MODELING ARCHITECTURE TO SUPPORT GOAL ORIENTED HUMAN PERFORMANCE

Susan Archer and Nils LaVine
Micro Analysis and Design, Inc.
Boulder, Colorado

Abstract

Traditional human performance models have often been criticized for failing to represent and predict goal-oriented behaviors, and for failing to predict measures that are meaningful to other training and equipment simulations. To address this criticism, in 1999 the Air Force Human Research Laboratory began an effort to develop a human performance modeling environment that could interact with other simulations using an HLA-compatible protocol. One element of that environment is a model development tool that enables users to create a detailed simulation of a goal-oriented human agent, operating in a complex environment. In this context, the simulation predicts what the human is likely to do next based on the currently relevant goals, and on the status of other parallel simulations. A practical example is a combat pilot who has a primary mission to conduct reconnaissance of a target area. Therefore, the pilot's original goal is to fly a well-defined path and to use a variety of sensors to collect data. However, if during that flight the pilot identifies an incoming threat (from a parallel radar simulation), the goal will change immediately to "evade and survive." This dictates a change in tasks as the pilot suspends his execution of the pre-planned flight path and begins new tasks to dump chaff and to conduct high-speed maneuvers.

This is an extremely dynamic and demanding modeling challenge, because goal states change based on events in the scenario as well as on occurrences experienced by agents in other linked simulations. For this reason, they cannot be scripted. The problem is also complicated by the interaction between goals, in which a high priority goal can suspend, halt, or restart a lower priority goal. This must be accomplished with as little burden on the user as possible through the automatic exchange of data and the implementation of sophisticated algorithms to mediate competition between active goals.

Biographical Sketch:

Susan Archer is the Director of Operations at Micro Analysis & Design, Inc. In addition to her responsibilities in that capacity, she is also the program manager and technical lead of the Army Research Laboratory's Improved Manpower and Personnel Integration Tool (IMPRINT) effort. This is a software development effort to develop an advanced integrated Manpower, Personnel and Training (MPT) analysis tool for the Army. She was also the Program Manager for HARDMAN III, the predecessor to IMPRINT. Ms. Archer manages several contracts involving Human Systems Integration (HSI) efforts associated with advanced command and control concepts.

Nils LaVine is a Principal Systems Engineer at Micro Analysis & Design, Inc. He is the Program Manager for the Combat Automation Requirements Testbed (CART) project, as well as for several projects that predict the effects of Nuclear, Biological and Chemical (NBC) agents on human performance. He has been instrumental in linking the Micro Saint discrete event human performance modeling engine to other simulation environments.

MODELING ARCHITECTURE TO SUPPORT GOAL ORIENTATED HUMAN PERFORMANCE

Susan Archer and Nils LaVine
Micro Analysis and Design, Inc.
Boulder, Colorado

INTRODUCTION

The Air Force Research Laboratory/Human Effectiveness Directorate (AFRL/HECI) supports research development and acquisition (RD&A) for the Air Force in the areas of human/system design. AFRL/HECI is leading the development of a methodology and a suite of computer simulation tools that will be used to evaluate crew system and cockpit design for determining the impact of human factors and performance issues on the requirements generation process. The software tools from this effort will allow users (e.g. SPOs, labs, etc.) to impact design decisions very early in the requirements generation process in ways that historically have only happened in the design phase of the system procurement process. Being able to affect design much earlier than was previously possible will allow for better decisions to be made with less cost. The simulation tools being developed for this effort are called the Combat Automation Requirements Testbed (CART) Human Performance Modeling (HPM) Environment software.

The CART HPM Environment consists of an expanded version of the Army Research Laboratory (ARL) Human Research and Engineering Directorate (HRED)-developed Improved Performance Research Integration Tool (IMPRINT) (see Allender et. al., 1995). IMPRINT is Government-owned software and consists of a set of automated aids to assist analysts in conducting human performance analyses. IMPRINT provides the means for estimating manpower, personnel, and training (MPT) requirements and constraints for new weapon systems very early in the acquisition process. The CART HPM Environment expansions to IMPRINT mainly consist of adding the capability for inter-model/simulation communication via the High Level Architecture (HLA) Runtime Infrastructure (RTI) and the addition of a goal orientation capability for simulating human performance. This paper discusses both of these new capabilities.

The Need for CART

The purpose of the CART HPM Environment effort is to provide a modeling environment in which human performance models can be developed that are able to interact with engineering level, constructive, and first principle human performance models. This will help analysts more fully understand the implication of human performance on total system performance, and to use that understanding to impact system requirements and to assess future training needs. The intent of CART is to enable users to affect system design prior to Milestone 1 of the acquisition process.

Currently, human performance and engineering level simulations are introduced in the acquisition process in either Phase I (Program Definition & Risk Reduction) or Phase II (Engineering & Manufacturing Development). At this point, the opportunity for human factors issues to affect system design is greatly reduced. In addition, the costs associated with changing system design in these phases are greatly increased. The CART HPM Environment has capabilities that support human performance model development and varying levels of specificity. This allows human factors issues to be incorporated into system design and requirements at a much earlier time. The CART HPM Environment helps requirements generators to develop human performance models that interact with engineering level models. This model interaction provides users with an indication of how human operators might interact with new technologies.

An example of the type of analysis that a CART user might perform is in the evaluation of the performance requirements for a new radar system to be used by a combat pilot. Our user might be very concerned with how the capabilities of the radar system would affect the performance of the pilot flying a reconnaissance mission during which the pilot is to locate and identify a potential target. If the pilot detects a threat during this planned mission, then the pilot will probably interrupt the planned mission and evade the threat. The time at which the threat is detected will affect the point in the mission that the pilot begins the maneuver and the time available to successfully evade. Similarly, if the pilot detects a high payoff target within range, he may interrupt the mission to attack the target. Depending upon the success of that maneuver, the planned mission might be resumed, aborted, or restarted.

In this example, the pilot has a mission that could be interrupted if either of two competing goals is triggered (to evade the threat or to attack the target). These goals have an inherent priority (evading the threat is much more important than attacking the target) and their impact on the mission and each other is complicated by environmental variables (i.e., whether the pilot attempts to resume the mission after evading the threat depends on his position and how much time is left to perform the remaining tasks).

In order to represent this in CART, the analyst would use the point and click GUI interface to draw three independent task networks. The first one represents the mission (see Figure 1).

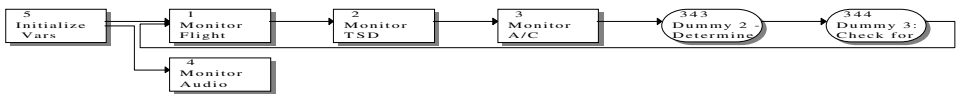

Figure 1. Perform Mission Task Network

Although Figure 1 is simplistic, the mission network must be sufficiently complex to include parameterization of environmental aspects. For example, the mission network and its associated logic must be powerful enough to predict a change in performance based upon aspects such as weather, target availability, or any other variables that would affect the performance but not change the overall tactics.

The second and third figures show the networks that represent the two competing goals that influence the pilot's behavior (see Figures 2 and 3).

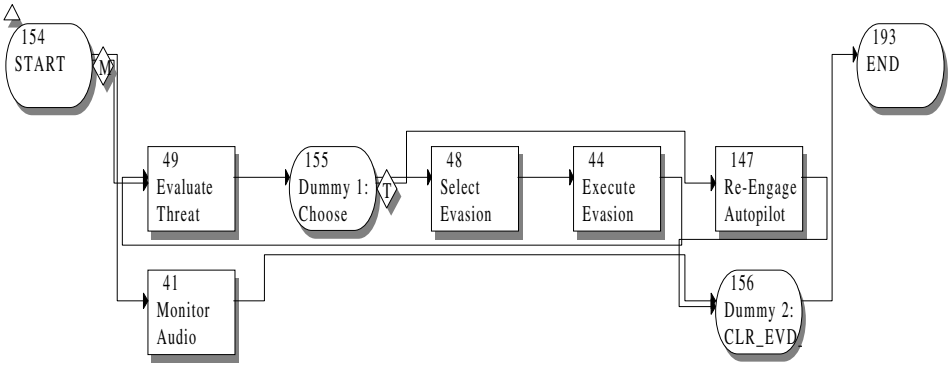

Figure 2. Evade Goal Task Network

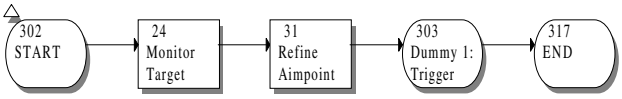

Figure 3. Attack Goal Task Network

Similarly to the mission, each goal network must be sophisticated enough to represent the range of actions that the pilot might select, given that the goal is triggered. The key to predicting the realistic performance of the pilot becomes the prediction of when goals are triggered, and then the representation of the newly triggered goal's impact on the other actions that the pilot takes.

CART SYSTEM ARCHITECTURE

In order to develop a modeling tool for this situation, we began by selecting a task network modeling environment, named the Improved Performance Research Environment (IMPRINT) owned by the US Army (Archer and Adkins, 1999). We chose this environment primarily because the discrete event simulation techniques included in IMPRINT are very well suited for human performance modeling. Secondly, IMPRINT is a stable software tool, originally developed by the Army Research Laboratory (ARL) Human Research and Engineering Directorate (HRED) to support the assessment of human performance in the context of total system performance in complex environments. IMPRINT provides a mature architecture and database structure that can be expanded to incorporate a modeling method for representing goal-oriented behavior.

IMPRINT (and also the modified version of IMPRINT being used by CART) assists users in estimating the likely performance of a new weapon system by helping them build models of each operational mission that the system will be required to accomplish. Since it is often easier to describe the mission by breaking it into smaller "sub" functions than it is to describe the mission as a whole, users build these models by breaking down the mission into a network of functions using a point and click GUI (see Figure 4). Each of the functions is then further broken down into a network consisting of other functions and tasks. Then, users estimate the time it will take to perform each task and the likelihood that it will be performed accurately.

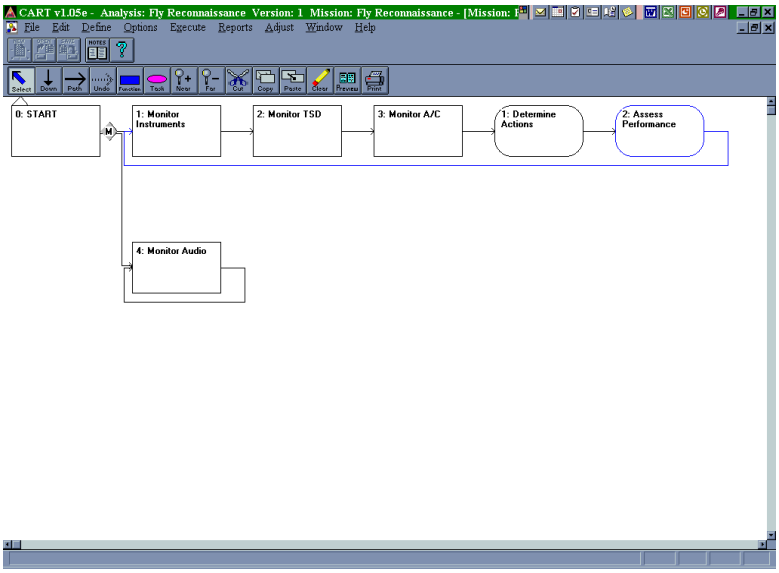

Figure 4. CART Task Network Drawing GUI

Finally, by executing a simulation model of the mission, users can study the range of results that occur in the mission. A description of the variability of each element can be obtained for further analysis. Additionally, at the completion of the simulation, IMPRINT can compare the minimum acceptable mission performance time and accuracy to the predicted performance. This determines whether the mission met its performance requirements.

We expanded IMPRINT to represent goal-oriented behavior by implementing two fundamental changes.

First, we enabled users to represent the tasks that the human crew would take in response to a goal as separate task networks, not linked to the mission level model. Each of these networks is then associated with an initiating, or triggering, condition. An example of a triggering condition might be that a threat has approached within sensor range. The CART user lists the goals and enters the arithmetic and logical expressions that specify when each goal will be triggered (see Figure 5).

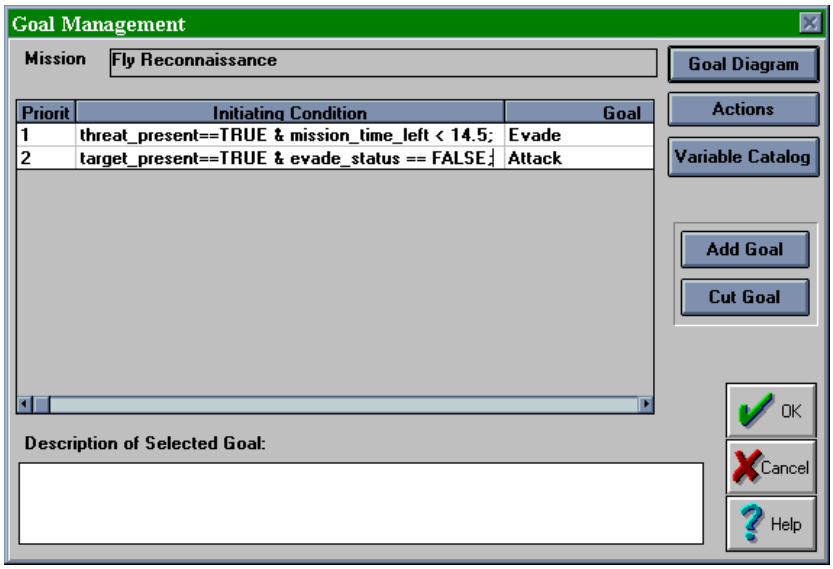

Figure 5. Goals and Triggering Conditions

Second, the goals must be prioritized so that they interrelate properly. For example, our combat pilot will have one over-riding goal-- to accomplish the assigned reconnaissance mission. However, the advent of a threat will change the pilot's immediate goal to "evade threat and survive." After the pilot has successfully evaded the threat, he may resume the mission at the appropriate place on the flight path. Alternatively, the second, lower priority goal of attacking a target, may be triggered by the target being available. To complicate this situation, if a target appears during the prescribed mission, and the pilot attacks it (by triggering the attack goal) and then a threat appears, the pilot will immediately abort the attack and begin evasive maneuvers. This equates to aborting the lower priority goal when the high priority goal is triggered. In CART, the user can achieve this behavior through the Goal Action matrix (see Figure 6).

In this simple case, there are only two goals, and they are mutually exclusive. In a more typical case, you would have several goals that compete for the pilot's attention. Therefore, the tool must have a robust capability through which users can specify which goals are most important and, once triggered, whether the tasks associated with ongoing lower priority goals are interrupted or halted.

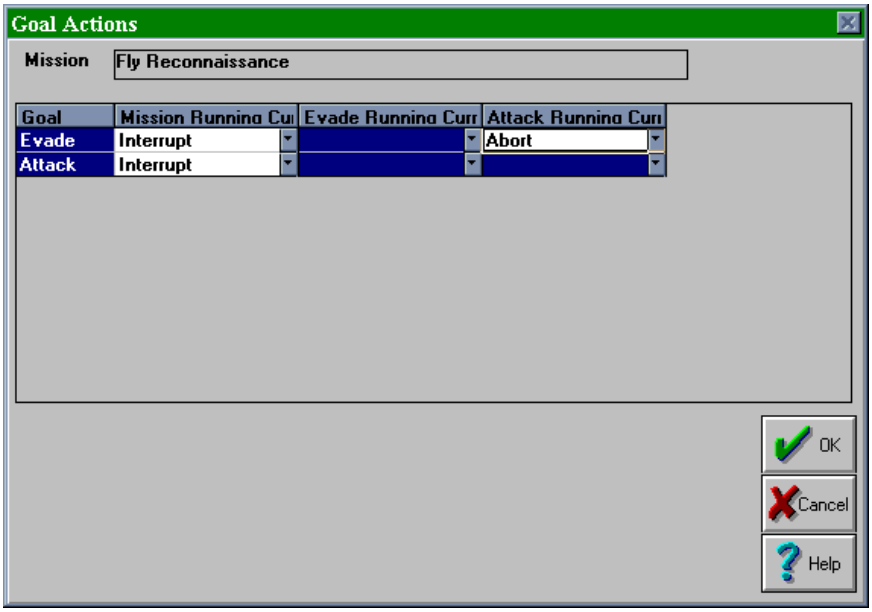

Figure 6. Goal Action Matrix

In order to develop a tool that is capable of representing this behavior, many complex issues had to be resolved. First, the task network model has to be capable of exchanging variable values and model control parameters with other currently executing models. In this way, variables that control the triggering conditions (e.g., threat availability) can be established through other models, such as radar or sensor models. Second, the underlying task network structure must be robust enough to anticipate the range of goal states in which you are interested, even if some of those goal networks may never be triggered and hence never executed during a particular model execution. These capabilities are provided through a straightforward architecture (see Figure 7).

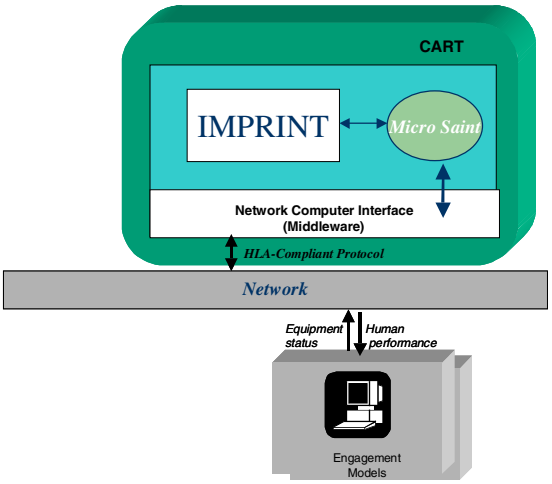

Figure 7. System Architecture

The executing goal-oriented human performance model is in a Micro Saint format, and can either run in a stand-alone mode or in a network mode. The translation from the IMPRINT model to the executable model is accomplished through a process we refer to as "Model Generation." In this process, information users have entered in the IMPRINT GUI is stored in a relational database structure, and then converted into code that the commercial simulation tool Micro Saint understands. This conversion is very complex, since it converts users' task networks, goal triggering conditions, and goal interactions into a fully parameterized simulation model. This model can then be executed, and the results are sent to the user in formatted CART reports. The CART users do not have to understand Micro Saint syntax to build simple models, nor do they have to be simulation experts to develop simple human performance models.

Trigger Communication and Middleware

The success of the architecture shown in Figure 7 rests upon the ability to send and receive variables from the CART human performance simulation model to other simulations. To achieve this, CART HPM acts as a federate within a High Level Architecture (HLA)-compliant federation. In order for this interaction to occur, middleware was developed. Most of this middleware is available commercially or free for download from the Defense Modeling and Simulation Office (DMSO). However, in order for CART users to be able to ensure that appropriate HLA-compliant functions are sent across the federation, the middleware was integrated into the CART software. The data sent across the federation, object attributes and interactions, were mapped to the Real-time Platform Reference Federation Object Model (RPR FOM). It is through this middleware that the goal is triggered and simulated human performance is communicated.

The components of the CART middleware are described below (see Figure 8). User-defined external variables are the interface between CART models and other HLA federates prior to and during federation run-time. During federation run-time, Component Object Model (COM) services are used to communicate between the middleware and the CART human performance models. VR-Link provides an interface to the Run-Time Infrastructure (RTI).

The Middleware Interface allows the CART user to map external variables to RPR FOM object attributes and interaction parameters prior to run-time. Middleware Run-Time Code allows the RTI with VR-Link to interact with COM during run-time.

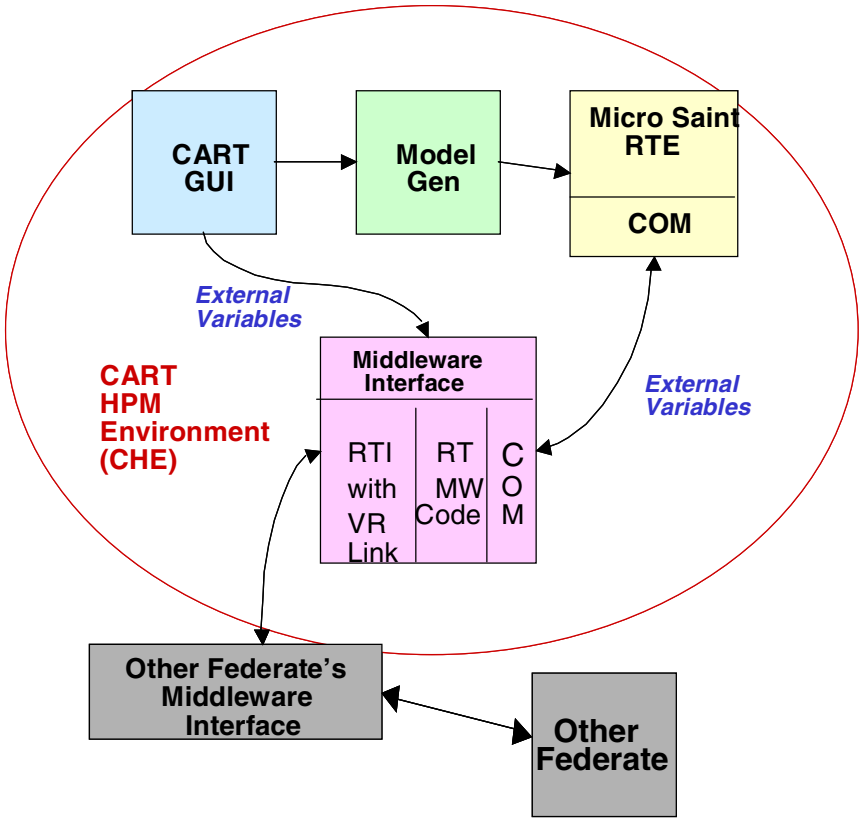

Figure 8. CART Middleware Components

External Variables

To support this concept, a capability was added to the CART GUI through which the user could identify any of the human performance model variables as "externals." This means that these variables would either be received from or sent to another model in the federation. These external variables are the key to interoperability between a CART model and other federates in the CART federation.

Prior to federation run-time, the user maps the external variables to HLA RPR FOM object attributes and interaction parameters, and to actions to be taken when the variables are encountered during federation run-time.

During federation run-time, COM services within the human performance model determine which variables are external, and communicates to the middleware that these variables need to interact with the federation. For example, if an external variable is used in a CART model, and that variable has been mapped to send specific RPR FOM data in the Middleware GUI, data will be sent through COM to the middleware at the simulation time that the variable is used. The middleware calls the appropriate VR-Link services, which in turn call the appropriate RTI services. These services send the data out to other federates in the CART federation that subscribe to those data. Federates that subscribe to those data will update the variable value at that simulation time. Similarly, if an external variable has been mapped to receive data in the Middleware GUI and is published by another federate in the CART federation, the CART model will receive an update from COM services via VR-Link and the RTI when that variable changes.

Component Object Model (COM) Services

COM services provide the primary method of data transfer between the CART middleware and the human performance model during federation run-time. The COM services recognize external variables as described above. External variable values and other simulation management functions are passed between the executing human performance model and the CART middleware through COM services. The CART model receives external variable updates, expressions representing interactions (or events) triggered by variables, and time advance grants based on input from other models in the federation through the middleware. COM also allows the CART human performance model to send variables and the time for the next event to the middleware, which will then be sent to other federates in the federation.

The Middleware Interface

The Middleware Interface allows the CART user to map external variables to HLA object attributes and interactions (or events). It also allows the user to tell the rest of the middleware that the human performance model wants to send (or publish) and/or receive (or subscribe to) the HLA object attributes and interactions. The interface includes a Middleware GUI (see Figure 9).

The External Variable Name and Type columns include a list of external variable names assigned by the CART user in the CART GUI and the variable type for each. Each name is associated with all of the other parameters of the variable, including the variable type (i.e., real number, integer, array of reals, or array of integers). A user selects any of the variables by External Variable Name and then maps them to the rest of the table using a series of pull down menus.

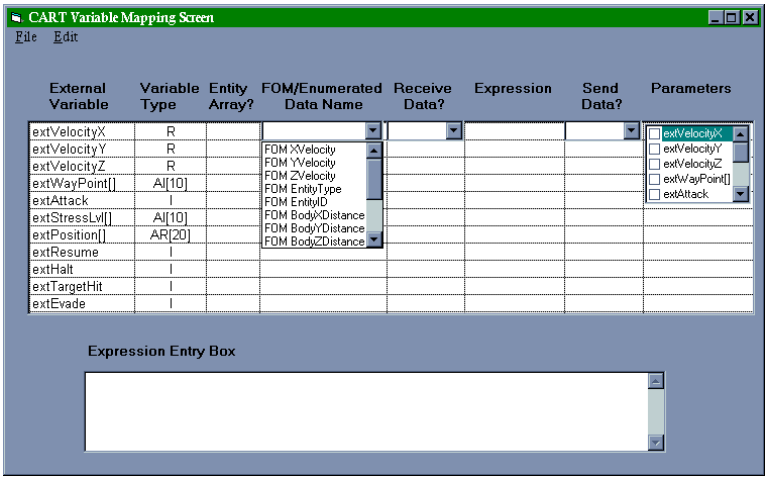

Figure 9. Middleware GUI

The FOM/Enum Name column has a list of RPR FOM object attributes and interactions and enumeration names from the enumerated data list.

The Receive column allows the user to tell the rest of the middleware whether the CART model will receive (R), receive as an event (RE), or not receive (NA) each variable.

The Expression column only applies to variables that have been assigned with an "RE" in the Receive column. For any variable received as an event, the user can input a CART/Micro Saint expression that will be passed into the human performance model event queue when the variable is received by the CART model during federation run-time.

The Sent column allows the user to tell the rest of the middleware if each variable is one that the CART model wishes to send (S), send as an action (SA), or not send (NA).

If a user maps a variable to "SA" in the Send column, the final column of the table allows them to select other external variables to package and send as parameters of the variable (representing an event or interaction) being mapped.

HOW THE CART MIDDLEWARE MANAGES HLA

There are six defined areas of HLA management, namely:

- Federation Management
- Declaration Management
- Object Management
- Ownership Management
- Time Management
- Data Distribution Management

In order for a federation to be HLA-compliant, it must employ the first three types of management listed above. CART also employs Time Management, but does not utilize Ownership Management or Data Distribution Management.

Federation Management and Time Management are handled by the CART middleware with no user intervention required once CART models are built and running. Most of the required user intervention is needed prior to run-time when the user maps external variables to RPR FOM object attributes and interactions and assigns receiving and sending action to them. This mapping helps to establish the Declaration Management functions in the RTI with VR-Link. Once the user has

mapped the external variables to RPR FOM object attributes and interactions, the RTI and VR-Link will pass Object Management functions between the CART model and other federates as the model runs.

Federation Management

Federation Management is handled by the RTI with VR-Link without any interaction from the CART user. The CART middleware calls VR-Link services, which in turn calls RTI services.

Another Federation Management function that is invoked frequently as the simulations run is the tick() method. Frequent invocation of this method ensures that the RTI does not fail to process other HLA management functions.

CART also employs federate synchronization, saving federations, and restoring federations.

Declaration Management

As stated earlier, the user uses employs the Middleware Interface to map external variables to FOM defined object attributes and interactions and tell the middleware whether the CART model wants to send and/or receive each one. This queues the RTI to send the necessary Declaration Management function calls across the federation for those attributes and interactions. For data that the CART federate wants to send to the federation, publish function calls are created. For data that CART wants to receive from the federation, subscribe function calls are created.

Object Management

Once external variables have been mapped to object attributes and interactions in the Middleware Interface, Object Management is automated based on the use of external variables in the CART model during run-time. The Middleware Interface allows the CART user to map external variables to RPR FOM object attributes and interactions, and to tell the rest of the middleware how to treat the variables. Once these variables are mapped to object attributes, interactions, and desired actions, the RTI with VR-Link can invoke the appropriate HLA function calls when the variables are encountered during run-time.

Time Management

CART models are discrete event simulations. In order to utilize Time Management, and for these models to properly interact with other simulations in the CART federation, they need to be both regulating and constrained. In other words, they can send and receive Time-Stamp-Ordered (TSO) events. CART uses a single step, zero look-ahead time management scheme. This ensures that the models do not miss any object attribute changes or interactions that are received between time advancements. Because of the zero look-ahead, CART does not have to consider out-of-order event processing. In other words, the CART system does not have to cycle back in time to recover TSO events.

Within the human performance model during run-time, the concept of a "Time Event Group," or "TEG," was invented to represent events or ticks that occur during the same logical or simulation time (represented by "clock" in Micro Saint). A TEG can consist of one or more events within a model during run-time. This concept is very important to understand how HLA Time Management and Object Management execute. Variables that represent object attributes may change several times within a TEG, however, only the last variable value within a TEG is transferred between federates. The human performance model will advance from one TEG to another with a tick subsequent to the TEG, unless an event (or interaction) is sent to the model from another federate which is time stamped between the times of the two TEGs.

CART uses the nextEventRequest Time Management function to determine how far the human performance model can advance in time within our federation. This function allows events between time advancements to be received, unlike the timeAdvanceRequest function. CART sends out nextEventRequest function calls based on the time of its next scheduled TEG. The RTI with VR-Link sends CART a timeAdvanceGrant to the next event that occurs in the federation.

A Word about HLA Compliance

Although the CART Middleware is utilizing the RPR FOM, the CART middleware itself does not guarantee that HLA-compliant simulations will be created. The CART middleware provides tools to build HLA simulations, but CART users need to develop and integrate HLA-compliant simulations (or federates) into an HLA-compliant federation. A useful document in determining if a federation is HLA-compliant is the HLA Compliance Checklist. This document contains three separate checklists; one for a federate, one for a federation, and one for the RTI.

For each CART model and for the other federates in the CART federation, the CART user will develop specific Object Model Templates (OMTs) and Simulation Object Models (SOMs).

FUTURE EFFORTS

During the current phase of the CART effort, the project team has maintained a "wish list" of items that could significantly improve the performance of the tool.

Improved Performance

The current version of CART, when integrated into a HLA federation, does not run in real time. Efforts are currently underway to develop a profile of the linked system to determine whether the problem lies within the communication protocol design, or whether the problem lies with one of the individual simulations in the federation. It is certain, however, that the CART middleware will need to be optimized for performance if the human performance model is ever to meet a real time standard.

32-bit Data Structures

The original IMPRINT software, which forms the core of the CART HPM was originally developed in 1993. While it has been continuously updated and improved, some of the original limitations of the 1993-era operating environment have remained. One of these limitations is that the data structures and supporting input/output code were developed in a 16-bit environment. Efforts to upgrade the data structures to 32-bit have been delayed for practical reasons. The current database structure and engine can be distributed without fee to the users, making it much easier to maintain and manage user updates to new versions of the tool. A 32-bit SQL database with a similar distribution agreement is not available, and to update to a product with a more restrictive distribution agreement would be difficult for the Government to manage.

Unfortunately, however, the limitations of the 16-bit data structures are overcoming the concern with distribution issues. These limitations are occurring in two areas. The first area is in the loss of accuracy during the conversion of external variables to internal CART representations (possibly affecting the proper communication of triggers). The second concern affects the internal capacity and processing efficiency of the tool. Users of CART do sometimes experience system problems with large models, particularly if other software programs are open at the same time. A 32-bit data model would alleviate this problem.

Integration with Improved Decision Making Micromodels

The human performance modeling community has invested considerable effort recently in developing a better understanding of how to model human decision-making. CART team members have been involved in this work, and believe that the CART software and the underlying IMPRINT tool structure could benefit from a higher fidelity representation of decision making.

This would require a standard representation of human situation awareness, one that is context specific and can be populated as the simulated human encounters tasks in the network (which are, of course, driven by the current goal state).

The primary benefit of this addition would be to better capture the impacts of personnel selection and training on the performance of cognitive tasks such that system designs can be altered to better support the decision making process.

CONCLUSION

CART is still in development, but is far enough along that we believe we have solved the most significant problems with automatically suspending and resuming task networks to represent the actions associated with current goal states. We can also demonstrate how data from external models can be used to control the modeled behavior of a goal-oriented crew member. While the tool has not yet been verified or validated, we are satisfied that CART provides capabilities not previously available to human performance modelers. The integration with other models provides a platform with which personnel training and selection questions can be studied early in the weapon system acquisition process.

REFERENCES

Allender, L., Kelley, T., Salvi, L., Headley, D.B., Promisel, D., Mitchell, D., Richer, C., and Feng, T., "Verification, Validation, and Accreditation of a Soldier-System Modeling Tool. In the Proceedings of the 39th Human Factors and Ergonomics Society Meeting, October 9-13, San Diego, CA. Available from the Human Factors and Ergonomics Society, Santa Monica, CA. 1995.

Archer, S. and Adkins, R. "IMPRINT User's Guide" prepared for US Army Research Laboratory, Human Research and Engineering Directorate, April 1999.