"Hello Java!" Linking ACT-R 6 with a Java simulation

Philippe Büttner
Technische Universität Berlin
Centre of Human Machine Systems
Franklinstr. 28-29, FR 2-6
10587 Berlin, Germany
pbu@zmms.tu-berlin.de

Abstract

Many more applications and simulations are developed in the Java programming language than in the Lisp programming language. This can be attributed to a number of reasons, including platform independence, object-oriented programming, etc. Due to the fact that the ACT-R software was programmed in Lisp, incompatibility issues between it and Java arose. These issues necessitated the establishment of a tool capable of preventing a Lisp reimplementation of existing Java applications. Consequently, "Hello Java!" was developed to link cognitive models written in ACT-R with Java applications, namely simulations. In order to achieve this, a package must be added to the Java simulation so that it can observe and perform actions on the frame, as well as communicate with the ACT-R software. The line of communication between Java and Lisp is established through a TCP/IP connection. As a result, the simulation and cognitive models can be run on different computers. Since the release of ACT-R 6, the methods for perception and action have been externalized. These externalized methods can be utilized as devices for the ACT-R software, making it possible to, consequently, use a Java simulation as a device for ACT-R.

Keywords: ACT-R; device; Java; simulation; network

Introduction

ACT-R (Anderson & Lebiere, 1998) is a computational theory of human cognition with two separate, but interacting, knowledge stores developed in Lisp. Both declarative knowledge and procedural knowledge are unified into a production system where procedural rules act on declarative chunks. The ACT-R system includes the capability to create simulated environments, such as screen interfaces. Production rules have the ability to interact with this environment by perceiving objects and making motor movements through perceptual and motor buffers.

The externalization of all necessary methods involved in the perception of objects and the facilitation of motor movements makes it possible to extend the environment to a world outside of ACT-R, without requiring a modification of its architecture. An interaction with the production rules can be enabled with any device which meets the specifications of these externalized methods. Due to the fact that Lisp supports communication via the TCP/IP network protocol, it becomes possible for ACT-R to interact with any environment also possessing the capability to communicate via TCP/IP.

I was able to take advantage of this trait and developed "Hello Java!" as an open source tool for linking cognitive models written in ACT-R to applications and simulations written in Java. This tool and additional examples, including the source code, are available on the following website: http://www.zmms.tu-berlin.de/kogmod/tools/hello-java.html

Design

To link a cognitive model written in ACT-R to a frame-based Java simulation two elements that coordinate the interaction between the model and the simulation are required. The first of these elements is a Java package that must be added to the simulation. The second element is a device for the ACT-R software that bridges the incompatibility between Java and Lisp with the TCP/IP network protocol. I will now proceed to describe the communication and synchronization of the elements in detail:

Communication

The line of communication between Java and Lisp is established through a TCP/IP connection, which is a protocol commonly used to connect computers to the internet. All information sent with this protocol can be transmitted as a string representation. This string representation carries data pertaining to perception and action that is exchanged between the cognitive model and the simulation. In order to be able to communicate via TCP/IP, each side must encode and decode information using a common vocabulary and unified grammar. ACT-R receives data pertaining to perception and sends back data pertaining to the execution of motor movements. Java, meanwhile, receives data instructing it to perform the cognitive model's desired action and provides a visual representation of all objects visible in the frame.

By using a network structure it is possible to run a cognitive model and a simulation on different computers. This increases computing power for each, cognitive model and simulation.

Synchronization

In order to keep the cognitive model and the simulation synchronized, the simulation must adjust its cycle speed to that of the cognitive model. The opportunity to do this applies to all clock-controlled simulations. A clock-controlled simulation updates all elements visible on the screen after one cycle. The length of a cycle depends on the time resolution of the simulation. The shorter the cycle, the smoother a simulation appears to be.

ACT-R is responsible for adjusting the cycle speed of the simulation. It accomplishes this by synchronizing the execution of the simulation's cycles with the computed time interval of the cognitive model. ACT-R's scheduler is used to trigger the cycles in a calculated time interval.

As an example, let us assume that the cycle length of a clock-controlled simulation is one second. The scheduler will also be adjusted to one second and it assumes control of the cycles of the simulation. Due to the fact that the time interval of the scheduler is synchronized with the computed time interval of the cognitive model, the same time will elapse for the simulation and the cognitive model.

Extending the Java simulation

Java has established itself as a widely used, universal, platform-independent programming language. In order for ACT-R to be able to access a Java simulation, the simulation itself must be extended by a package. In general, a Java package contains classes, methods and functions that extend an application. This package coordinates the functions of observing all objects from the simulation, performing actions on the simulation, and exchanging information with ACT-R. In order to run the package, one must first add it to the simulation and initialize it. The package consists of three sub-packages. Their descriptions and roles are listed below:

Robot

This sub-package triggers actions like clicking a mouse button, moving the mouse, stroking a key, moving the attention pointer and speaking, if the cognitive model decides to perform one of these actions.

GUI

GUI contains the information of all objects visible in the frame. By recursively accessing objects in the frame, data pertaining to the following objects becomes accessible: labels, text fields, buttons, radio buttons and toggle buttons. In principle, every Java object can be accessed. Therefore, information pertaining to the kind, value, colour, size and relative position of an object must be encoded to a string representation. This information is necessary for the visual icon of ACT-R. An object can be one of the following types: text, line or oval. Because ACT-R interprets an oval as a button, every type of button is assigned as oval.

Net

This sub-package provides all methods responsible for encoding and decoding information and handles the network connection with ACT-R. A socket process is started that waits for a connection from ACT-R on a predefined port. If the socket process receives information, it proceeds to parse it, thereby enabling it to perform actions based on methods obtained from the robot sub-package. Furthermore, it can send back visual information from the GUI package.

ACT-R device

As described above, ACT-R 6 provides externalized, accessible methods responsible for the perception of objects and the execution of motor movements. These methods can be implemented and utilized as a device, resulting in an interaction between this particular device and the production rules. In order to be able to link ACT-R to a Java simulation, it was necessary to implement the following methods:

- device-move-cursor-to: ACT-R sends an action to the Java simulation to move the mouse pointer to a given position.
- device-handle-click: ACT-R sends an action to the Java simulation to perform a mouse click.
- device-handle-keypress: ACT-R sends an action to the Java simulation to perform a keystroke.
- device-speak-string: ACT-R sends an action to the Java simulation to speak a string.
- get-mouse-coordinates: ACT-R sends a request to the Java simulation to gather data pertaining to the position of the mouse.
- build-vis-locs-for: This method updates the visual icon of ACT-R's vision module with all visible objects of the device. It will be invoked after the proc-display command is called.
- device-update: This method is called after ACT-R computes one cycle. At this point it is optional to update the visual icon of ACT-R's vision module or to perform other tasks.

Updating the visual icon of the vision module

To update the visual icon, it is necessary to call an update-method. This method regulates the gathering of visual information, but is not one of ACT-R's device-methods. The update-method sends a request to the Java application instructing it to collect data pertaining to all visible objects present in the simulation. Once this visual data is transmitted to ACT-R, it is then written into the visual icon of ACT-R via the "build-vis-locs-for" method. The visual icon provides information about the kind, value, colour, size and relative position of visible objects in the environment. This method responsible for updating the visual icon can be triggered in two different ways:

- By utilizing a scheduler corresponding to a regular time interval, resulting in reduced network traffic for longer time intervals. A scheduled update is especially practical on clock-controlled simulations in which the simulation updates the screen after a certain interval.
- By allowing every instance in which ACT-R calls the "device-update" method to serve as a trigger. Although this method ensures that the visual icon is always up-to-date when being accessed by ACT-R, it incurs higher network traffic costs.

Discussion

"Hello Java!" was developed as a tool to directly interface ACT-R with an external system. Its defining trait is its ease of use resulting from the fact that no modification of the simulation is required, that no restrictions are imposed upon the model and that it is possible to synchronize the simulation with the model.

References

Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought.* Mahwah, NJ: Erlbaum.