Automated SAF Adaptation Tool (ASAT)

Roy Stripling[1], Joseph T. Coyne[1], Anna Cole[2], Daniel Afergan[2], Raymond L. Barnes[2], Kelly A. Rossi[2], Leah M. Reeves[3], and Dylan D. Schmorrow[4]

[1] US Naval Research Laboratory, 4555 Overlook Ave, SW, Washington, DC 20375
[2] Strategic Analysis, Inc., 3601 Wilson Blvd, Suite 500, Arlington, VA 22201
[3] Potomac Institute for Policy Studies, 901 N. Stuart St, Suite 200, Arlington, VA 22203
[4] Office of Naval Research, 875 North Randolph St, Arlington, VA 22203
{roy.stripling, joseph.coyne}@nrl.navy.mil,
{acole, dafergan, lbarnes,krossi}@sainc.com,
lreeves@potomacinstitute.org, schmord@onr.navy.mil

Abstract. The purpose of this paper is to describe a new, user-friendly tool that will enable researchers and instructors to setup and run virtual environment scenarios that adapt to the VE user's real-time performance and cognitive status. This tool, the Automated SAF Adaptation Tool (ASAT), will work with existing performance and cognitive state assessment software, and with existing semi-automated forces (SAF) behavior engines. ASAT will collect processed performance and cognitive state data from the assessment software and trigger SAF behavior setting manipulations that were pre-selected by the SAF operator. A key feature of ASAT is the ability to setup and execute these real-time manipulations without the need to alter code in either the assessment software or the SAF engines.

Keywords: semi-automated forces, SAF, real-time, adaptation, virtual environment, training, cognitive state.

1 Introduction

At present, Virtual Environment (VE) training packages do not support adaptive training – the ability of the system to modify the ongoing tactical scenario based on real-time assessments of the trainee's performance and/or cognitive state. It has been hypothesized that the ability to execute such "on-the-fly" modifications, if properly implemented, would enhance training experiences by optimizing the trainee's cognitive state for training and by tailoring the training experience to individual's or team's evolving skill level [1, 2, 3]. However, currently there is no adequate or practical adaptive training system in which to test this hypothesis.

To date, our work with adaptive VEs has focused on military infantry training systems. These systems leverage complex behavior-model architectures to provide "semi-automated" behaviors for the computer generated forces (CGF). The most commonly used behavior engine for military simulation and VE training is Joint Semi-Automated Forces (JSAF). Newer SAF architectures are also in development. We have worked with both JSAF and OTBSAF (the testbed for the system that will

D.D. Schmorrow, L.M. Reeves (Eds.): Augmented Cognition, HCII 2007, LNAI 4565, pp. 346–353, 2007.

be released as OneSAF). A detailed description of these SAF architectures is beyond the scope of this manuscript. However, it is useful to point out that they allow the SAF operator to set up scenarios by placing CGF individuals, squads, and vehicles throughout the virtual environment. The SAF engines also provide complex Graphical User Interface (GUI) controls and menus, which allow the operator to adjust a variety of behavior-relevant settings including CGF competency, range of engagement, underlying cognitive model, and response to particular events such as gunshots. Manipulating these and other attributes during a scenario and in response to the level of performance or to the cognitive state of individual or teams of trainees would constitute adaptive training. An operator using the SAF's GUI could manipulate these settings, but the complexity of the GUI and the limited rate of human processing and motor abilities limit the ability of the operator to implement these changes in real-time. Additionally, the SAF operator cannot collect, process, and interpret the many streams of behaviorally and physiologically based performance data that reflect trainees' cognitive status and skill level. Automated systems that do assess these aspects of trainee performance have been developed and are the continued focus of additional research. Detailed description of such measures and systems are also beyond the scope of this paper. However, such measures include measures of trainee arousal [4], and engagement [5, 6] among others. The key tool that is missing before robust user-friendly closed-loop training environments can be produced is an automated method for triggering adaptive manipulations within complex SAF applications.

We recently demonstrated that it is possible for an automated system to collect, process, and trigger adaptive manipulations in a SAF application (OTBSAF) in real-time. For this demonstration, we collected ECG (electrocardiographic data) from two-man teams immersed in a virtual infantry trainer. The ECG data was processed in real-time to derive a measure of physiological arousal using the methods described by [4]. Persistent high or low arousal levels from individuals or from the two man team triggered a variety of automated behavior setting changes in OTBSAF, including, at its most extreme, activating and sending in an increasing number of enemy CGFs. These manipulations in the SAF settings were transmitted via UDP messages to a Linux computer running OTBSAF. This approach, however, left much to be desired and was found to be impractical for several reasons. First, it did not produce stable behavior in JSAF, the SAF engine more commonly in use today. Second, it required source-code access and modifications to the SAF environments. Third, it required several weeks of iterative coding to produce a small set of potential manipulations that could be triggered. Fourth, hard-coding the manipulations limited the range of scenarios that could be generated for use in the closed-loop training demonstration. The last of these problems is particularly worrisome because it restricts the flexibility of the training system, rendering it obsolete as soon as new enemy tactics or scenarios for training are desired. Collectively, these problems combine to produce a very cumbersome and user-unfriendly environment in which to conduct research and training.

We are now pursuing a new approach that will enable the rapid development of a large set of operator-defined manipulations without requiring code changes to the SAF applications. This approach will also work with independently-developed software products that produce real-time behaviorally- and/or physiologically-derived

measures of performance, and will work with any military training or mission rehearsal scenario that can be developed in the SAF applications. The end product of this effort will be a sufficiently user-friendly executable that can be used for a broad variety of research and training operations. Of particular importance, with further development this application will also be able to incorporate methods for auto-selecting unique combinations of automated manipulations based on educationally-sound logic rules. These logic rules would specify the type and timings of manipulations that are appropriate in the current training context AND in the context of the trainees' performance and cognitive status. The development of educational logic rules is beyond the scope of the current effort.

In the near term, this approach is expanding on available open-source code to develop the Automated SAF Adaptation Tool (ASAT). ASAT, functioning as a stand-alone executable, will reach across platforms to receive behaviorally- and/or physiologically-derived measures of performance and activate, at the appropriate time, user-defined, pre-recorded SAF manipulations based on either user- or vendor-specified performance thresholds. ASAT currently collects performance data by reading such data from ASCII formatted files or by reading the data from user-specified fields displayed on the performance assessment application's own GUI. In the future, ASAT will also collect performance data via networked transmission from these applications that conform to ASAT API specifications,

2 Performance Assessment Input Module

One of the key features of the ASAT tool is the ability to interact and use data from a range of real-time performance and physiological sources. The current ASAT design allows for data to be collected from a user-specified source through one of two methods. These methods include reading data from an ASCII (American Standard Code for Information Interchange) file or reading data from a program developed in the LabVIEW programming environment (National Instruments). Both methods allow for the data to be read by ASAT as it is being recorded to a file or updated on the LabVIEW GUI.

The majority of systems that collect continuous physiological data append the information to a file for later analysis as the information is collected. This data may be collected in application specific files or in files with standardized file formats. ASCII data is a common format for this type of data, and ASAT seeks to exploit this feature by allowing data to be read in from an ASCII file source. ASAT will perform this function by copying the data file and reading the last lines of the file. The ASAT system will not read from the original data file to reduce the chances of errors and file corruption associated with multiple programs reading and writing to the same file at the same time. ASAT allows users to specify how often they would like to read the performance data into the system (which should be dependent on how frequently the source application generates the data and saves it to file).

The ASCII file input component of the ASAT tool will help users preview files, select the appropriate data column to be used and also store the preferences for later use. The system also allows more advanced users to use the stored settings to

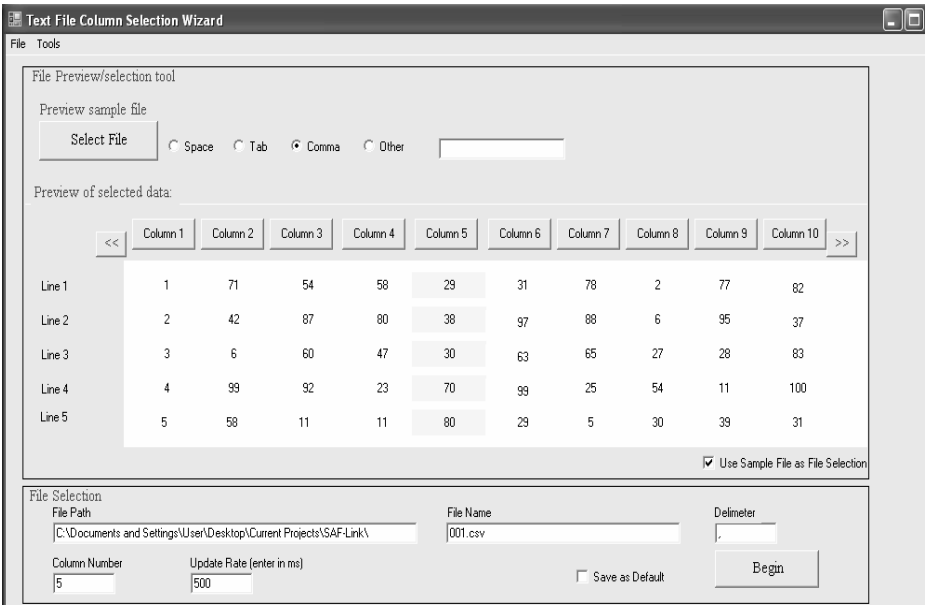

Fig. 1. Graphical interface for inputting an ASCII file into ASAT

minimize the set up time. An advanced user may only need to enter the new file name associated with the data to setup the data input. Figure 1 below presents a screen capture depicting the file preview tool used to select and input ASCII data into the ASAT package.

The second source of performance or physiological data that ASAT is currently being configured to use as an input source is data from a LabVIEW program. The LabVIEW interface requires minimal system resources compared to the ASCII tool and is a common programming environment for many researchers developing performance assessment tools.

As shown in Figure 2 the user specifies the LabVIEW program name and reference object within LabVIEW. Similar to the ASCII interface, the user can also specify the update rate used to import the data and can pull data in from LabVIEW faster than the

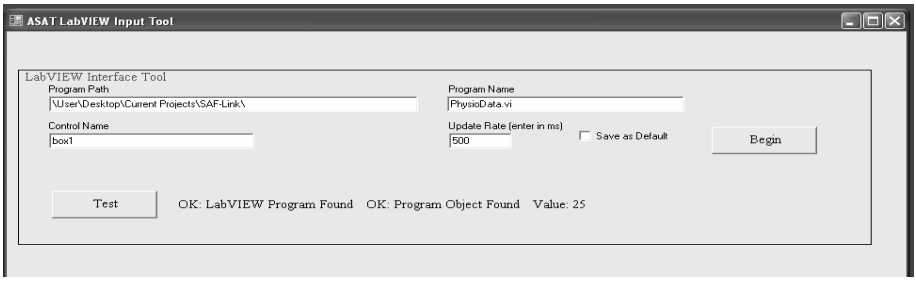

Fig. 2. Graphical interface for importing LabVIEW data into ASAT

ASCII method. The LabVIEW tool also allows the user to test the link between ASAT and LabVIEW to ensure the correct data is being imported.

The combination of the ASCII and LabVIEW methods for importing performance or physiological data allows ASAT to interact with a wide range of already developed software. The current version of ASAT allows for only one data field to be used, but future versions will be able to import data from multiple sources.

3 GUI Interface

Once performance assessment data is collected by ASAT it can be used to trigger changes in SAF entity behavior settings. For this aspect of ASAT to function, the user must first develop and save SAF manipulation Macros, then indicate when these macros should be executed by ASAT by setting up logic rules through the ASAT GUI.

The ASAT GUI is responsible for the initialization of the performance assessment input module described above and for the recording of user-created macros that will affect the SAF program. When the operator launches ASAT, the GUI (Figure 3) opens with an embedded SAF application (having been previously launched by the user on a networked computer). The GUI launches a sidebar along the left side of the screen, while the SAF program will take up the rest of the screen. Besides opening the performance assessment input module, or "logger", the GUI allows the user to create, store, and load behaviors that can be implemented when the scenario reaches certain conditions.

A macro is a mechanism that records a series of user inputs events (mouse moves, mouse clicks, and keyboard presses) as a single executable series of events that can later be played back by the computer. This allows the automated adjustment of SAF entity behavior settings through the SAF GUI at a faster rate and with greater repeatability than an actual user could achieve.

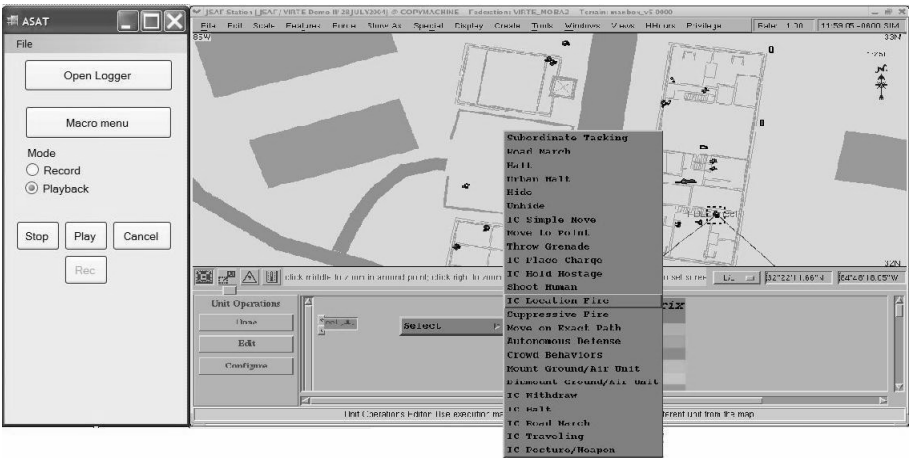

Fig. 3. ASAT GUI with JSAF running

ASAT also offers a macro manipulation window (Figure 4) that allows the ASAT operator to establish logical rules that determine when to execute previously created macros. Users create conditional logic statements by selecting either a physiological or performance variable and an operator, and then entering a threshold value into the text field. After they have created a conditional logic statement, they specify which macro should be run when the conditions are met. The window gives the option to add more rules to the equation (using statements such as AND, and OR) or to finish the equation. Previously created rules are displayed in a panel at the top center of the window. By combining the two menus, the operator is able to record actions and then play them back on command or automatically during certain circumstances. This allows the actions to be identical each time and eliminates many outside variables involved in virtual environment training.

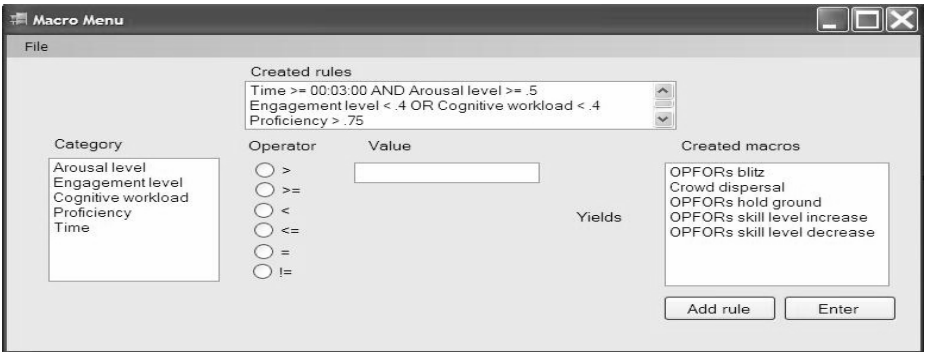

Fig. 4. ASAT Macro manipulation window

4 Macro Recording and Execution

To enable ASAT to carryout its task in a user-friendly manner we are leveraging an open-source network remote control application called Virtual Network Computing (VNC). VNC was first developed by AT&T Laboratories in Cambridge, and is now maintained and further developed by RealVNC in the UK (http://www.realvnc.com/). VNC is open-source software that can simultaneously run on multiple operating systems and enable any computer on a network to view and remotely control the GUI of another computer connected to the same network. VNC uses the remote framebuffer (RFB) protocol to transmit keystrokes and mouse events, along with screen updates between a VNC server and a VNC viewer (Richardson 2006). In the context of the ASAT application, the VNC server is the Linux-based machine that has the ability to generate and change VE scenarios using a SAF application. The ASAT GUI, running on Microsoft Windows, has a VNC Viewer control that allows the user to create a sequence of actions (macros) in JSAF that will automatically execute once cues are received from the performance assessment logger module (as described above).

VncSharp, an open-source client library and custom control, is being utilized to embed a VNC Viewer and add VNC functionality to the ASAT application.

VncSharp was written by David Humphrey, Matt Cyr and Chuck Borgh at Seneca College as a project under the Centre for Development of Open Technology initiative (http://cdot.senecac.on.ca/projects/vncsharp/index.html). It is an implementation of VNC's RFB protocol written in Microsoft's C# language for the .NET framework. VncSharp provides a conduit for ASAT to programmatically take advantage of the methods and properties provided by VNC while writing software in the Visual Studio .NET development environment.

When ASAT is launched, the panel on the right of the ASAT GUI shows the remote machine's GUI (JSAF, for example). This panel is the VncSharp Windows control, which serves as the VNC Viewer for ASAT. The VncSharp Windows control screen is continually updated, and all keystrokes and mouse events in this ASAT panel are transmitted back to the JSAF machine. This allows the user to create a scenario in JSAF (running on a remote machine) from ASAT, in addition to making edits to existing JSAF scenarios. When the Macro Record screen is active, ASAT has a procedure independent of VNC that programmatically taps into the Microsoft Windows messaging system and logs, filters and stores all of the keyboard and mouse messages in a custom file format that can later be referenced to execute or "playback" the macro by acting upon the SAF GUI in the VNC Viewer.

After the user creates and saves a custom macro, he can then navigate to the Macro Menu window from the main ASAT GUI to define rules and associate macros with those rules. For each rule, a particular condition must be met before a chosen macro is executed (i.e. if the Arousal level of the participant is < 0.5, execute the OPFORs blitz macro). When ASAT is active during a simulation, the performance assessment input module continually polls the data file to determine if and when the condition in the user-defined rule(s) are met. If they are, the macro module accesses the macro log file created during the macro record procedure for the given rule. The macro module then directs the focus of the program to the VNC Viewer and automates keyboard and mouse events using Microsoft Win32 API's and a Microsoft .NET Framework Class Library, SendKeys. When the mouse and keyboard events are automated and sent to the VNC Viewer within ASAT, they are directly transmitted to the JSAF machine to alter the training scenario based on performance and neuro/physiological measures acquired during the simulation.

5 Conclusions

ASAT will provide a means of rapidly developing robust closed-loop integrated training systems that can continue to leverage the well-developed SAF architectures that currently drive many military training simulations. In doing so, it will provide a robust tool for testing many worthwhile hypotheses regarding the use of Augmented Cognition approaches in training and educational environments. For example, ASAT can support the development and testing of training systems which sense an individuals attention levels and then adjust the VE to maintain those attention levels within a specified range. Of broader significance, ASAT is not conceptually restricted to use with SAF architectures, but should work with any GUI based application that enables human operators to modify the application's settings during

operation. Thus the ultimate impact of ASAT may reach far beyond the military training and education communities.

References

1. Schmorrow, D., Cohn, J.V., Stripling, R., Kruse, A.A.: Enhancing Virtual Environments Using Sensory-Multiplexing. At: Interservice/Industry Training, Simulation and Education Conference (I/ITSEC) Annual Meeting, Orlando FL (2004)
2. Raley, C., Stripling, R., Kruse, A., Schmorrow, D., Patrey, J.: Augmented Cognition Overview: Improving Information Intake Under Stress. In: Proceedings of the Human Factors and Ergonomics Society Annual Meeting (HFES 2004), New Orleans LA (2004)
3. Cohn, J.V., Stripling, R., Kruse, A.A.: Investigating the Transition from Novice to Expert in a Virtual Training Environment using Neuro-Cognitive Measures. At: 1st Annual Meeting on Augmented Cognition, Las Vegas NV (2005)
4. Hoover, A., Muth, E.: A Real-Time Index of Vagal Activity. Int. J. Human-Computer Interaction 17, 197–209 (2004)
5. Freeman, F.G., Mikulka, P.J., Scerbo, M.W., Scott, L.: An Evaluation of an Adaptive Automation System Using a Cognitive Vigilance Task. Bio. Psych. 67(3), 283–297 (2004)
6. Pope, A.T., Bogart, E.H., Bartolome, D.S.: Biocybernetic System Evaluates Indexes of Operator Engagement in Automated Task. Bio. Psych. 40(1-2), 187–195 (1995)
7. Richardson, T.: The RFB Protocol version. 3.8. In: RealVNC Ltd (October 2006) http://www.realvnc.com/docs/rfbproto.pdf