N93-32126

ADVANCED ARTIFICIAL INTELLIGENCE TECHNOLOGY TESTBED

Mr. Craig S. Anken

Air Force Material Command
Rome Laboratory
Griffiss AFB, NY USA
13441-5700

Abstract

The Advanced Artificial Intelligence Technology Testbed (AAITT) is a laboratory testbed for the design, analysis, integration, evaluation and excercising of large-scale, complex, software systems, composed of both knowledge-based and conventional components. The AAITT assists its users in configuring various problem-solving application suites; observing and measuring the behavior of these applications and the interactions between their constituent modules; gathering and analyzing statistics about the occurrence of key events; and flexibly and quickly altering the interaction of modules within the applications for further study.

1.0 Introduction

The importance of "intelligent", large-scale military decision support systems for effective command and control is becoming more and more apparent. In time, decision aids will be prevalent throughout every aspect of military operations, aiding in decisions that will have major impacts on the battle. These systems will be composed of both knowledge-based and conventional modules, and will interact as part of some predefined problem-solving strategy. The ability to iteratively define this strategy, integrate, and deploy these suites thus becomes very important.

An intelligent command and control (C2) decision aid often begins its life in a "sterile" laboratory environment. Each decision aid is developed to solve a relatively narrow problem within the overall C2 decision-making/ management problem. To provide the user guidance, this aid has sources of data and knowledge stored in local format(s), a problem solving methodology which uses the knowledge to reason over the data, and usually the ability to interact with the user. Testing may be performed by presenting the aid with predefined scenarios and comparing the results to some standard. While this approach may be adequate in the laboratory setting, what happens when this tool is brought into the operational realm? That is, when it is brought into an environment where information may be stored in several databases and where multiple decision aids and/or conventional programs are working at various portions of the overall problem. Finally, how reliable will this system be, having been developed in isolation thousands of miles from the battlefield? To answer some of these questions, the Rome Laboratory has embarked on the development of the AAITT, a distributed environment that will support the integration, testing of, and cooperation among multiple decision aids.

2.0 Advanced AI Technology Testbed (AAITT)

Before describing the AAITT, several terms need to be defined. A component is a stand-alone software program designed to solve part of an overall problem (see Figure 1). User-supplied components (USCs) refer to those components which the application developer supplies and wants connected to the testbed. A module is a component with some type of software "wrapper" that allows it to communicate with other modules. An AAITT application is a collection of modules configured to work interactively to solve an overall problem.

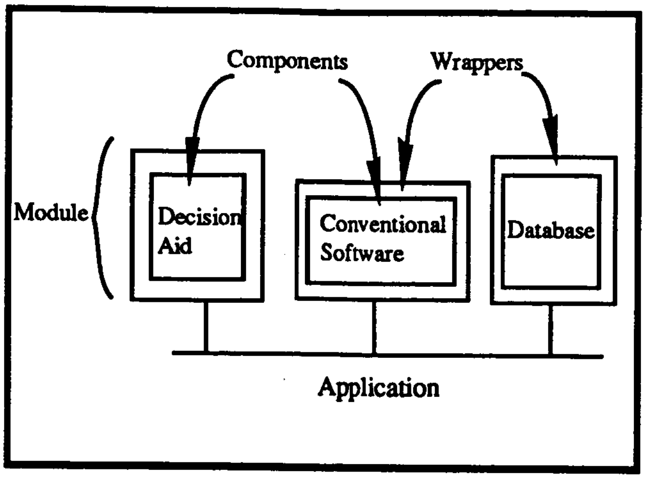

Figure 1. Testbed terminology

The goal of the Rome Laboratory Advanced AI Technology Testbed is to develop a decision aid/conventional software integration environment which will allow the user to:

1. Easily configure various application suites by providing tools to add or remove user-supplied components, or to modify the communication paths of the various problem solving modules,

2. Provide generic database and simulation modules that can be tailored to the needs of the current application,

3. Observe and trace these modules' actions and interactions,

4. Select, gather, and review metrics and statistics on run-time performance, and

5. Rapidly change the flavor of the interactions among the suite's components based upon the results of previous runs.

The components of the AAITT, shown in Figure 2, consist of the testbed manager (later described as the MCM Workstation), a database, a simulator, and various USCs.

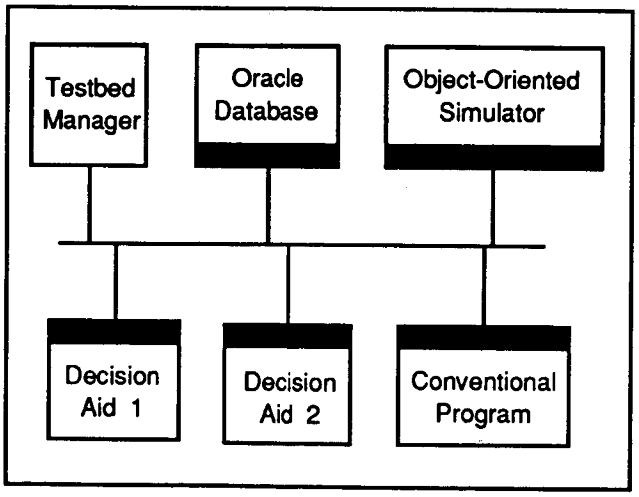

Figure 2. Network View of AAITT Application

The grayed sections in the figure above represent interface software. A software controlled architecture (soft architecture) has been chosen for the AAITT. A soft architecture is one that can be graphically configured and modified by a testbed manager, while minimizing the recoding of the interface software by hand. An example configuration is shown in Figure 3. To try a different communication architecture (e.g., a blackboard approach instead of data-flow), the user would make the necessary changes using the graphical interface provided by the testbed manager.

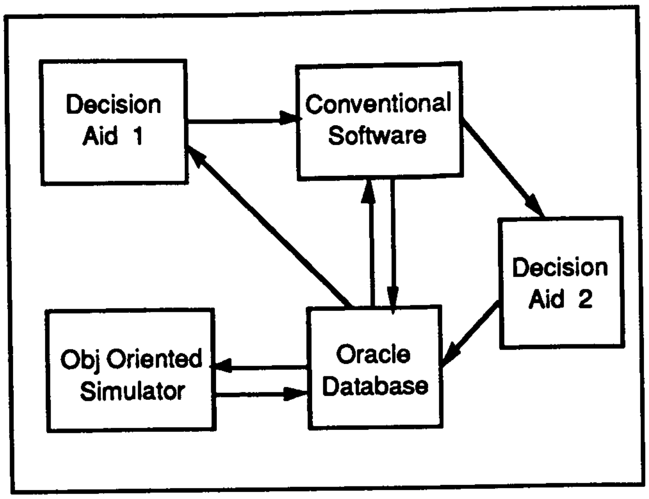

Figure 3. Experimental AAITT Architecture

To support the soft architecture envisioned by the AAITT, three major sub-systems are being developed. These are the Distributed Processing Substrate (DPS), the Module Framework, and the Modeling, Control, and Monitoring (MCM) Workstation.

2.1 DPS

The Distributed Processing Substrate will allow dissimilar software systems (e.g., databases, simulations, expert-systems, and conventional software) running on heterogeneous hardware platforms (e.g., VAX, SUN, and Symbolics) to interact with one another. An important feature of the DPS is that is will translate data representations between the various hardware systems and programming languages. Several candidates were considered by the contractor team for providing the foundation of the testbed and are shown in Table 1.

Product	Developer
Alpha	Carnegie Mellon Univ.
Cronus	BBN Corp.
ISIS/Meta	Cornell University
Mach	Carnegie Mellon Univ.
MetaCourier	Symbiotics, Inc.
Star O/S	ADS Corp.

Table 1. Candidate Environments [GE-91]

The criteria for selection was:

1. Runs on Sun 3/4 and Symbolics 36xx.

2. Runs on Vaxs running VMS and ULTRIX.

3. Runs with applications written in C/C++ and Common Lisp.

4. Runs with applications written in Ada.

5. Provides interface specification and stub code generators.

6. Provides application building tools.

7. Stable, mature product.

The resulting evaluation of each candidate is shown in Table 2.

Product	Alpha	Cronus	Isis/ Meta	Mach	Meta Courier	Star O/S
Sun3/4 Symbolics	No	Yes	No	No	Yes	Yes
Vax,VMS Ultrix	No	Yes	No	No	Unknown	No
C/C++ Lisp	No	Yes	No	No	Yes	Yes
Ada Interface	No	Future	No	Yes	No	No
Int. Spec & Stub Code Gen.	No	Yes	No	Yes	Yes	No
Appl Bld. Tools	None	Excl.	Good	Good	Good	None
Stability	Poor	Excl.	Excl.	Excl.	Fair	Excl.

Table 2. Candidate Env. Evaluation [GE-91]

Based upon this analysis, the Cronus distributed computing environment [BBN-89] was selected. Cronus provides heterogeneous host support for distributed application development. It gives the user an object-oriented view of resources on a network. In addition to this object-oriented view provided by Cronus, the DPS will use ABE [Erman-90], providing a module-oriented programming capability. This will allow the structuring of a distributed testbed application at a higher level than Cronus. The ABE environment supports modules which are independent entities that communicate data and control among themselves through very well-defined interfaces. The goal is to use the higher-level, computational and architectural model provided by ABE with the lower-level, distributed system environment support provided by Cronus.

2.2 Module Framework

The Module Framework will allow the user to easily embed a new component in the AAITT. This framework is used to create Component Interface Managers (CIMs). A CIM is a wrapper that provides the interface between the distributed testbed and each component. The Module Framework will facilitate the creation of the CIM by guiding the user through the creation process and by providing a library of generic CIMs. It will also provide a semi-automated generator for easily creating and modifying CIMs. CIMs will also allow the testbed to control module loading, initialization, resetting, etc.

There are several types of communication that will occur at a given CIM. These types include simple reception and transmission of data, and may include more complicated transmissions which then wait for a response. When transmitting, the CIM must know which module to send the message to. The name of the destination module will be set when the user completes the modeling phase at the MCM Workstation. The recipient can easily be changed by making the appropriate changes at the workstation. (Note: the actual location on the network of the receiving module will be maintained, transparently to the user, by the DPS.) CIM to CIM message types include ASCII text, integer, real, and database query response.

2.3 MCM Workstation

The Modeling, Control, and Monitoring Workstation will provide a central user console for building and configuring applications, and for the display and analysis of performance metrics gathered during application runs.

The modeling functions will allow a user to specify graphically how the modules are to interact with one another during execution. This will define the architecture of the application. The control functions will allow the user to load, initialize, execute, and reset the components distributed over the testbed's network. Break point capabilities will aid in the debugging process. The monitoring functions will allow the user to specify measurements, monitors, and instrumentation. Measurements refer to the quantifiable features that help the user understand system performance. Monitors are the procedures through which measurements are captured. Instrumentation allows the user to process the resulting measurements and present them in an appropriate manner. The monitoring capabilities of the MCM Workstation will be very important in testing put the USCs and set the AAITT apart from testbeds that simply try to connect existing systems.

2.4 Testing

As previously stated, it is very important to understand the strengths and weaknesses of the USCs before providing them to operational units. These weaknesses may include actual errors, problem size limitations, excessive run times, incompatibilities, and so on. The AAITT will have features to address the problem of verification and validation of USC's. The questions, is the software doing the job right, and is the software doing the right job, will need to be answered.

"Job Right"

Determining whether the SW is doing the job right will provide the end users with some confidence that the USCs will work as advertised. To verify the results of the USCs, the AAITT will provide metrics for evaluation and heuristics. Timing and memory limitations will be addressed by these metrics. In addition, bottlenecks will be identified, possibly indicating that a new application architecture should be selected or that faster computers/networks should be used. Higher level metrics will also be addressed, for instance the number of mission routes planned per minute, or the speed at which the simulation simulates one hour of time.

The quality of a solution cannot be determined by the speed of the inference engine or the number of queries handled by the database per minute. Quality can sometimes be assessed by using some type of heuristic. To support this, the AAITT will need some type of generic heuristic evaluation module that can be tailored to evaluate a given class of USCs. For instance, a route planner heuristic evaluation module may take the planned routes and determine for each route if the specified aircraft can fly at the chosen altitudes, if the aircraft has enough range, given the available on-board fuel, and whether the legs of the route are too short (difficult to follow) or too long (vulnerable to enemy attack). This evaluation module could be used regardless of which specific route planner was being tested.

"Right Job"

In determining whether the software is doing the right job, end users need to make sure the USCs really address their particular problems. To validate whether or not the USCs are meeting their needs, the testbed will provide rapid prototyping facilities and the benefit of graphical simulations.

The testbed will allow component developers the ability to quickly demonstrate their systems in a realistic environment. By providing both database and simulation capabilities, developers can concentrate on software algorithms and user interface issues, instead of developing sets of problems and ways to evaluate results. This will mean that USC users can be brought in earlier in the development process, allowing them to direct or redirect the developers before design decisions are made. This process will help developers better understand the users' needs, not just the users' stated requirements.

The simulation component will provide a simulation language and various tools for developing graphical simulations. This will allow users to monitor the execution of plans developed by the USCs. The simulation language is designed to be interactive so that user developed simulations can be stopped, queried, modified, and continued at any point. This type of interactive simulation should help in determining whether the USC developer has really addressed the user's needs.

2.5 AAITT Support Components

The AAITT will have generic components to facilitate the development and testing of USCs. These generic components have been used to create demonstration components to plan missions in the tactical air combat domain.

Conventional and "expert" software work by understanding the problem at hand, automatically or semi-automatically attempting to solve the problem, and then posting the results (usually to a screen or file). Databases and simulations can provide a supporting environment for evaluating these types of software systems. A database can be used to present the USCs with specific problem scenarios. Once the USCs finish processing, the results of the components can be stored in the database for later evaluation. Heuristics can be helpful in evaluating the quality of a result, for instance the routes chosen by a route planner could be evaluated on the percentage of time that allied aircraft are exposed to enemy air defenses. If a heuristic is not available, a simulation could be helpful, allowing a human evaluator to see the results of following the recommendations of a particular USC. Generic components will be provided to develop these types of databases and simulations.

The testbed will have a generic database for storing and retrieving information needed by the USCs. Results of the USCs can be stored in the database for later evaluation. The Oracle relational database has been chosen as the testbed's database shell. Using the DPS, the database will appear as a module in the testbed that can respond to structured query language (SQL) statements. As additional databases are created, they can then be used by other USCs.

The testbed will have a generic simulation capability to show the consequences of following the results of the other modules. The ERIC object-oriented simulation language has been chosen as the testbed's simulation language [Hilton-90]. ERIC is based on the Common Lisp Object System (CLOS) and allows the simulation developer to develop a hierarchy of object classes with associated behaviors and attributes. ERIC allows objects to be created dynamically and for the simulation to be halted, modified, and continued without recompiling. The message parser provided by ERIC allows for expressive message passing. Results from the simulation can be posted to the database or sent on to USCs.

2.6 Demonstration Components

Demonstration components will be used during the AAITT demonstrations to show the feasibility of the testbed.

TAC-DB

The Tactical Database (TAC-DB) provides a realistic, though unclassified, laydown of tactical units and equipment in the central European theater [Kearney-90]. This database was developed by Knowledge Systems Corporation (KSC) in 1989 and reflects the NATO and Warsaw Pact capabilities at that time. In addition to identifying where units are located, the database contains information on individual weapon systems. As new USCs are added to the testbed with new requirements for domain information, TAC-DB will be extended to provide the necessary support.

TAC-DB provides the "blue-view" of the world for the USCs. This means that information in the database is only as accurate as blue intelligence is. This makes sense since USCs should not be able to access more information than the blue side knows. (Note: there may be some "red-view" information stored in the database used only by the simulator to create red units, etc., that are currently unknown to the blue side.)

LACE

The Land Air Combat in ERIC (LACE) simulation simulates the tactical engagement of NATO and Warsaw Pact forces in the central European theater [Anken-89]. The simulation reads in friendly and enemy unit information from TAC-DB, creates these objects, and then executes air tasking orders (ATOs) also stored in the database. Friendly air missions must penetrate through enemy air defenses in order to attack targets. As missions return, their results are posted to TAC-DB for use by the USCs.

LACE contains the ground truth for the scenario. The USCs do not have direct access to this ground truth data, only to the TAC-DB. The USCs could, however, be used to schedule LACE reconnaissance missions which will post their intelligence reports to TAC-DB upon completion. This information then becomes available to the other modules of the testbed.

AMPS / RPDW

Two decision aids will be included as part of early demonstrations. These include the Air Force Mission Planning System (AMPS) developed by The MITRE Corporation and portions of the Route Planner Development Workstation (RPDW) developed by the Jet Propulsion Laboratory. AMPS [Dawson-90] was developed to support the planning and replanning of ATOs. ATOs are the plans used by NATO to conduct air missions. AMPS contains domain knowledge concerning aircraft and weapon capabilities, weapon system availability, and scheduling constraints. In addition to planning these missions, AMPS was developed to intelligently support replanning, the process of fixing a plan which has problems, without replanning the whole ATO.

The RPDW [Cameron-85] was designed to facilitate the development and evaluation of route planning algorithms. Our route planner is one of the algorithms provided by the RPDW. To use this planner, the user must first develop a threat-contour map based on the location and line of sight of the various air defense systems. The planner then determines a route through the map while attempting to minimize aircraft vulnerability. The results generated by AMPS and the route planner will be sent to TAC-DB and then on to the simulation. The idea is to provide feedback to the decision aid developers concerning their systems.

3.0 Summary

C2 decision aids and conventional programs are being developed to support operational commanders. An environment is needed to allow these dissimilar systems to easily work together. In addition, realistic testing of these components is required to make sure they perform as expected. The AAITT is designed to address these concerns by providing a loosely coupled, distributed support environment, allowing independently developed aids to cooperate. The generic database and simulation capabilities will allow for easy development and evaluation of C2 decision support systems. If the database is large enough and the simulation can provide updates at fast enough rates, the testbed can present the decision aids with realistic wartime environments. This is the kind of interoperability and evaluation that is needed before fielding composite operational systems.

REFERENCES

[Anken-89] Anken, C.S., "LACE: Land Air Combat in ERIC," RADC-TR-89-219, October 1989.

[BBN-89] BBN Systems and Technologies Corp., "Cronus Advanced Development Model," RADC-TR-89-151 Vol. 1, September 1989.

[Cameron-85] Cameron, J., Cooper, B., Mishkin, A., "Route Planner Development Workstation, Volume 2 - Software User's Guide," JPL D-2733 Vol 2, U.S. Army Engineer Topographic Laboratories, Jet Propulsion Laboratories, November 1985.

[Dawson-90] Dawson, B., Day, D., Mulvehill, A., 90 "The AMPS," RADC-TR-90-131, July 1990.

[Erman-90] Erman, L., Davidson, J., Lark, J., Hayes-Roth, F., "ABE: Final Technical Report," TTR-ISE-90-104, May 1990.

[GE-91] GE/ATL, "Advanced AI Technology Testbed, Technical Information Report (DPS) Analysis Draft," F30602-90-C-0079, February 1991.

[Hilton-90] Hilton, M. and Grimshaw, J. "ERIC Manual," RADC-TR-90-84, April 1990.

[Kearney-90] Kearney, H., Lazzara, A., Pendergast, J., Richer, A., Erich, R., "Cooperative Red and Blue Database System," RADC-TR-90-98, June 1990.