Proceedings of the 1997 Winter Simulation Conference
ed. S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson

EAGLE VIEW: A SIMULATION TOOL FOR WING OPERATIONS

Eric A. Zahn
Kerris J. Renken

TASC
2555 University Blvd
Fairborn, OH 45324 U.S.A.

ABSTRACT

This paper highlights a project performed by TASC for Armstrong Laboratories, Wright-Patterson AFB. This project describes a simulation tool to help wing-level planners at an Air Force Base with analysis of operations and assist in their decision making process. The tool, named Eagle View, gives planners an “eagle-eye” view of the base and, at any time, allows a projection simulation from the current state to be run, gaining insight into the future outcomes. For example, projection simulations can be used to evaluate various plans, manpower schedules, or possible contingencies and their effects. The simulation is created with IMDE, an object-oriented simulation package developed by TASC. A demonstration prototype has been built; its construction has been detailed, and a sample scenario where the Eagle View prototype is used is presented. Future enhancements of the Eagle View model are then discussed.

1 INTRODUCTION

At the heart of any operation of a US Air Force fighter wing lies the critical decision-making process of wing-level commanders. These commanders control a whole spectrum of information and materiel used in all facets of normal operations of the wing. Some of the activities the Wing Commander must perform are to assess the ability of his wing to perform tasked missions assigned for it and identify any potential shortfalls and limiting factors that may impair the operations, like a shortage of essential supplies. He also must ensure that detailed and executable plans are developed that will allow forces to deploy to contingency locations within a limited amount of time.

Current methods of tapping into all the necessary information involve obtaining access to several different systems to obtain the overall view of the wing needed by the commander. In many cases, this information is spread out over several different databases on scattered computer systems. What is needed for the wing is an automated system that can integrate real-time data into a single, centralized application to give planners an accurate view of the current status of the base.

The wing commander is often tasked to make decisions to deploy the wing based on higher headquarters’ direction, and his ability to respond to the command is based on the current state of the base with respect to the status of the planes, supplies, and manpower.

This paper presents a demonstration of the Eagle View system. The Eagle View system is currently a prototype that represents this type of wing logistics planning tool. The Eagle View system allows planners and logisticians to have an accurate view of the current state and situations of wing operations in real-time, giving the planners an "eagle-eye view" of the base at any time. The Eagle View system also allows simulations to be performed, assessing a specific plan that can be executed in the future and analyzing the results of the plan through animation and detailed data analysis. This simulation would also be synchronized with the database so that the assessment simulation can initialize itself with the current state of the base as read in through the databases.

The discussion is broken down in to the following sections. Section 2 details the wing command and the different processes it oversees in more detail. Section 3 describes the architecture used to develop the centralized system and the different parts that make up Eagle View. Section 4 explores how Eagle View can be used in a sample scenario; and section 5 describes what future enhancements and implementations are currently being studied.

2 SYSTEM DEFINITION

The complete realm of the scenario to be modeled and analyzed is now detailed further. Normal operations of an air base wing are primarily concerned with having the aircraft present fly a specified flying schedule. For the

case under study, this wing contains 24 F-16s that are utilized to fill missions. The schedule for the wing contains information on the specific takeoff time, abort time, lead time, and cancel time for the mission. In typical operations, the airbase is notified of the day's various missions, and attempts to assign available aircraft from the idle pool to a mission at the beginning of the mission's lead time. If there are no aircraft available at this time, the mission is queued. (Zahn 1995)

Various supplies are needed from different locations around the base to support the flying operation. For instance, the aircraft need to be refueled before each mission takeoff. The fuel used by the aircraft comes from a POL building (Petroleum, Oil and Lubricants) on the base. Plans for replenishing the POL supply would also be present in the database and would be needed for the Eagle View system to track its projected level. Other such buildings around the base that contain needed resources are the AGE (Aerospace Ground Equipment) shop, Munitions, Avionics, Supply, and Engine buildings. The aircraft utilizes these resources during its pre-flight period, the routing post-flight check-up, or in an unscheduled maintenance stop where a critical system has broken and repairs are needed before the aircraft is functional again.

A main concern that arises in the wing commanders office is a change of flying schedule. Can the base, with its current levels of resources and current states of the aircraft, be capable of flying a more active schedule? The Eagle View system needs to have such a capability for the assessment of changes in plans or schedules. These schedule analyses are best served by a simulation based on the current values of all objects of interest to the model.

Another situation that a wing logistician must plan for is a change in the overall plan for the operation of the wing. One example is if the wing is asked to prepare for a deployment. This tasks involves other complex objects and logic not detailed above.

The preparation process for a wing deployment involves palletizing numerous supplies for the aircraft and placing the resulting palettes on chalks for cargo aircraft to transport them. Aircraft also must be assigned to travel on the deployment. Each resource location (POL, Munitions, etc.) must pack up a specified number of palettes for the deployment (the exact number is detailed in the plan). Forklifts then transport each palette from the resource building location to the marshaling area, where they are then lined in chalks. The transportation of the palettes is not a trivial process, as the resource buildings could be miles away from the marshaling area. These changes in plans for the wing operation can be studied through a simulation. How would the changes in the plan affect key measures of effectiveness of the base? This would include aircraft abort rates, and whether the deployment request can be fulfilled on time. Also, what about the levels of the support resources? A simulation to analyze different plans would be useful before action is taken. (Zeck undated)

Any assessment simulation in the Eagle View system must be able to read in the current state from a centralized database and initialize itself to the current value. An object-oriented simulation is best served to perform this task, as the plan changes may only affect one object in the model (like the rate in which the base POL supply is replenished). The assessment simulation must track the same state values as the real-time, current view shows, only at a faster speed. Multiple replications of any simulation must be done in order to create confidence intervals for the future values of state variables. The assessment should also present the simulation data in a fashion that is clear and easy to understand, preferably in graphical format.

2.1 Object Classes in the Scenario

There have been numerous object classes mentioned above that need to be defined in a robust manner. The normal operations of the base include objects such as aircraft, squadrons, missions and manpower. In the scenario, the main object is the Wing Operations Center. This building is the information center of the base and controls the day-to-day operations of the base. It will decide what plan the base is currently running under, whether it is normal operations or in a full-scale deployment. The Wing Operations Center has, under its domain, various buildings that serve as the supply centers for the aircraft. Ten of these resource buildings are modeled in the scenario; for example, a munitions building controls the supply of the munitions resource. These aircraft in the airbase are controlled by a squadron, who receives missions and tries to assign any available aircraft to the scheduled mission. If aircraft can not be given to the mission because of maintenance problem or lack of idle aircraft the mission is queued until aircraft are available or the abort time of the mission is reached.

Each one of these objects needs to have certain capabilities in the proposed Eagle View system. First, each object should contain its current state and output the current value to an external source at any time. It also needs to update itself whenever a new value or a change comes in from the field. For example, if aircraft #10 is idle, but then the word comes in from the maintenance crew that #10 has a broken radio, the status of Aircraft #10 in the Eagle View system changes itself from IDLE to BROKEN. These inputs will, in the future, be fed in

electronically. Each object may have a plan attached to it describing the behavior or characteristic of the object. For example, a new schedule would be a plan for the Mission Assignor object; and a new operations plan would be attached to the object representing the wing operations center.

Certain objects may be hooked up to an expert system, which contains rules on when the object should flag the analyst about its current state. A good example of an expert system usage would be for the resource buildings. The expert system specifies a level that the status color of the object turns to yellow (meaning caution) or red (signifying a dangerous level). Separate levels can be set for each of the buildings to monitor its resource level.

Finally, and most importantly, each object must contain logic and attributes to be used in assessment simulations. The assessment simulation must communicate with the current status and initialize each object to be the state of the object at this point in real time. Methods and attributes of each object must be fully robust to enable a complex, correct simulation of the plan being assessed to run.

The next section contains more detail on how these objects and their functions are implemented. The implementations may differ now in the demonstration version than the fully implemented version of Eagle View, but the current demonstration is a proof of concept for these technologies to interact together in making a powerful analysis system such as Eagle View.

3 MODEL ARCHITECTURE

The previous sections have briefly outlined the various capabilities that are involved in the Eagle View system. The run-time view of the system, assessment simulations, animations, and expert system are CPU-intensive processes that may slow down the computer system that is hosting Eagle View. That is why Eagle View separates some of the components into different processes. This architecture provides several advantages. The different processes may run on separate CPU's, either a multi-processor workstation or a different workstation, allowing the processes to run more efficiently. It also creates a greater level of modularity, by allowing the individual capabilities of Eagle View to be upgraded independent of the other parts. The rest of this section highlights the main components of the Eagle View system in more detail and briefly lays out how each one works with the other components during run-time. Figure 1 below shows how all the parts in the system work together to form the Eagle View tool.

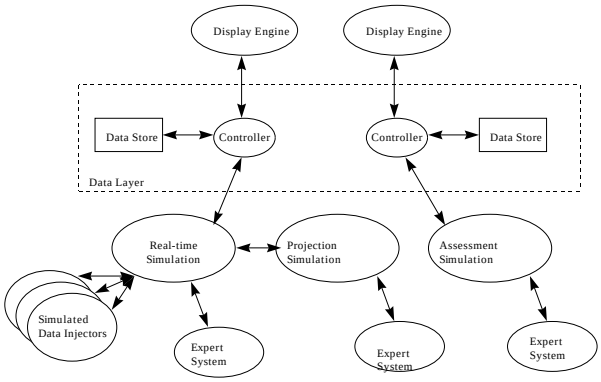

Figure 1: How the Technical Components of Eagle View Interact with One Another

3.1 Real-Time Simulation

The main component of Eagle View is the real-time look at the base. In a fully deployed system, this real-time view of the current state is constantly updated by data feeds coming into the system from different parts of the field. Incoming messages are read in, the appropriate values are updated, a warning is sent to the wing commander if the expert system rules specify, and operations continue.

However, because of current technological limitations, this signal technology does not exist yet in a practical form. A substitute for the real-time data feeds must be found, and can take the form of a simulation. A real-time simulation has been built to model the workings of the base in the absence of the feeds. The simulation must be able to interpolate the levels of state variables and possess the logic to propagate other changes in different objects, if necessary. The real-time simulation must also be able to receive data updates from the data injector, which simulates some real-time data feeds. Once the signal feeds are developed, they can replace parts of the real-time simulation in future Eagle View systems.

This real-time simulation was built by TASC using the Integrated Model Development Environment (IMDE). IMDE is a domain-independent tool used to develop object-oriented discrete event simulation models. The USAF Armstrong Laboratories Logistics Research Division has funded its development at TASC, Inc. since 1990. It is a single application running under either UNIX or Windows NT, and it 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 the arena of object-oriented technology. All model parts in the simulation are defined as "objects" that have both variables that describe the state of the object, which are called attributes, and functions that specify its behavior, which are termed methods. By storing all objects in its ODBMS, these simulation objects can be re-used in other simulations without recoding. (Sumner 1996).

There are three ways that the state variables in the objects of the real-time simulation can change. First, the simulation possesses the logic to make alterations of critical variables as stated above in the absence of external data. Secondly, the real-time simulation contains the capability to receive data from external sources. The data receivers collect the information in a generic format while the simulation is running. The data can be translated into an action (a method call) in the simulation. For purposes of the demonstration, there are a fixed set of events that can be translated. In a production system, the simulated data injectors would be replaced by real data feeds from the real-world objects. Third, a change of plans can be made by the commander. A specific plan entered into the simulation provides information on known future events and how to handle them. Plans provide a capability to change or add to the logic in the simulation without re-compilation. One example of a plan change is an adjustment of the flying schedule. More details about the last two components are seen below.

3.2 Data Injector

The data injector is used to emulate real data feeds. The injector itself is a separate simulation object that injects events into the simulation at the correct moment in time. The basic structure of the injector's functionality is shown in Figure 2.

The injector parses a file which contains events that occur during base operations. The events can be of several types. One type is to change the level of a resource; other more generic events may cause a method of some object in the simulation to be invoked. An event from the file detailing a failure in a certain aircraft would cause the injector to call the broken() method of a particular aircraft in the simulation at the dictated time. The net effect of this process is that a real-world event has just appeared to occur in the real-time simulation. The simulation is unaware of whether the event was initiated by a real data feed, an injected event, or by the underlying logic of the model.

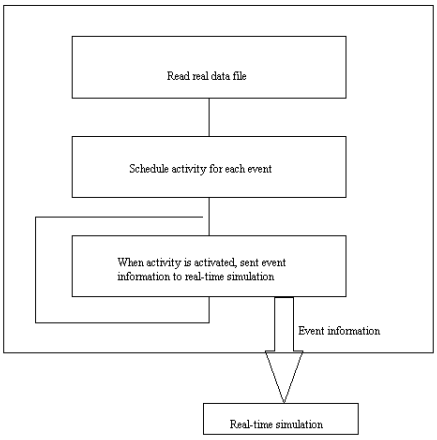

Figure 2. Data Injector Structure

3.3 Plans

During the course of the real-time simulation execution, the commander may not be pleased with the way things are going in the operation. This may be signaled by the expert system that concurrently runs with the real-time simulation; more detail about expert systems and how they work is given below. The primary way for the wing commander or logistician to alleviate any situations that arise is through the modification or addition of existing plans. New or modified plans may also be generated in attempts to improve the current process, or give a new mission directive, in the absence of any problem situations.

Entities in the real-world simulation can be selected to have either their currently executing plan or alternate plans brought up into an editor (if a plan exists), where modifications can be made. Plans are developed using an interpreted plan language developed by the Battelle, Inc. The language allows variable assignments, asynchronous event taskings, and normal process control structures (such as if-then, loops). (Winchester 1996)

At any time in the system, a plan may be analyzed and assessed. Plan additions and/or modifications are made to the system, and then are evaluated by Eagle View through an assessment simulation. The assessment simulation then gives statistical output on what effect the new plan will have on resource levels, mission accomplishment, etc. By analyzing this output, the user can determine whether the changes in the plan will be acceptable. If so, the user then presses a button and implements the new plan. The next section describes the assessment simulation and its function in more detail. (Renken 1996)

3.4 Assessment Simulation

The assessment simulation component of Eagle View provides the analyst with the capability to evaluate how a current plan, or a new/altered plan, will affect the simulation in the future. The assessment simulation runs concurrent with the real-time simulation of the base, but at a faster speed, so the user can view the ongoing changes in the status of the base while the assessment simulation is running. For this demonstration system, only one replication is performed; in a production system, many replications would be made by the assessment simulation to create the confidence intervals and statistical validity to make decisions.

The assessment simulation is started by a fork call from the real-time simulation. This fork causes a separate simulation to be generated from the current real-time simulation. The effect is to make a clone of the current simulation; thus all objects have the same state and values that the real-world simulation has at this time. A separate animation is then created for the cloned simulation, and the assessment simulation begins.

There are two important ways to view the assessment simulation and the results that are produced. First, at any time in the assessment, the user can view time traces of state variables to see what values it took on during the assessment with the new plan. These values are contrasted to the projected value of that variable if no action was taken. This compare and contrast method is automatically done through the Eagle View system and can show dramatic changes in the level of variables with the use of different plans. An example of such a graph for a resource building, comparing the differences between the current plan and the assessed plan, is shown in Figure 3. Also, the expert system component is also hooked into the assessment simulation, triggering signals when values have reached warning levels or have breached a point where action is required immediately. During the assessment, users can then look at the animation to tell if things would go wrong with the plan currently being evaluated and at what point in time these signals occur. This information can then be used to reformulate the plan and assess again. The expert system and its implementation is discussed in more detail below.

3.5 Expert System

One of the features of the Eagle view system is to alert the user when levels of a certain resource fall to dangerous quantities. In order to fully harness this capability, an expert system is employed into the simulation. An expert system is a program which is intended to model human knowledge or expertise. These systems are not necessarily linked to simulation applications; one typical use of an expert system was to have it control a manufacturing process. If the conditions of the current state warrant, the expert system would know how to recognize the problem and make minor adjustment in the process to return the state to a desired level. The knowledge used in this system, although simple, is to know when to alert commanders when resource levels fall precariously low. The system needs to know to turn a building in the animation a yellow color if there is a concern about the level, and to turn the building red if the resource levels are significantly low and will affect operations. The technological challenge involved is to seamlessly integrate the external expert system with the Eagle View system.

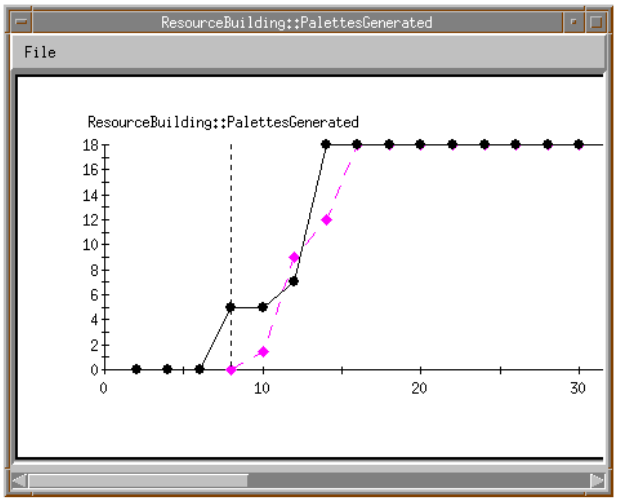

Figure 3: Status and Projections of a Sample State Variable

The expert system used with Eagle View is CLIPS (C Language Integrated Production System) (Giarratano 1993). CLIPS is an expert system tool containing the complete environment for developing the expert system - objects, facts and rules. The objects have already been created by the simulation, and there is no need for facts to be imported in, but the Eagle View system does need to know about the rules of the expert system. These rules contain the information about when to turn which specific resource building a specific color.

A sample rule in Eagle View would be to specify two levels for each resource building. The first level is the "yellow" warning, the level in which a warning message is sent and the building turns yellow, and the second level is the "alert" level, specifying at what point the building should turn red. Each resource building has its own separate set of rules. The expert system now interacts with the simulation directly by viewing the value. Based on the rules it has, the expert system will

tell the display engine if any views should change color. The engine receives that animation and changes colors of buildings, if necessary. More detail about the overall animation process of Eagle View is described below.

3.6 Display Engine

The display engine is a separate computational process that drives the animations for the Eagle View system. The advantages of having the animations of the model be on a separate processor have been outlined above: speeds of both the animation and simulation are increased, and modularity is enhanced.

The display engine contains a cache which represents the large space of state variable that is updated by the real-time simulation and, if active, any assessment simulations running. This cache is updated periodically by the display engine as it reads in new data that is fed to it by the simulation through inter-process communication. Through this data, the animation is updated and positions and statuses of objects are changed. (Renken 1996)

The animation starts with the drawing of the base layout. Icons are then associated with the entities in the simulation by connecting them through IMDE. The x-y position of the entities on the base are also defined graphically through IMDE. Figure 4 below shows the layout of the objects on the base before the simulation starts. Once the initial attachments are made between objects in the IMDE simulations and the graphical icons, the display engine updates the icons as it reads in new data from its cache. Some entities are moving objects which update their location; others just have state variables that are changed. In either case, more information about an object can be obtained by selecting its icon; an information view about that icon can be displayed, which provides detailed information about the associated entity being viewed. The next section outlines a typical scenario at an airbase and how Eagle View would be utilized to help solve operational problems as they arise.

4 EAGLE VIEW UTILIZATION SCENARIO

A description of the Eagle View system at work in a demonstration airbase is now explained. This section should show the reader how the simulation technologies outlined in Section 3 work together to support the airbase in their analysis of problems and quandaries that arise during typical operations.

The Eagle View application starts out with the real-time simulation running and its animation appearing on the screen. A flying schedule has already been loaded into the system and is being used as the current plan for the squadron. Planes are flying according to this schedule and everything seems to be running smoothly in the real-time simulation.

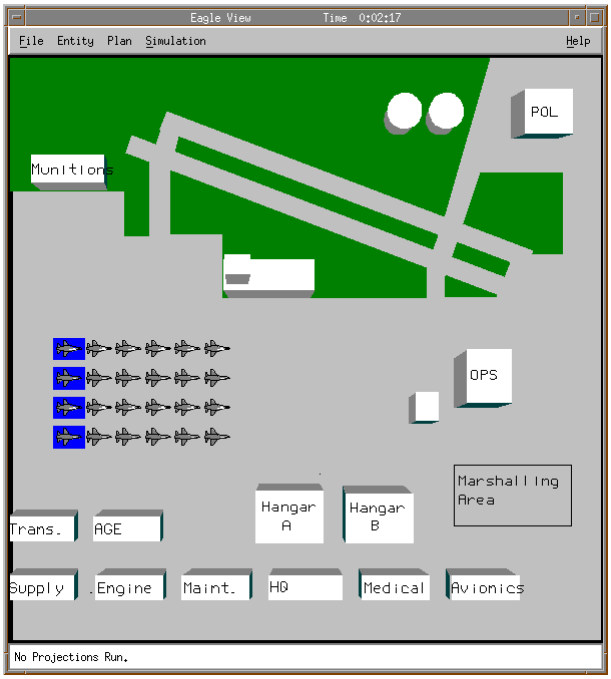

Figure 4: Layout of Objects in Eagle View

However, a problem then arises as the POL building turns yellow. The wing commander interacts with the animation and brings up the “info view” of the POL building. This view shows the respective details about the state of the building; the level has fallen to a warning level. The commander knows that the POL should be re-supplied soon; an assessment simulation is then launched from the current state. In the assessment simulation, the POL building starts out as yellow (as defined by the rules in the Expert System), but in a few days turns back to normal again, signaling the resupply of fuel. The fuel level of the POL building never reaches the critical red level; no action is taken.

The wing is then notified by its command that it needs to deploy 15 F16s for thirty days with the necessary equipment - and the wing needs to be ready for pick up in 48 hours. Can this be done? This question spawns off many processes in the Eagle View system.

First, the plan of the operations center needs to be changed from normal operations to a deployment plan. The operations center is highlighted, and a new plan is brought up to be assessed. This plan not only needs to assign aircraft for the deployment, but each of the resource buildings (POL, maintenance, supply, parts, etc.) need to pack up the necessary equipment and also

be ready in 48 hours. An assessment is then made of the plan, and the result comes back without problems. According to the assessment, the wing can take on the deployment and be ready in 48 hours. The plan for the operations center is then changed, and the new plan is implemented for the base. Buildings start to generate palettes to be loaded on for the deployment; forklifts transport them from the building to the marshaling area, and aircraft are being assigned for the trip when they become available.

However, not all problems can be forecast in the assessment. One unexpected event arises when an aircraft, which was scheduled to leave on the deployment, is found to have a failed part. This discovery is entered into the real-world simulation via the data injector; in a fully implemented Eagle view system, a live data feed would transport this message from the maintenance shop to the operations center. The failed plane turns red; the operator sees that it can not be fixed in time for the deployment, so another unassigned plane is reserved for the deployment by changing its plan. The deployment has 15 aircraft, and operations continue. A similar change to each aircraft‘s plan can be made if the issuer of the deployment command decides more or less aircraft are necessary for the deployment. These changes can be implemented while the deployment plan is still running; more assessments can be made to see whether the resulting increase of supplies can be packed up in time.

Near the end of the deployment, the marshaling are turns red, signaling a major problem with the palettes. A click on that area shows the information from the real-world simulation: the configuration of the incoming planes have changed; more chalks of shorter lengths are needed. The plan of the marshaling area needs to be changed to correspond to the need. Eagle View then assesses the new plan; the results show that the alteration can be done in time for the deadline. The new plan for the aircraft chalks is accepted. (Renken 1996)

As seen by the above example that arise during airbase operations, the Eagle View system provides instant insight in to the requirements, relations and status of the operation. When problems occur, a touch of the screen results in options or information. Plans can then be analyzed quickly and easily, and the implementation of those plans is a minimal task. Figure 5 below shows a quick snap shot of the operation of Eagle View during the deployment plan. The next section outlines what technologies can be modified and advanced to make eagle View a better analytic tool for Air Force logisticians and commanders.

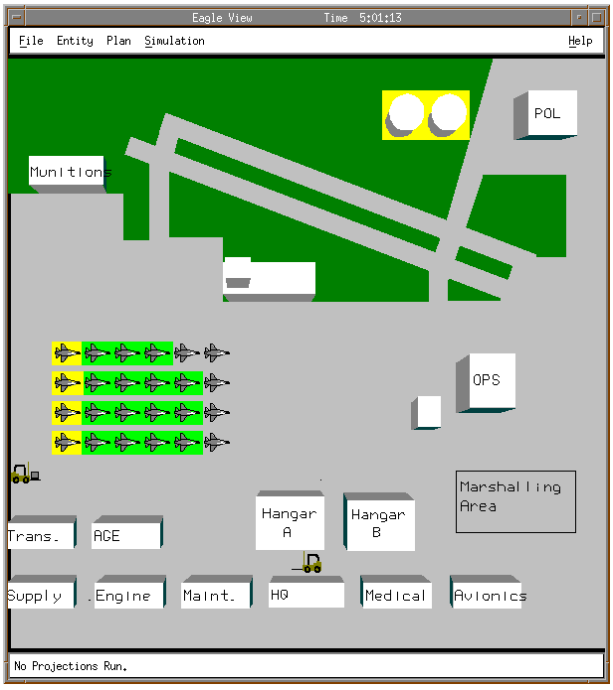

Figure 5: Screen Shot of Eagle View During Run-Time

5 FUTURE DIRECTIONS

There are many ways in which the Eagle View system can be enhanced and improved. Three directions in which improvements could be made are outlined below.

5.1 Real-time simulation

Currently, the real-time simulation operates primarily by simulating base activities in the absence of data feeds coming into the system. In a fully-implemented situation, the data feeds would drive the status of the base, and rely much less on the logic of the simulation. The interactions between the objects of the base would not be simulated but would be directly controlled by the data feeds of the object coming in to the system. This approach, to get away from the real-time simulation and depend more on the data feeds, would require a totally different kind of base model and would also necessitate an advance in the technology of how the data feeds work and their interactions with other computational and analysis tools.

5.2 Assessment simulation

As outlined above, the current Eagle View system only completes one replication when an assessment simulation is started. This is done by calling a fork process that takes the current run-time simulation, clones that, and starts the assessment using the clone as the starting point.

To do 30 simulation runs for an assessment, 30 fork calls could conceivably be made and run in parallel. However, this has not been tested as yet. More investigation is needed to examine the computational resources needed to make multiple replications of the projection simulation.

5.3 Data Injector

In the Eagle View simulation, real time data feeds are only simulated. In a production system, this simulation will need to be replaced by a process that monitors incoming data sent in by the data feeds. If the data coming can be easily translated into events in the real-time simulation, only enhancements to the model are necessary; if a data feed comes in that does not correspond to a method call, more investigation will be needed into how to handle this occurrence.

CONCLUSION

The Eagle View system presented in this paper is just a first step in assisting Wing Commanders in their decision making. With the current system, commanders can evaluate future plans based on current states and choose a course of action with more confidence. As mentioned above, there are plenty of areas for enhancement and improvement in the Eagle View system; for example, the absence of real-time data feeds from the base to the system. However, the basic technological framework is there for analysts to use in their planning of present and future wing operations.

ACKNOWLEDGEMENTS

The authors would like to thank many individuals at TASC for their technical support during the development of this project. Mr. John Platt, Mr. Neil Reams, Ms. Mary Brooks and Mr. Jeff Sumner all contributed crucial support to the Eagle View system. The authors also thank Capt Todd Carrico and 1Lt John DiPasquale, both from Armstrong Laboratories at Wright-Patterson Air Force Base, for their domain knowledge and technical assistance. Finally, Mr Pat Clark provided key assistance in the writing of this paper. This research was sponsored by the Logistics Research Division of Armstrong Laboratory, U. S. Air Force, Wright-Patterson AFB, OH, 45433-7604 under Contract No. F33615-92-D-1052.

REFERENCES

Giarratano, Joseph C. "CLIPS User's Guide". NASA Lyndon B. Johnson Space Center Information Systems Directorate, May 28, 1993.

Johnson, Bob. "Eagle View Scenario". Unpublished Paper; Armstrong Laboratory, Wright-Patterson AFB, 1996.

Renken, Kerris J. "Eagle View Final Report". TASC, Inc. Fairborn, OH. 1996.

Sumner, Jeffrey E. et. al. "A Simulation of the Evacuation of American Citizens with an Object-Oriented, Animated Model". Published in the *1996 Winter Simulation Conference Proceedings*, pp. 967-974.

Winchester, Win. "IMDE Controller/Plan File Design Document". Prepared by Battelle, Inc., Fairborn, OH, 1996.

Zahn, Eric A. et. al. "An Object-Oriented Simulation of Air Force Support Equipment Usage". Published in the *1995 Winter Simulation Conference Proceedings*, pp. 1193-1199.

Zeck, William Z., and Carrico, Todd 1Lt. "The Joint Development of Eagle View and the Deployment Capability Assessment Tool- A proposal for an In-House Research Task. Armstrong Laboratory, Logistics Research Division, Wright-Patterson AFB, Ohio. Undated.

AUTHOR BIOGRAPHIES

ERIC A. ZAHN is a Staff MTS at TASC in Fairborn, OH. 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 six years of experience in both continuous-time and discrete-event simulation, and has worked on the IMDE simulation system for the past three years. His main interests include object-oriented model development and discrete-event simulation.

KERRIS J. RENKEN is a Senior MTS at TASC Inc. in Fairborn, OH. He received his B.A. in Mathematics and Computer Science in 1989 from The University of Northern Iowa. He received his M.S. in Computer Information Systems in 1990 from The Ohio State University. He has seven years of experience in object oriented software development. He has worked on the IMDE simulation system for six years. His main interests include object-oriented software development applied to technical applications.