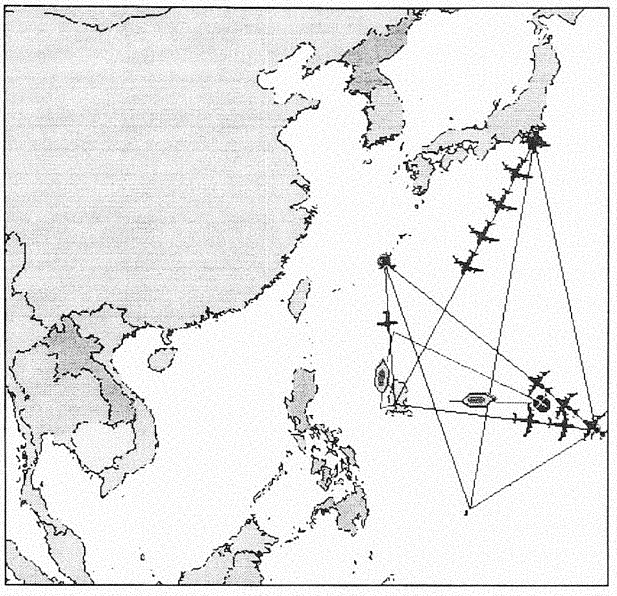

Figure 2: The Southeast Asia Window

The hot spot views can also show several other occurrences, depending on the actions specified in the plan file. Rebel groups, appearing as red icons in the shape of a person, can destroy links in a road network and also take over a port. In the latter case, the port is shown to be under hostile control by displaying a rebel flag over the port. The NEO simulation allows the planner to send in SH-60 helicopters carrying troops to regain control over the port. When this happens, a helicopter flies from one of the carriers, lands at the port, and returns to the carrier when control has been established. To signify control of the port, the rebel flag is replaced by an American flag.

Three other windows are displayed during the simulation. First, a summary window shows current counts of American citizens evacuated, number of missions generated, number of sorties generated, number of failed sorties, etc. Another window displays the current simulation time in hours and also provides a button to allow the user to pause and resume the simulation at will. Lastly, the third window allows the user to select several class attributes to display run-time statistical graphs during the simulation.

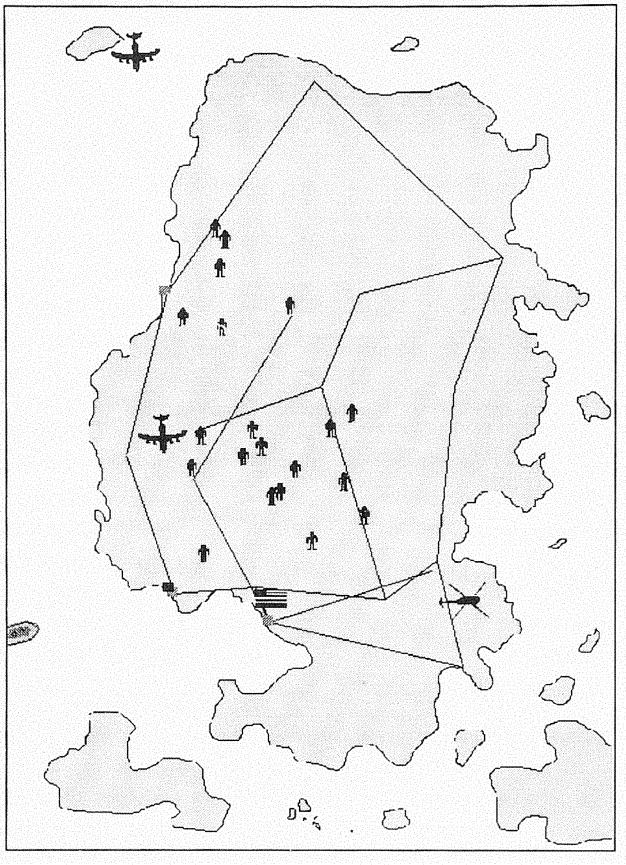

Figure 3: A Sample Hot Spot Window (Gompin)

3.5 Plan File

Since the purpose of the NEO simulation is to allow a JTF planner to determine how effective a plan will be in an actual evacuation operation, a textual plan file is used to convey to the model the types of events that will occur and the times that they will happen. There are several types of events that can be controlled through this file. For each event in the plan file, a time is specified that dictates when the event is to occur. First, the JTF object is told to perform a plan action which starts the simulation and performs any necessary initialization procedures. Next, the people in the area are told to evacuate at a specific time. At this time, each group of people determines the shortest possible path to an evacuation point and starts traveling toward that location. Also, helicopters such as the SH-60 can be told to travel to a hostile area and secure it from rebels.

The event definitions in the plan file pertaining to aircraft and ships are slightly different. These evacuation vehicles, or platforms, follow a specific pattern of departing from a set location (an airbase or somewhere at sea), traveling to an evacuation point to load passengers, transporting and unloading the passengers to a safe haven, and returning to the original location to end the mission. Therefore, these entries in the plan file specify the time that the mission will start, the number and type of platforms to use, the starting location, the evacuation point, and also the safe haven.

4 DEMONSTRATION OF THE MODEL

The NEO demonstration model was utilized at JWID '95 to model a real NEO situation. The scenario called for 4200 people to be evacuated. The two bases in the area, Yokota and Kadena, both had five C-130H aircraft available for use in the NEO. A fictional point near a popular city was used for the evacuation point in Gompin. The evacuation of the U.S. citizens on Gompin was to begin at the end of Day 4.

Two plans were used to compare and contrast to one another. The first plan attempted to evacuate the citizens using all five aircraft at each airbase, ten in all. The second plan was a more conservative one, keeping two C-130s back at each base in case there was a need somewhere else in the region, so six C-130s in all were used for the NEO mission. In both plans, the evacuation order by the JTF was given at the 96 hour mark; at that time, the people groups on Gompin moved to the port by using the shortest path. This path took into account any blocked highways and, in some cases, meant moving off the main highways for a majority of the time.

The two charts below highlight the differences between the two plans. In the first case, when utilizing all ten planes available, all evacuees were safely rescued within 50 hours of the evacuation order - about two days, as seen in Figure 4. In the second case, shown in Figure 5, the operation took a little over three days - 80 hours. This information can now be presented to whomever is coordinating the NEO in the JTF for their decision, or another plan could be created and then analyzed using this simulation.

5 POST-JWID EFFORTS

Since JWID '95, a great deal has been done to improve and enhance the NEO simulation. First, the capability has been added to allow the user to define multi-leg missions in more detail. Second, the model was enhanced to account for subsystem failure and maintenance activities on planes and ships. Third, the model was successfully converted from MODSIM to C++. The following sections describe each of these items in detail as well as the benefits realized from these modifications.

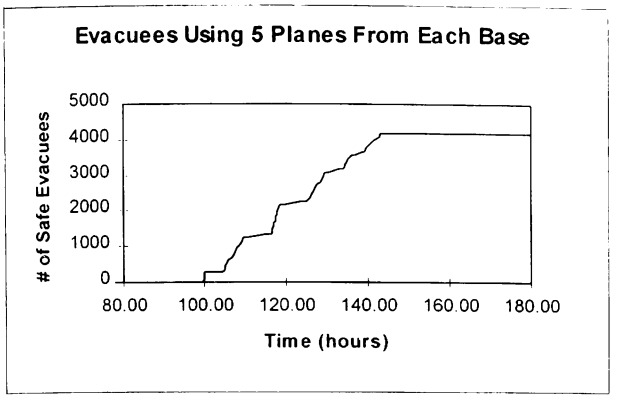

Figure 4: Time Trace of Evacuees with the First Plan

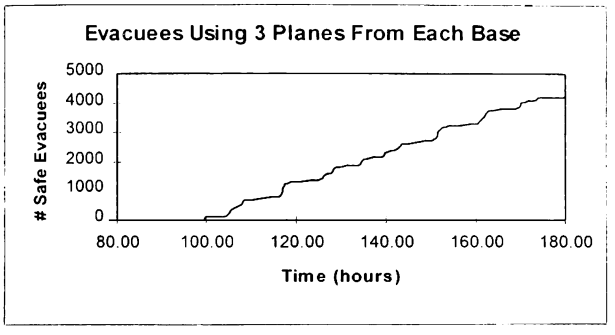

Figure 5: Time Trace of Evacuees with the Second Plan

5.1 Multi-leg Missions

Previously, any mission performed by an aircraft or ship comprised of exactly three legs: departing from a home base and arriving at an evacuation point to load evacuees, traveling to a safe haven to drop the evacuees off, and returning to the original base. With the development of a mission leg file, a JTF planner can create missions of varying lengths. For each mission, one line is entered into the file for each leg.

Several pieces of information are specified for each mission leg. One of the most important items is the name of the mission to which the leg belongs. Also, the name and quantity of the desired platform is specified. A mission leg also contains an originating location, an ending location, and an action. A relative departure time is included which specifies how long after the mission starts that the mission leg should begin. For example, if the plan file specified that a mission should begin at time 2500, and the mission contains two legs with relative departure times of 0000 and 0250, then the first leg would start at time 2500 and the second would start at 2750.

A mission leg also specifies a lead time and cancel time. The lead time specifies the amount of time needed to prepare for the leg, and the cancel time specifies the amount of time that a platform can delay departure before the leg is considered to have failed. Therefore, if the second mission leg in the previous example had a lead time of 0100 and a cancel time of 0200, the platform would start preparing for the leg at time 2650, and the leg would be considered to have failed if departure did not occur by 2950.

The main benefit of the mission leg file is the ability to define the structure of a mission as the JTF planner sees fit. The basic three-leg mission can still be used, but missions can now be made to have as many legs as desired. This provides a great deal more flexibility to the planner, which in turn greatly increases the possible options that can be simulated and evaluated. Also, since the mission leg file is read in at the beginning of the simulation, changes can be made to the file in order to see the effects on the operation without needing to recompile the model.

5.2 Platform Maintenance

An operation of any kind rarely follows a best-case scenario. Random elements and unforeseen events are always possible and even likely. Therefore, the possibility of failure and subsequent maintenance activities have been added to the NEO model. For each aircraft, a small number of highly fallible subsystems have been added. For example, the C-130 is currently comprised of five subsystems. In reality, the number of subsystems is much larger. Modeling all of these subsystems would drastically reduce the execution speed of the simulation. Therefore, the number was kept small enough to retain a high execution speed, but also large enough to show some noticeable effects of component failure.

For each subsystem, a mean time between failures (MTBF) and a mean time to repair (MTTR) are specified. At the beginning of the simulation, a failure time is determined by drawing a random number based on the MTBF. After each mission leg, the flying time is subtracted from the failure time. When the failure time reaches zero, the component has failed and the platform cannot continue on another leg until the proper maintenance has occurred. The amount of time needed for maintenance is determined by drawing a random number around the component's MTTR (Zahn et al. 1995).

The primary benefit of this modification is that the JTF planner can see how well a plan executes when random events such as component failure occur. Failure and maintenance could cause a platform to depart after the cancel time has passed, thereby generating a failed sortie. If too many sorties have failed due to required

maintenance, then the planner may need to modify the mission legs and plan files in order to specify a more conservative flying schedule.

5.3 C++ Simulation and Animation

Since JWID '95, TASC has developed its own C++ simulation engine and graphics library. With this engine, TASC has modified the IMDE system to generate C++ source code as well as MODSIM II. The original MODSIM II version of the NEO model has been ported to C++ through the use of the new simulation engine and graphics library, which was beneficial because several advantages are realized by using C++. First of all, C++ is more widely known and recognized, which makes it easier for people familiar with C++ to understand how a model is constructed. Second, C++ is a more cost effective solution since it is possible to use public domain C++ compilers. Third, modelers now have the capability of incorporating third party libraries written in C++ into their models.

6 CONCLUSION

There are many high-technology software tools available to help military advisors in their decision making. One of these tools has been detailed in this paper, a model to simulate a possible NEO task. The flexibility of the model allows the user to analyze many different COAs before settling on an optimal choice. The animation involved in the simulation facilitates the decision process by graphically illustrating the outcome of a potential plan. With the use of IMDE, the object-oriented simulation package used to model the NEO, graphical comparisons can be made between two or more candidate COAs quickly and easily.

The conversion to C++, the addition of maintenance activities to the platforms and the addition of multiple leg missions have increased the power and fidelity of the base NEO model. Future enhancements may include expanding the maintenance capabilities by adding more subsystems to each platform, modeling the possible replacement of failed aircraft, and adding the functionality to graphically define road, sea and air networks.

ACKNOWLEDGMENTS

The authors would like to thank Mr. Patrick Clark and Capt Todd Carrico for their invaluable help in the writing of this paper. The research presented was sponsored by DARPA and the Logistics Research Division, Armstrong Laboratory, U.S. Air Force, Wright-Patterson AFB, OH 45433-7604 under Contract F33615-92-D-1052, DARPA Order Number 0008.

REFERENCES

Barkakati, N. 1991. *Object-Oriented Programming in C++*, Carmel: SAMS.

CACI Products Company. 1991. *MODSIM II Reference Manual.*

Carrico, T. 1995. The Object Model Working Group Document. Logistics Research Division, Armstrong Laboratory, U.S. Air Force, Wright-Patterson AFB, Ohio.

Clark, P. K., et al. 1994. Object-oriented Simulation with IMDE. In *Western Simulation Multiconference - Object-oriented Simulation Conference '94 Proceedings*, 131-136.

Zahn, E. A., N. J. Stute, and P. K. Clark. 1995. An Object-Oriented Simulation of Air Force Support Equipment Usage. *In 1995 Winter Simulation Conference Proceedings*, 1193-1199.

AUTHOR BIOGRAPHIES

JEFFREY E. SUMNER is a Staff MTS at TASC, Inc. He received his B.S. in Computer Engineering in 1990 and his M.B.A. in Project Management in 1995, both from Wright State University. He has been involved in numerous software development projects and has worked on the IMDE simulation tool for the past three years. His interests include object-oriented software development and discrete-event simulation.

ERIC A. ZAHN is a Staff MTS at TASC, Inc. He received his B.S. in Systems and Control Engineering in 1993 and his M.S. in Systems, Control and Industrial Engineering in 1994, both from Case Western Reserve University. He has utilized various simulation packages for the last five years, and has worked on the IMDE simulation tool for the last two. His interests include discrete-event and object-oriented modeling and simulation.