

DAG CE-6 Golden Nuggets Modeling and Simulation Design

NRA Task Order 76

Prepared for:



National Aeronautics and Space Administration
Ames Research Center
Moffett Field, CA 94035

Prepared by:



Micro Analysis & Design, Inc.
4949 Pearl East Circle, Suite 300
Boulder, CO 80301



CSSI, Inc. – Advanced Programs
400 Virginia Avenue SW, Suite 740
Washington, DC 20024



Seagull Technology, Inc.
1700 Dell Avenue
Campbell, CA 95008

17 October 2002

Preface

This document serves as the final deliverable for NRA Task Order (TO) number 76, “DAG CE-6 Golden Nuggets Modeling and Simulation Design.” This document compares and contrasts fast-time modeling approaches proposed by the TITAN/SRC Team.

Authors

This task was led by Micro Analysis & Design (MA&D), and supported by CSSI, Inc., and Seagull Technology. The authors of this document are, in alphabetical order:

- George Couluris, Seagull Technology
- Paul Davis, Seagull Technology
- Ken Leiden, MA&D
- Stephane Mondoloni, CSSI
- Steven Peters, MA&D

Table of Contents

<u>Section 1 - Objective</u>	1
<u>1.1 Introduction</u>	1
<u>1.2 Objective</u>	2
<u>1.3 Requirements</u>	2
<u>1.4 Organization of this Report</u>	3
<u>Section 2 - Modeling the DAG-TM CE-6 Concept</u>	3
<u>Section 3 - ATSP Model</u>	4
<u>3.1 CTO-8 Human Performance Model</u>	4
<u>3.2 Micro Saint Modeling Environment</u>	5
<u>3.3 Interoperability with the ATSP Model</u>	10
<u>Section 4 - Airspace Model</u>	12
<u>4.1 CSSI Airspace Component</u>	12
<u>4.2 Seagull Technology EDASim</u>	12
<u>Section 5 - ATM Model Approach Features</u>	13
<u>5.1 Conclusions</u>	13
<u>5.2 Recommendations for CSSI Approach</u>	13
<u>5.3 Seagull Technology Approach</u>	14
<u>Section 6 - Reference</u>	16

List of Figures

Figure 2-1 Model DAG-TM CE-6	4
Figure 3.1-1 Event handling for six sector controllers	5
Figure 3.3-1 Airspace and ATSP model interoperability.	11
Figure 3.3-2 COM to HLA Middleware.	11
Figure 3.3-3 Data interaction.	12

Section 1 - Objective

1.1 Introduction

As part of Advanced Air Transportation Technologies (AATT) project, NASA is developing and evaluating the feasibility of Distributed Air-Ground Traffic Management (DAG-TM) Concept Element (CE) 6. This concept element, titled En Route Trajectory Negotiation, delivers benefits via integration of ground-based decision support tools (DST), the aeronautical data link system (ADLS), and the airborne flight management system (FMS).

Concept Element 6: En Route Trajectory Negotiation for User-preferred Separation Assurance and Local Traffic Flow Management (TFM) Conformance. CE-6 permits automated negotiation of user-preferred trajectory changes for separation assurance and meeting of TFM constraints. Common information is automatically shared between the users and the air traffic service provider (ATSP). This concept integrates air and ground-based DST capabilities through data link to reduce ATSP workload and facilitate user-preferred trajectories in the presence of traffic and TFM constraints.

CE-6 enables a set of capacity-enhancing mechanisms, known as the DAG “golden nuggets”, beyond trajectory negotiation enabled by user-preferred trajectory selection. The expected benefits to the en route environment focus on enhancements to operations enabled through integration of controller DST, air/ground communication, and flight deck automation. DAG CE-6 provides a capability for trajectory negotiation between the ATSP and users of the National Airspace System (NAS). Three golden nuggets have been identified:

1. **Basic data exchange.** Basic data exchange has been investigated to Technical Readiness Level four (TRL-4). Through the exchange of data between the air and ground, improvements in the accuracy of airborne and ground-based decision support tools are possible. This mechanism allows operations at a higher level of performance due to improved DST prediction accuracy.
2. **Integration of controller DST with ADLS.** This golden nugget provides improved DST performance based on better knowledge of clearances being issued and the reduced controller workload based on automated message composition and loading. The integration of controller DSTs with ADLS allows advisories produced by the DST to be automatically converted into messages for transmission to the aircraft. Not only does this reduce the likelihood of transcription errors, it reduces the controller workload necessary for transmitting these messages. Furthermore, the automatic updating of the DST with clearances sent to the flight deck can ensure that the DST is operating on the best level of intent information.
3. **Integration of controller DST with FMS using ADLS.** The final golden nugget furthers the effects gained from the previous golden nugget and improves flight operations based on the capability to implement complex clearances. It allows the flight deck to receive clearances and instructions that are automatically loaded

into the FMS. While the pilot may still need to accept these clearances, this interaction allows a reduction in errors due to transcription and execution of the clearances by the flight deck. A more significant benefit is likely to be the ensuing reduction in controller workload stemming from the ability to deliver complex clearances as a single uplink rather than a collection of instructions. These complex clearances can further improve situations by allowing more precise execution of instructions than currently encountered.

The expected benefits of the DAG golden nuggets will be achieved through increased controller productivity and increased sector throughput.

1.2 Objective

The objective of this effort has been to formulate viable approaches for the near-future fast-time modeling and simulation for the DAG CE-6 golden nuggets, specifically items 2 and 3 above (the first golden nugget has been previously investigated.)

A main focus of this task has been on ensuring that the latter two golden nuggets can be modeled to adequately understand their significance and influence on controller workload and sector throughput. The modeling effort emphasized in this study is the simulation interoperability issues between existing simulation components. This objective has also been balanced by potential schedule and budget constraints.

1.3 Requirements

The modeling and simulation capability described in this report must be capable of investigating a broad set of conditions typically beyond the scope of traditional human-in-the-loop simulation related to air traffic flow situations. These conditions include the following:

- Arrival delays absorbed through speed control, vectoring as well as holding
- Conflicts in a range of traffic densities from light to heavy including conflicts between all combinations of arrival and overflight
- Automation capabilities including no automation (manual), trial planning and en-route descent advisor (EDA) capabilities
- Aircraft equipage can include data link communication (not integrated with FMS) as well as flights integrating FMS with data link
- A variety of weather conditions from nominal to severe
- Other off-nominal problems such as controller and pilot blunders
- Inclusion of real-world system delays and errors attributable to:
 - communication transaction times and errors
 - trajectory prediction errors
 - aircraft track position and velocity
 - winds aloft
 - weight and aircraft performance models
 - intent

Additional requirements on the simulation include the ability to model a minimum of three sectors and the coordination between two or more ATSP facilities. The approach must model the controller workload, critical en-route airspace problems and flight trajectories.

1.4 Organization of this Report

The software architecture for modeling and simulation of the DAG CE-6 is the focus of this effort. In Section 2, the need for interoperability between the simulation component representing the Air Traffic Service Provided (ATSP) and the simulation components representing the airspace, aircraft, and DST are discussed.

Section 3 describes the MA&D ATSP model that will be used to simulate the human performance characteristics and workload constraints of Air Route Traffic Control Center (ARTCC) controllers. The Micro Saint model of ATSP operations is presented.

Section 4 introduces competing airspace modeling approaches proposed by CSSI and Seagull Technology. ***Due to the proprietary concerns, each approach has been placed in separate Client Private appendices.***

In Section 5, the modeling approaches are compared and contrasted. The focus of this section is the simulation interoperability issues between the ATSP model and the airspace/aircraft/DST model.

Finally, Section 6 is a reference list.

Section 2 - Modeling the DAG-TM CE-6 Concept

Distributed Air-Ground Traffic Management (DAG-TM) is a proposed concept for gate-to-gate national airspace system (NAS) operations in which flight deck crews, air traffic service providers (ATSP) and aeronautical operational control (AOC) facilities use distributed decision-making. The goal of DAG-TM is to enable user preferences, provide greater flexibility and efficiency, and increase system capacity, while meeting air traffic management (ATM) requirements and without adversely affecting system safety or restricting user accessibility to the NAS. It will address dynamic NAS constraints due to bad weather, airspace/airport congestion (requiring metering/spacing of traffic), and Special Use Airspace (SUA). DAG-TM will be accomplished with a human-centered operational paradigm enabled by procedural and technological innovations. CE-6 is the en route component of this larger concept.

Modeling and simulation will be an invaluable tool in the analysis of both CE-6 and other concept elements within DAG-TM. The modeling and simulation effort to be used for this task will consist of two simulations. The first simulation, referred to in this report as the ***airspace model***, contains components for:

1. En route and transition airspace
2. Aircraft trajectories
3. Decision support tool for integrated conflict detection, resolution, and flow rate conformance

The second simulation, referred to in this report as the ***ATSP model***, will simulate ATSP tasks, roles, responsibilities, procedures and corresponding workload.

Together, these two simulations will interoperate to allow for fast-time simulation and analysis of the DAG CE-6 golden nuggets. The airspace model will have the capability

to capture the effect of improved prediction accuracy of trajectories due to the 3rd golden nugget (i.e., complex clearances auto-loaded into the FMS). The ATSP model will have the capability to measure workload reduction due to integration of DST and ADLS for both simple and complex clearances.



Figure 2-1 Model DAG-TM CE-6.

The following two sections will discuss the Airspace and ATSP models and how they will be configured to interoperate.

Section 3 - ATSP Model

The starting point for the ATSP model will be the human performance model developed for Air Traffic Management System Development and Integration (ATMSDI) CTO-8. Micro Saint, a discrete event simulation tool, will be used as the modeling environment for the ATSP model. (Section 3.2 more fully describes Micro Saint.)

3.1 CTO-8 Human Performance Model

In order to have the simulated controllers be presented with and interact with all of the types of events of concern for studying the CE-6 golden nuggets, controllers for six (6) sectors will be modeled. Each sector will have a corresponding R-side and D-side controller.

Via the task of situation monitoring, controllers in the CTO-8 human performance model must detect four events, namely:

- Handoff initiation
- Top of descent
- Metering violation
- Conflicts

Once a controller is presented with one of these events, he must take the appropriate control action, called respectively:

- Handoff
- Descent clearance
- Metering conformance
- Conflict resolution

Shown in Figure 3.1-1 is a task network diagram that implements controller operations for six sectors. Each of the four event types is shown in rectangles that represent sub-networks. Each sub-network contains its own task network diagram.

Network 3000 CTO 8 EDA

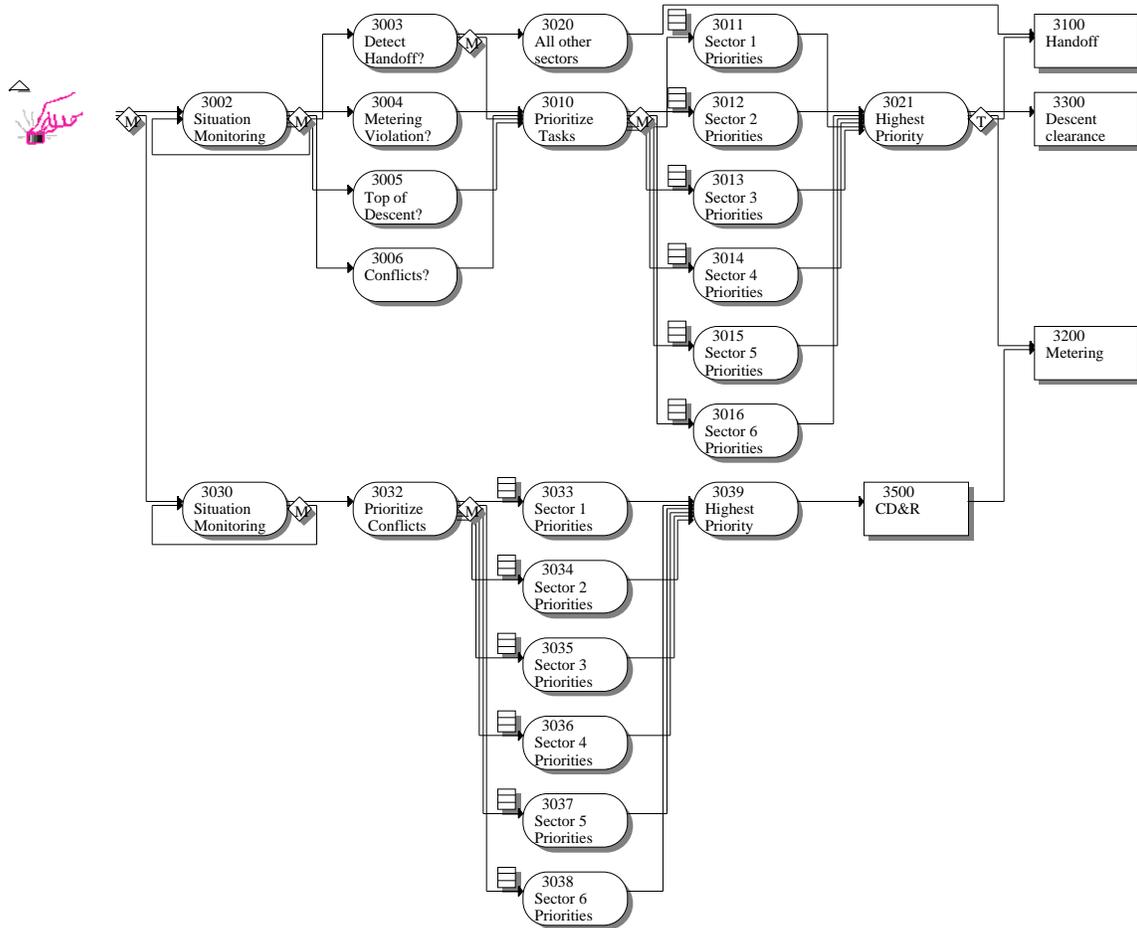
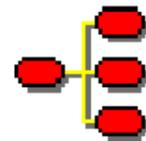


Figure 3.1-1 Event handling for six sector controllers.

Each of the event types that will be encountered by the controllers is modeled. The task network model of controller operations to handle each event describes the sequence of tasks, the time to perform each task, and the conditions that affect the controller actions.

3.2 *Micro Saint Modeling Environment*

Micro Saint is a discrete event simulation engine and runs under the Windows operating system. With Micro Saint models, users can gain useful information about processes that might be too expensive or time-



consuming to test in the real world. Some common application areas for simulation modeling include the following:

- Modeling manufacturing processes, such as production lines, to examine resource utilization, efficiency, and cost.
- Modeling transportation systems to examine issues such as scheduling and resource requirements.
- Modeling service systems to optimize procedures, staffing and other logistical considerations.
- Modeling training systems and their effectiveness over time.
- Modeling human operator performance and interaction under changing conditions.

The following subsections briefly describe various capabilities contained within Micro Saint.

3.2.1 Task Network

The user defines the flow of the process through a network of activities. Defined in this network are the activities the system must perform, the basic sequence of activities, the conditions that must be met before an activity can be performed, and the effect of activity performance on other activities and aspects of the process.

3.2.2 Process Logic

Entities (e.g. controllers, aircraft, pilots) flow through the system in many complex ways. In Micro Saint, the user can define the set of decision rules for their specific process. Through Micro Saint's own C-like language, the user defines the system's behavior without ever leaving Micro Saint. Micro Saint logic can be redefined as the model is running, to refine the model or to experiment with process improvements.

3.2.3 Queues

Queues are one of the most important building blocks of discrete-event simulation. Automatic queue data collection collects data on the maximum length of the queue, the minimum length of the queue and the wait time. By using Micro Saint's built-in tools, the user can perform detailed statistical analysis on the queues.

3.2.4 Resources

Resources can be used to quickly create a new resource variable and allocate a resource to tasks in the model.

3.2.5 Animation

Micro Saint offers two ways to animate a model. First, the task network diagram comes to life when you run your model. As each task is activated it is highlighted, thus indicating network flow and activity. You have a choice of symbols as well as task ovals so you can make this view more relevant to your application. The second type of animation, ActionView, allows the user to create realistic quality animations. With ActionView, background scenes can be scanned in, brought in from a CAD drawing or other paint packages. The user can place icons on the background and have them move in a manner they can represent the process. Micro Saint contains a library of over 500

icons and additional ones can be created by the user. ActionView animation is an excellent way to present simulation models in a quick and effective manner.

3.2.6 Interoperability

Micro Saint can be coupled with a module called Component Object Model (COM) Services, a model for binary code developed by Microsoft. With COM Services, interoperability can be set up between Micro Saint and another application. A software application can receive variable and event queue data from Micro Saint along with messages indicating the status of model execution. Developing the structure for communication between Micro Saint and other software applications can be done in any programming environment that is COM-compliant, such as Visual C++ or Visual Basic.

COM Services includes the capability to:

- Pass variable values into and out of Micro Saint during model execution.
- Insert scenario events into the event queue at specified times in the future.
- Pause, halt, and abort the model through parsed expressions.
- Send messages to the user when a model has ended or if errors have occurred.

Following is a list of the COM routines available in Micro Saint. A description of the function of each routine is provided along with the variables associated with that routine.

COM to Micro Saint Routine	Purpose
long AllNextQueue(char *)	Return the events for the next scheduled clock time. This includes the event type (i.e. clock, task) and any other information involved in the event such as tag, and task name if available.
long AllQueue(char *)	Return the events in the Event Queue from the current clock time to the end of model execution. This includes the event type (i.e. clock, task) and any other information involved in the event such as tag, and task name if available.
long BringModelInFocus(char *modelToBringInFocus)	When more than one model is loaded, select the model to be active.
long CheckModelSyntax(char *modelToCheck)	Perform a syntax check of the active model.
long CloseModel()	Close the active model.
long Connect(char *comProgramName)	Connect the application to Micro Saint
long Disconnect()	Disconnect the application from Micro Saint
long DisplayActionView()	Display Action View.

COM to Micro Saint Routine	Purpose
long ExecuteModel(char *modelToExecute)	Execute (start) the active model
long ExecuteOneClockTeg(statefloat untilClock)	Execute the active model until the clock reaches the time specified by the user. Then returns the current time and the time of the next event.
long HaltModel()	Halt the active model, if it is running.
long InsertIntoQueue(char *expr, short firstOrLastAtTime, float timeOffset, float eventTime = 0.0)	Insert an event into the Event Queue at a user specified location.
long JustNextQueue(char *)	To report information regarding the next event in the Event Queue. This includes the event type (i.e. clock, task) and other information involved in the event such as tag, and task name if available.
long OpenModel(char *modelToOpen)	Open a model in Micro Saint
long PauseModel()	Pause the active model, if it is running.
void Quit()	Shut down Micro Saint
long ReceiveOnlyConnect (char *comProgramName)	Connect to Micro Saint in a mode where it can only receive messages. Micro Saint will not send messages.
long ResetEventQueue();	Reset the Event Queue.
long SendExpression(char *exprToSend).	Send an expression to the Execution Monitor.
long SetCommandLineSnapshots(long onOff)	Turn collection of snapshots on or off and automatically overwrite existing snapshot files with the same names.
long SetFloatArrayVariable(char *variableToSet, float newValue, char *indexes)	Set a variable in a floating point array to a value
long SetFloatVariable(char *variableToSet, float newValue)	Set a floating point variable to a value.
long SetIntArrayVariable(char *variableToSet, long newValue, char *indexes)	Set a variable in an integer array to a value.
long SetIntVariable(char *variableToSet, long newValue)	Set an integer variable to a value.

COM to Micro Saint Routine	Purpose
long SetQueueSwitch(long onOff);	Set queue data collection on or off.
long SetRandomSeed(long randomSeed)	Set the random seed to an initial value.
long SetResourceSwitch(long onOff)	Set resources on or off.
long SetShowMSaint(long showFlag)	Set the display of Micro Saint to be on or off.
long SetSnapSwitch(long onOff)	Set snapshots data collection of variables on or off.
long SetStandardSwitch(long onOff)	Set standard deviations to zero, on or off.
long SetTaskDataSwitch(long onOff)	Set task data collection on or off.
long SetTimesToRun(long timesToRun)	Set the number of times to run the model.
long SetTraceSwitch(long onOff)	Set trace data collection on or off.
long SingleStepModel()	Single step the active model. Causes the next event in the Event Queue to execute.
long StopQueue()	Stop Micro Saint from returning events in the Event Queue.

The following routines are used to send data and the status of execution from Micro Saint to the COM software application:

Micro Saint to COM Routine	Purpose
long ModelCompleted()	Indicate that model execution has completed.
long ReceiveQueue(char * eventString)	Return the value in the Event Queue.
long MicroSaintShutdown()	Indicate that Micro Saint closed.
long NextClockIncrement(double currentClockValue, double nextClockValue)	Indicate that the model has run to a time set by the user.
long ReceiveIntVariable(char * variableToReceive, long newValue)	Obtain the value of an integer variable.
long ReceiveFloatVariable(char * variableToReceive, double newValue)	Obtain the value of a real variable.
long ReceiveIntArrayVariable(char * variableToReceive, long newValue, char * indexes)	Obtain the value of an array of integer variables.

Micro Saint to COM Routine	Purpose
long ReceiveFloatArrayVariable(LPCTSTR variableToReceive, double newValue, LPCTSTR indexes)	Obtain the value of an array of floating point variables.
long SingleStepDone()	Be informed when the single step is completed.

3.2.7 *Size of Models*

The size of a model is constrained only by computer memory. With systems currently available, processes have been modeled with tens of thousands of activities simulating millions of entities flowing through a process. Micro Saint is only limited by the power of the computer.

3.2.8 *Power and Flexibility*

Micro Saint models are built in a C-like programming environment that gives it the power of a programming language. Anywhere in Micro Saint that you can enter a numeric value, you can enter an algorithm consisting of a sequence of algebraic and logical expressions that calculate that value.

Operators include: () = + / * := ==

Logical operators include: if...then...else, while...do

3.3 *Interoperability with the ATSP Model*

An Airspace model will stimulate the ATSP model and they will interact as airspace events and corresponding controller actions occur. That is, a model of the DST and associated aircraft trajectories will be simulated and will present events for the sector controllers to act on. As a sector controller determines an action to take to deal with the event, the aircraft trajectory model will be informed of the requested action and institute the action. In simpler terms, one model will simulate the airspace and the aircraft flying in it. The other model will simulate the controllers acting on events presented to it by the airspace model. For example, as an event such as a conflict detection occurs, the controller model will be notified of the conflict along with an EDA conflict resolution advisory, it will then determine the time for the controller to issue the conflict resolution back to the aircraft trajectory model, and finally the aircraft will implement the corrective action thus avoiding the conflict.

This type of interaction between the Airspace model and the ATSP model will take place by allowing the two simulations to interoperate. The resulting simulation system will allow for a fast-time simulation tool for analysis of the golden nuggets. Figure 3.3-1 shows the basic architecture that will allow the two simulations to communicate. Also contained within the communication of data between the models is the management of time. That is, both simulations have a sense of time and must coordinate the incrementing of time between them.

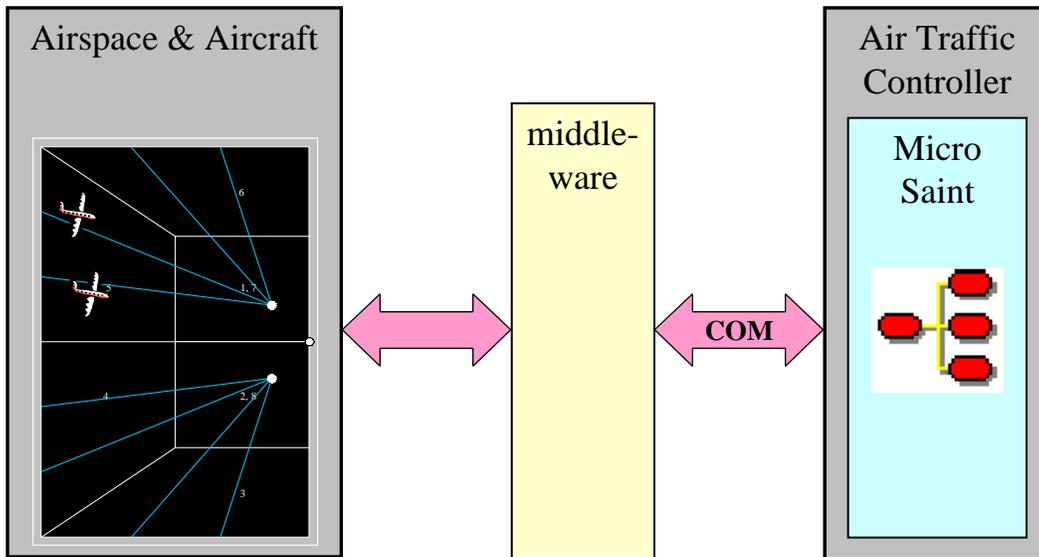


Figure 3.3-1 Airspace and ATSP model interoperability.

Micro Saint already includes the ability to send and receive data via a COM interface. (With a minimum of effort – about a man-week – an interface using sockets could also be implemented in Micro Saint.)

MA&D has developed a middleware that communicates to Micro Saint using the COM protocol and to other simulations using the High Level Architecture (HLA). (See Figure 3.3-2) It currently implements the Defense Modeling and Simulation (DMSO) Run Time Infrastructure (RTI) 1.3ng-V6. (Note that as of 30 September 2002 DMSO no longer supports or distributes an RTI. Presently only private vendors will an RTI.)

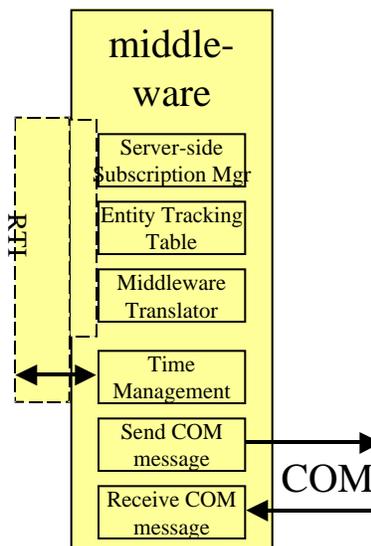


Figure 3.3-2 COM to HLA Middleware.

A third-party vendor, MäK Technologies, supplies one piece of the middleware labeled VR-Link in Figure 3.3-2. This provides software routines that decode the RTI messages into a format easily usable by the middleware. A single license to use the software costs about \$2,000.

The interaction of data between the two models requires a mapping of data parameters and agreed upon packaging of data into functional structures. Figure 3.3-3 simply shows the transfer of data between the two models via the middleware.

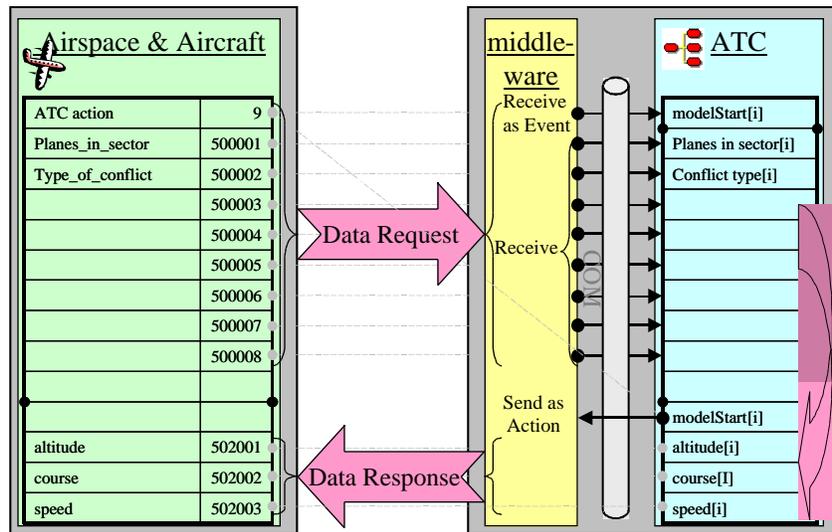


Figure 3.3-3 Data interaction.

Section 4 - Airspace Model

Two basic architectures are presented for developing a fast-time modeling approach suitable for modeling the DAG CE-6 golden nuggets. Both approaches connect the ATSP model developed by MA&D and discussed in Section 2 with an Airspace model. Each of these approaches is discussed separately in client private appendices.

4.1 CSSI Airspace Component

CSSI proposes an approach to *rapidly* develop the capability to investigate the golden nuggets by integrating components capable of performing specific functions required to support the analysis of each golden nugget. The integrated components would be integrated with an executive that would interact with Micro Saint via COM Services calls. This approach allows the development of investigations and results within a *short time horizon* at *relatively low risk*.

The appendix titled "Airspace Component" describes the airspace modeling approach proposed by CSSI.

4.2 Seagull Technology EDASim

Seagull proposes to implement the En route and Descent Advisor Simulation (EDASim) to evaluate aeronautical data link (ADL) applications within the framework of the DAG CE-6 En Route Trajectory Negotiation. EDASim is planned as an independent, down-scaled derivate of the Airspace Concept Evaluation System (ACES) currently under development in CTO7 [1]. ACES is a National Airspace System (NAS)-wide fast-time computerized simulation with advanced modeling capabilities. It has a high-powered,

comprehensive trajectory-based model of NAS-wide operations, intended to support trade-off studies of system level concepts. ACES provides:

- agent-based modeling of ATSP and traffic flow management (TFM) functions for air route traffic control (ARTCC),
- terminal radar control (TRACON) and
- airport operations.

The ACES architecture is highly flexible and adaptable and allows for the substitution of new models and models with different levels of fidelity (even dynamically).

The appendix titled "Simulation Interoperability between Micro Saint and EDASim" describes the airspace modeling approach proposed by Seagull Technology.

Section 5 - ATM Model Approach Features

5.1 Conclusions

Two independent approaches have been formulated to provide a near-term capability to NASA for modeling and simulation of the DAG CE-6 golden nuggets.

CSSI and Seagull Technology, each in conjunction with Micro Analysis & Design, have proposed approaches for providing interoperability between their model of the airspace with the ATSP model developed by MA&D. Both propose to use existing modeling capabilities and modify them to meet the needs of this effort.

5.2 Recommendations for CSSI Approach

CSSI has demonstrated an outstanding display of understanding the issues surrounding the DAG CE-6 golden nuggets and the simulation capabilities needed to perform this task. Included in their appendix are detailed descriptions of each of the model components and their interactions needed to perform this task including the interactions with the HPM model such as data and timing coordination issues.

The focus of this approach is to re-use previously developed component models, modify them where needed, and implement a simple message-based interaction capability rather than using a more complex "architecture." While some components may need adjustments to meet the requirements of this effort, the main task becomes an integration effort. The intent is to use a simplified approach that will allow for development in a short time with low risk.

5.2.1 Advantages

- CSSI's approach is to re-use software components that have already been developed, effectively turning this into an integration effort. In their approach, they have gone to great lengths to investigate and describe the situations required to fulfill the analysis of the DAG CE-6 Golden Nuggets. This has included documenting the airspace model components, the component interactions, and the message interactions with the human performance model of the air traffic controllers.

- Provides a modeling solution to specifically address the issues required in this proposal. The effort is efficiently designed to meet the requirements presented by the DAG CD-6 golden nuggets.
- CSSI will be developing the middleware data interaction capability thereby freeing up MA&D to focus on development of the ATSP model.

5.2.2 Disadvantages

- The proposed modeling solution does not provide a robust architecture easily expandable to future efforts.
- A visualization system and a data collection are not included since they are not listed requirements but they could easily be added.

5.3 Seagull Technology Approach

Seagull proposes to use EDASim, a down-scaled derivate of ACES, to leverage the verifying levels of fidelity and high level of configurability. By using ACES and its agent-based infrastructure, a highly flexible and adaptable system becomes available for this effort.

Seagull proposes two integration options, both using HLA for communication. The first option is to develop a C++ version of the EDASim Agent Infrastructure and integrate it with MA&D's existing C++ middleware. The second option is to reuse the existing Java EDASim Agent Infrastructure along with components of the existing C++ middleware and develop a new mixed-language version of the middleware. Neither option would require any modifications to the Micro Saint application.

Option 1.

- Reuse a large part of the existing MA&D C++ middleware
- Develop a C++ version of the Agent Infrastructure for the existing middleware which will
 - Package agent events and replies into HLA interactions
 - Create discrete time management functionality that is consistent with EDASim
- Adapt the Micro Saint middleware to use and interact properly with the EDASim Federation Object Model (FOM)

Option 2.

- Develop a new middleware application reusing parts of the existing MA&D C++ middleware
- Mix Java and C++ in the middleware application using the Java Native Interface (JNI)
- Reuse the existing Java EDASim Agent Infrastructure

5.3.1 Advantages

- Development using a series of three builds provides risk management in the event of reduced funding. While the completion of all three builds is necessary to provide a complete analysis of the golden nuggets of CE-6, some capabilities are still guaranteed if less than that becomes a reality.
- Builds off of existing MA&D developed middleware that provides data communication between the COM connection used by Micro Saint and time managed HLA.
- Uses HLA time management, a proven technology supported by the DoD.
- Includes a Simulation Control and Visualization Tool that provides configuration, initialization, and scenario control. The tool also provides visualization and monitoring capabilities during run-time.
- Includes an HLA commercial off the shelf tool for performing collection of data passed between the Airspace and ATSP (i.e., the Micro Saint model) federates. That is, data passed between the Airspace and ATSP models can be easily recorded. This capability will be useful during development and integration of the two models. However, data values exchanged within a federate are not accessible to this tool and this capability will have to be added, if needed.

5.3.2 Disadvantages

- ACES is currently under development in CTO7 and is the basis for developing EDASim. This presents a schedule risk.
- Since EDASim will be derived from an initial build of ACES improvements made during subsequent CTO7 builds and bugs that have been fixed will need to be migrated to EDASim. Coordination between the two programs will be necessary.
- EDASim and the Agent Infrastructure are written in Java while Micro Saint is written in C/C++. This will present some software development issues, the exact nature of which will depend on the development option selected.
- The use of HLA and its time management capabilities will require developers to be well versed in the use of the technology. Included in this is the need for the use of MäK Technologies VR-Link in the support of HLA data packet communication used by the Micro Saint middleware. VR-Link has an approximately \$2,000 cost and a \$500 annual support/maintenance fee.
- The middleware previously developed to interact with Micro Saint and HLA uses a Real-time Platform Reference Federated Object Model (RPR-FOM). This FOM is not supported by ACES and thus the use of a FOM by the middleware that accommodates the ACES architecture must be re-developed. For Option 1, a large effort will be required to adapt the Micro Saint Middleware to use and interact properly with the EDASim FOM. In Option 2, work must be in both Java and C/C++ and the Java Native Interface (JNI) must be used to develop an EDASim Agent Infrastructure. However, some of the original middleware will be able to be reused in either case.
- MA&D will be involved in the modification of the middleware it has developed to get it to conform to the Agent Infrastructure needed by EDASim. This MA&D

involvement may reduce the level of effort spent on developing the human performance model of ATSP.

Section 6 - Reference

1. Hunter, G., "ACES: Build 1 Modeling," VAMS 2nd TIM, Aug. 2002.
2. "Research Plan for Distributed Air/Ground Traffic Management (DAG-TM)", Version 1.1, October 6, 1999.