A Hybrid Architecture for Visualization and Decision Making in Battlespace Environments

J. Peng, J.W. Rozenblit and L. Suantak
Department of Electrical and Computer Engineering
The University of Arizona
Tucson, AZ 85721-0104, USA
jr@ece.arizona.edu

Abstract

This article presents a hybrid software/hardware architecture for commander's decision support in tactical operations. The architecture builds on the symbolic, object-oriented visualization software called Advanced Tactical Architecture for Combat Knowledge System (ATACKS). The extension discussed here is the design of a real-time robot agent layer that interacts wirelessly with ATACKS. This layer enacts decisions made by software agents (wargamers), continuously relays the execution states back to ATACKS, and updates its actions as advocated by replanning algorithms. The software layer is briefly described followed by the specification of the real-time requirements for the robotic architecture. The design and implementation are given with a small example that illustrates the hybrid system's operation.

1. Advanced Tactical Architecture for Combat Knowledge System (ATACKS)

In this section, we provide a brief background on the ATACKS architecture. The goal of our ongoing research and development work is to build a computer-based environment that can portray multifaceted military and other (e.g., disaster, refugee relief) operations in a manner that conveys the information to the commander in real time as an engagement progresses. In our previous work, we have designed the Advanced Tactical Architecture for Combat Knowledge Systems [6] (ATACKS). ATACKS is a three-dimensional visualization tool that facilitates rapid, flexible development of high-level battlespace representations as well as execution and assessment of wargaming scenarios. Written in Java using the Java 3D API, ATACKS allows the user to create and execute quickly major theatre of war scenarios based on its library of terrain and unit elements and their associated behaviors. It is used to study the synergy between human decision-makers and intelligent visualization systems operating in unconventional military situations such as small-scale contingency operations [6] where the planning and execution processes overlap and conventional military solutions may not be effective.

ATACKS expands standard battlefield symbology by providing abstract symbols on three dimensional (3D) abstract battlespace terrains. It extends normal spatial visualization through process-centered displays that seek to enhance the commander's understanding of the situation by presenting qualitative data in novel formats. In addition, external decision support tools communicate with ATACKS through an application program interface (API) that can allow communication over a network as well as between systems on differing operating systems. With this capability, ATACKS serves as an integration point for intelligent aiding systems.

ATACKS contains three distinct general layers: 3D spatial representation of the situation, process displays that present abstract representations of data, and the decision support layer that can provide and evaluate COAs, as shown in Figure 1. The process displays use the information from this layer to display a variety of abstract information, such as unit effectiveness, impact alertness of an event, or suggested support units. The middle layer manages the data of the current situation. The lowest layer (decision support) passes information about the current state of the battlefield to decision support tools and then forwards the results or interactions required back to the middle layer.

1.1. Visualization engine

The central layer of ATACKS provides a 3D analog of standard military and new, unconventional symbology for contingency operations. By grafting the 2D unit symbols onto 3D abstractions, the system allows commanders to apply existing domain

Proceedings of the 10 th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS'03)
0-7695-1917-2/03 $17.00 © 2003 IEEE

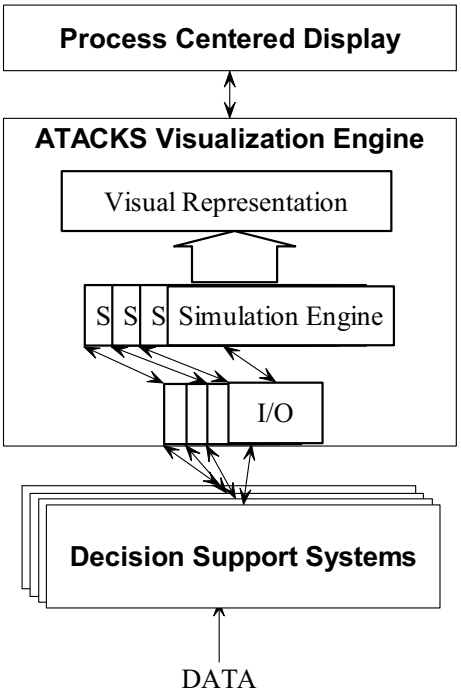

Figure 1. ATACKS' three-layer architecture

knowledge to understand the three-dimensional battlespace representation.

1.2. Configural displays

The top level of ATACKS creates process centered, or configural displays (CDs) which can provide a commander with a more thorough understanding of the current situation by presenting abstract representations of key events as they occur in the battle. Different types of CDs were designed to display various aspects of the wargaming and battle process. The basic CD designed for use in major theatre of war scenarios is shown below in Figure 2.

The chalked rectangular outline delineates the battle grid whose dimensions can be adjusted by the commander through the ATACKS GUI. Also represented in white are the Phase Lines (PL), the Line of Advance (LOA), the Forward Edge of the Battle Arena (FEBA) and other similar Command and Control features. The purpose of these outlines is to provide the viewer of the CD with references to the position and progress of the units along the battlefield. The position of the multi-colored bar is tied to the location of the units on the battlefield. In addition to position, the CD also portrays the Combat Effectiveness of the Blue force and the Red-Blue combat ratio in the case where the friendly unit encounters an enemy. Since a CD is created for each friendly unit before the scenario begins execution, the combined CDs provide an at-a-glance indication of the status and progress of the units according to the battle plan.

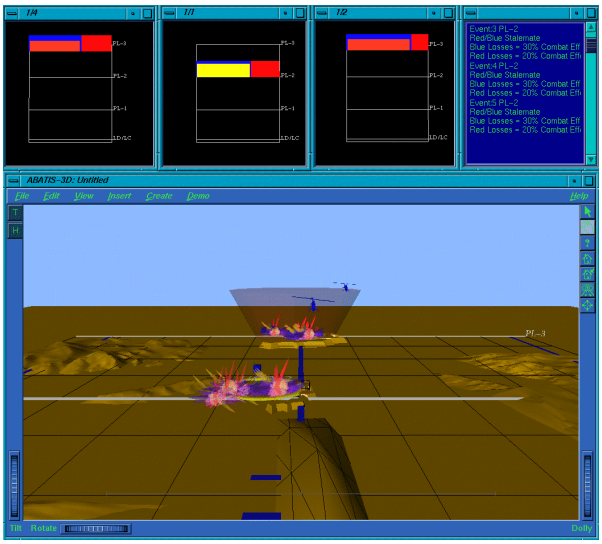

Figure 2. Configural displays and 3D battlespace

1.3. Decision support

By isolating the decision support functionality into one layer or module, ATACKS can include its own decision support software (DSS) or take advantage of external sources such as COTS products. It can then use its object-oriented visualization base to display the data, information, and knowledge derived from these external sources. This interface is defined through an Application Program Interface (API). Examples of such decision tools include FOX-GA [4].

FOX-GA uses a genetic algorithm to generate thousands of COAs and then narrows the choices down to the few best while ensuring that the selected options are sufficiently different from each other. The choices are presented to the user who ultimately decides which COA to select for execution. Most recent development includes Sheherazade, a new system being developed jointly by the University of Arizona and the US Army Research Laboratories. Sheherazade models Operations-Other-Than-War scenarios and coevolved COAs for multiple sides or factions. These COAs can be visualized and analyzed in ATACKS.

2. Mobile Agent Technology based Intelligent Robotic System (MATIX)

2.1. Concept exploration

In addition to the underlying software, we are currently developing a real-time, physical layer of autonomous devices that represent the real-world entities in the tactical environment. This layer is to be implemented by a Mobile Agent Technology based Intelligent Robotic System (MATIX). We introduce the hierarchical control concept in the development of MATIX, in which field robots are subject to the control of their coordinators; these coordinators have their own supervisors, and so on. This hierarchy places ATACKS at the top level. In this paper, however, we will present a simplified two-level system with ATACKS as the direct supervisor for the field robots. Hereinafter, we refer to coordinator as the robot control software that has been integrated into ATACKS. Figure 3 shows the vertical interaction within MATIX and the communication between MATIX coordinator and other parts of ATACKS.

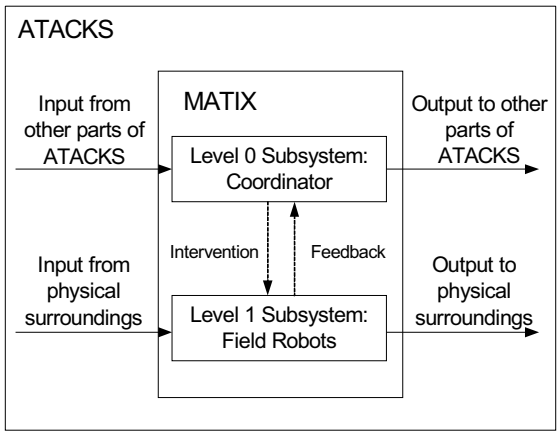

Figure 3. A two-level hierarchical representation of MATIX

Each of the field robots has its own ID and is equipped with wireless transceivers to communicate with ATACKS where it is represented as a visual 3D object. These configurable one-to-one mappings, and the dedicated two-way wireless communication, link the physical entities with the abstract object in the software environment. While commanders manipulate an object in the 3D environment, the corresponding robot carries out the same action and reports what is happening in its physical surrounding back to the commander. This is how the embedded devices are interfaced with software.

2.2. Requirement discovery

We begin with a set of high-level requirements that the MATIX should fulfill. Cost and schedules are general design and realization process constraints. Reductions in weight, size, and power consumption are required in order to enhance efficiency while providing desired functionality and fault tolerance.

The robot shall be able to make basic movements such as going forward, backward, left, right, as well as braking when necessary. The robot makes these moves only when it receives such commands from the coordinator; the environment conditions trigger a certain reaction according to the rule base; or it has been programmed to execute a list of commands that direct the robot to do so.

To be able to synchronize with its "image" in the virtual software environment, the robot shall communicate with ATACKS continuously to exchange their status information in real-time. This communication will preferably follow industry standards to facilitate future upgrading and integration with other computer based system.

It is necessary for the robot to have certain levels of autonomy. That is, computational capability and sensors to perceive its physical environments. Additionally, the software system should be designed to provide such interface that would allow MATIX to be scaled up and down by inserting or extracting a layer of coordinators in and out of the hierarchy.

2.3. Architecture

The design concept and requirements have been embodied in the layered architecture of MATIX robot as shown in figure 4.

2.3.1. Physical layer. This layer handles all physical movements of the robot and collects data from the real time environment by different types of sensors. The track-based robot moves forward, backward and makes turns depending on the combination of the two motors' rotation. Infrared wheel coders count the driving wheel's revolutions in a given period of time and convert these into linear distance, which is further used for location and navigation. Ultrasonic rangers, combined with touch sensors, help to detect objects, avoid collisions and complete tasks related to physical contact.

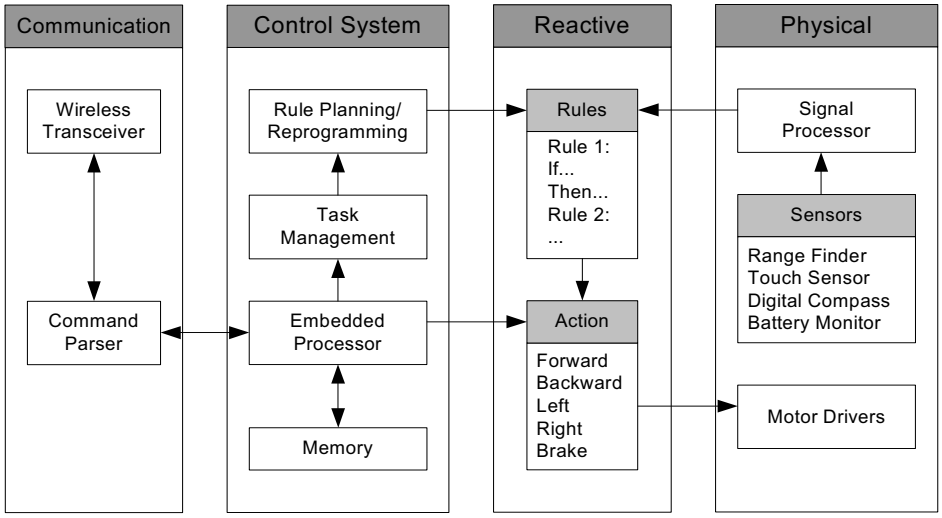

Figure 4. The MATIX robot architecture

2.3.2. Reactive layer. Each robot has a programmable rule stack as the basis for reasoning. The host coordinator also maintains a copy of the rule stack for each robot. The robot reacts to environmental stimuli based on its rule set, with a predefined conflict resolution strategy. If the coordinator needs to change a robot's rule set for certain purposes, it does so on its local copy first and updates the robot's rule stack wirelessly. By this mechanism, the robot could adapt itself to changing environments in real time.

2.3.3. Central control layer. The central control layer consists of an embedded processor, peripherals, and control software. The main program manages tasks, maintains rule stacks, handles interrupts, and accesses memory. If the robot needs to report to the coordinator in operation, the controller will write a command to the communication port to be transmitted. If the coordinator intervenes, the controller should switch to the incoming commands and process them based on the priority order. User codes and data are stored on the Flash ROM, which is reprogrammable through the processor's JTAG port.

2.3.4. Communication layer. The robots and ATACKS communicate through the wireless transceivers, one connected to the ATACKS workstation, and the other being onboard the robot. ATACKS preserves one channel for each robot. A parser encodes data into predefined packets before sending them to ATACKS, and interprets received packets into recognizable commands for the robot controller to execute. By setting a special flag bit, the robot can be toggled between passive and active communication mode. In passive mode, the robot only responds to incoming coordinator commands, but does not automatically report its status to the coordinator as it does in active mode.

3. Implementation

3.1. Function modules

The robot prototype was built in a bottom-up manner: functional modules were built and verified on the development platform, and then a motherboard was designed to hold all the modules seamlessly.

3.1.1. Java Controller Module. The latest embedded Java technology has been chosen in implementing the control system. Compared to other available RTOS and tool sets, the embedded Java excels in the following aspects in addition to the intrinsic advantage of the Java language:

- **Portability**
  By using an underlying Java run-time environment, applications can be easily developed on multiple operating systems with standard software development tools such as JBuilder. Hardware-specific code can be simulated on a desktop system. Then, by taking into account the underlying target hardware characteristics, applications can be moved with minimal effort to the specific target device.

- **Software reuse**
  Because ATACKS is a pure Java system, and both

the robot and coordinator have a similar communication layer, code written for the coordinator could be migrated to the robot without bothering with other languages. A native interface is also unnecessary.

The module we use in the robot consists of a native Java processor that runs a Java Virtual Machine and user codes stored in Flash ROM. Figure 5 shows how this embedded Java environment differs from the OS-based Java environment.

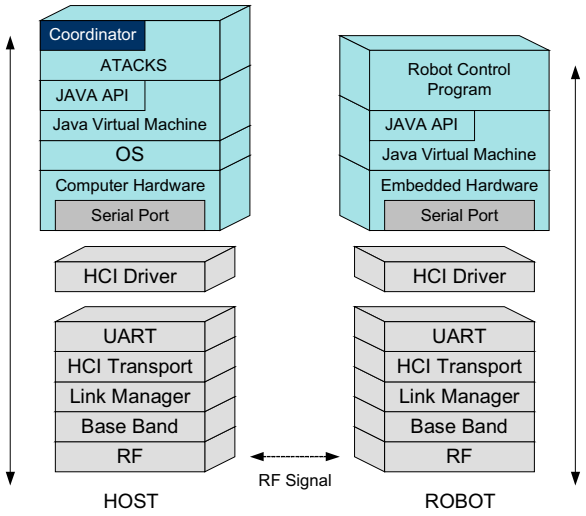

Figure 5. Data flow between MATIX robot and the coordinator in ATACKS

The processor supports Real Time Specification for Java and Java2 Micro Edition. The J2ME Connected, Limited Device Configuration provides a set of APIs tailored for embedded applications. The command parser in the communication thread passes control commands to the program no matter the robot is in standby or operation. The main program has a communication thread that exchanges real time information with the coordinator. It also contains a task manager agent, a runtime rule manager agent, and interrupts service subroutines for various sensors. All these parts are coded in Java conforming to J2ME CLDC, and executed by the native Java processor at 80MHz.

3.1.2. Bluetooth transceiver module. Bluetooth is a fast growing new technology for wireless interconnectivity. It is a universal radio interface in the 2.4 GHz frequency band that allows embedded devices to communicate via short-range, ad hoc networks. Bluetooth is based on a highly integrated low power chip, which delivers modest transfer rate for both data and voice. These features make it an ideal option in the battery-operated MATIX robots.

Bluetooth devices contain four major parts: a radio (RF) that transmits and receives data and voice, a baseband or link control unit that processes the data, link management software that manages the transmission, and supporting software. HCI (Host Control Protocol) is used to provide an interface between the Bluetooth hardware and a host via a physical connection, either UART or USB. In the MATIX design, we use UART connection for both the robot and the host computer. Figure 5 outlines the application framework and shows the data flow between the robot and its coordinator resident in ATACKS.

3.1.3. Ultrasonic Ranger Module. The ultrasonic range finder offers precise ranging information at distances from several inches to several meters. It transmits a sonar pulse outside of the frequency of human hearing. This pulse travels at the speed of sound away from the ranger in a cone. The ranger pauses for a brief interval after the sound is transmitted and then awaits the reflected sound in the form of an echo. When the controller requests a ping, the ranger creates the sound and waits for the echo. If received, the ranger reports to the control, which then computes the distance to the object based on the elapsed time. Two IO ports of the controller are dedicated for the ranger operation, one for pulse trigger input, and another to take echo output.

3.1.4. Battery Monitor Module. Power management and monitoring is another important issue in embedded devices. Traditional capacity monitoring schemes work by comparing the battery voltage to the rating value. However, this method lacks the required accuracy because many batteries have a relatively flat voltage profile in both charge and discharge directions. To make matters worse, the available capacity also changes as a function of self-discharge, cell aging, temperature, and discharge rate or profile. In order to provide precise capacity tracking and monitoring for the robot, the system uses a new battery monitor chip based on charge to voltage converter (VFC). The chip counts coulomb transferred through a low-value sensor resistor and stores the information into its registers. The Java processor then retrieves this information through the one-wire interface, thus getting the accurate and reliable information about the battery's state of charge by simple calculation.

In addition, this system has several other modules such as the power regulator, the RS232 level converter,

and the motor drive that are important components to make the robot functional.

3.2. Test and Integration

All of these above-mentioned modules have been developed and tested separately on a demo board. Although each of the modules has been verified to be working, problems occurred when they were put together onto the development station. Interrupt priority levels had to be rearranged, packet length and communication interval had to be adjusted, and other conflicts had to be solved before the final PCB board could be manufactured to fulfill every aspect of the designed functionality.

4. Experimental Results

In order to test the effectiveness of the added MATIX, we simulated a simplified peace-keeping mission in ATACKS. The operation was to send a team to rescue several hostages behind a building with a known target position. The scenario was created in ATACKS with different 3D objects representing the target, the hostile units, and the team. The commander planned a route for the team, and sent the path information to the robot representing the team. When the commander launched the scenario in ATACKS, the robot simultaneously started off along the path. Figure 6 shows a screenshot taken during the process of the experiment. The top window contains the robot control panel that shows the real-time status of the robot and the environment information sent back by it. The ellipse in the middle of the panel indicates the distance of the object ahead of the robot on a logarithmic scale.

Not only does the size of the eclipse vary according to the ultrasonic echo readout, but also does the color. Blue represents the distance greater than 3 meters; red represents the distance shorter than 8 inches; yellow represents any distances in between. On the sides of the radar screen are two odometers for the left and right tracks. The bar at the right bottom corner of the panel shows the remaining battery capacity, which updates itself every minute. Above the fuel gauge are switches for choosing communication modes and keyboard navigation modes. By selecting or deselecting these radio buttons, internal commands are sent to the robot and thus change its behavior. The scrollable pane at the left end of the panel logs communication between the robot and the commander.

Below the robot control panel is the ATACKS GUI that consists of two separate windows. The right ones provides drop down and pop-up menus for users to choose runtime engines, create scenarios, manipulate 3D objects, and place paths and lines. It also displays the execution of the scenario. On the left is the object property page that allows user to change object

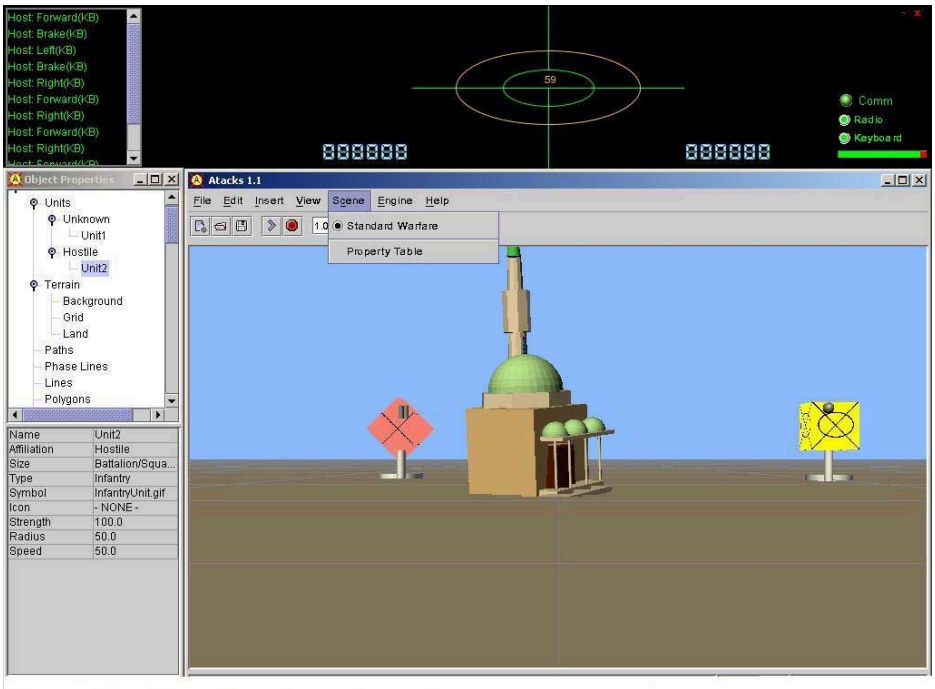

Figure 6. Screenshot of a running scenario

attributes such as unit affiliation, size, speed, terrain color, line weight, etc.

In this specific experiment setup, the robot is programmed to report its status to the commander every 300 milliseconds. The robot will automatically avoid obstacles according to its rule set. Should any object exist within 4 inches of the front of the robot, it will trigger a warning dialog for the commander to take further actions. In the actual experiment, the commander successfully received updated information collected by the robot's sensors, and was able to update the robot's rule set, part of which defines how the robot should avoid an obstacle. When the cooperator of the experiment placed an object within the warning range of the robot, it braked as programmed and wirelessly triggered the dialog for commander's decision making.

5. Conclusions

This paper presents the design of MATIX, a physical layer to interface with a battlefield simulation system. MATIX provides real-world information to a commander who verifies a plan created in a virtual environment. One of MATIX's most unique features is its ability to communicate bi-directionally with ATACKS so that the physical robot and its image can stay synchronized in real time.

References

1. K. Buchenrieder and J.W. Rozenblit, "Codesign: An Overview", in J.W. Rozenblit and K. Buchenrieder (Eds) *Condesign: Computer-Aided Software/Hardware Engineering*, pp.1-16, IEEE Press, 1994

2. J. Borenstein and J. Evans, "The OmniMate Mobile Robot Design, Implementation, and Experimental Results", *Proceedings of the IEEE International Conference on Robotics and Automation*. Albuquerque, NM, pp. 3505-3510, April 1997.

3. R. Bischoff, "Design Concept and Realization of the Humanoid Service Robot HERMES", in A. Zelinsky (ed.), *Field and Service Robotics*, pp.485-492, Springer, London, 1998.

4. C.C. Hayes, J.L. Schlabach, C.B. Fiebig, "FOX-GA: An intelligent Planning and Decision Support Tool. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 3, 2454-2459, Oct 1998.

5. M.D. Mesarovic, D. Macko and Y. Takahara, "Theory of Hierarchical, Multilevel, Systems", Academic Press, 1970.

6. F. Momen, J.W. Rozenblit, L. Suantak and M. Barnes. Three Layer Architecture for Continuous Planning and Execution. *Proceedings of the 2001 Army Research Laboratories Symposium*, College Park, MD, March 2001.