5-2001

Multi-Tasking and Cognitive Workload in an ACT-R Model of a Simplified Air Traffic Control Task

Christian Lebiere
Carnegie Mellon University, cl@cmu.edu

John R. Anderson
Carnegie Mellon University, ja@cmu.edu

Daniel Bothell
Carnegie Mellon University, db30@andrew.cmu.edu

Published In
Proceedings of the Tenth Conference on Computer Generated Forces and Behavior Representation, 2001.

To appear in the Proceedings of the Tenth Conference on Computer Generated Forces and Behavioral Representation

Multi-Tasking and Cognitive Workload in an ACT-R Model of a Simplified Air Traffic Control Task.

Christian Lebiere
Human-Computer Interaction Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
412-268-5920
cl+@cmu.edu

John R. Anderson
Dan Bothell
Psychology Department
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
412-268-2788, 412-268-3323
ja+@cmu.edu, db30+@andrew.cmu.edu

Keywords:
HBR, HLA, ACT-R, Cognitive Architectures, Human Performance Models, Multi-Tasking, Cognitive Workload

ABSTRACT: *We present a human performance model of operator control of a simplified air traffic control task developed under the Agent-based Modeling and Behavior Representation program (AMBR). The model was implemented using the ACT-R architecture of cognition. Using a well-developed cognitive architecture provided a number of benefits:*

1. *The reuse of common design patterns, such as unit task decomposition and retrieval/computation dichotomy, greatly simplified and accelerated model development.*
2. *The model inherited parameter values from previous models and required no fine-tuning of parameters.*
3. *The architectural learning mechanisms provided automatic learning of situations.*

The resulting model was quite simple, consisting of only five declarative chunks of information and 36 production rules, while providing a highly accurate model of human performance. The model matched a wide range of performance measures, including amount and type of errors, response latency and choice percentages. Performance variability is a fundamental aspect of human behavior in complex task. The model accounted not only for the average but also for the distribution of performance through the fundamentally stochastic nature of the architecture, amplified by the interaction with the dynamic simulation environment. Although ACT-R is a goal-directed architecture, the model provided a straightforward account of multi-tasking behavior through the use of interruptions triggered by the onset of messages. The model also provided a theory of cognitive workload based on architectural primitives.

The port of the model to the High Level Architecture (HLA) was relatively simple and straightforward. Providing a stochastic model of behavior, however, put strong demands on the efficiency of the simulation. The implications of such demands for HLA-enabled human performance models are discussed.

Introduction

While much progress has been made in the field of cognitive modeling over the years, theories and systems are too often applied solely to phenomena for which they were specifically designed and are too seldom challenged to step outside of those comfortable boundaries. They tend to evolve independently, staking out their pieces of territory and learning little from each other. The AMBR

Model Comparison [1] challenges those boundaries by making those models account for complex patterns of human behavior on common, challenging tasks. The comparison is structured as a sequence of tasks each designed to emphasize a specific aspect of human behavior. The behavior targeted in the first phase was multi-tasking. Pew and Mavor [2] describe multi-tasking as "clearly relevant to military simulations and to human performance generally" but note that "the relevant theories and models are not well developed or validated, and the computational models are somewhat ad hoc." The task designed to elicit complex multi-tasking behavior is a synthetic air traffic control simulation implemented in the Distributed Operator Model Architecture D-OMAR [3].

Our model was implemented using the ACT-R cognitive architecture [4]. Although the model is quite simple and was developed in a short time frame, it successfully accounted for a broad range of subject data, including aggregate performance, individual differences in performance, response times, choice percentages and amount and type of errors [5]. Even though ACT-R is a strongly goal-centered architecture, we could reproduce the subjects' pattern of multi-tasking behavior by adding to the architecture the ability to detect event onsets and interrupt the current task in favor of a more urgent event. We also added to the architecture a definition of cognitive workload based on existing architectural primitives, and the model's measure of cognitive workload closely matched the subjects reported values.

We report on the adaptation of the model to a version of the simulation using the High Level Architecture (HLA). [6]. The port of the model was relatively straightforward and transparent, but it raised important questions regarding the efficiency of simulations involving high-fidelity models of human behavior. We will discuss those questions after we briefly introduce the ACT-R theory and describe the model and its results.

ACT-R

ACT-R is a production system theory that tries to model the steps of cognition by a sequence of production rules that fire to coordinate retrieval of information from the environment and from memory. It is a cognitive architecture that can be used to model a wide range of human cognition. It has been used to model tasks as simple as memory retrieval [7] and visual search [8] to tasks as complex as learning physics [9] and designing psychology experiments [10]. In all domains, ACT-R is distinguished by the detail and fidelity with which it models human cognition. It predicts what happens cognitively every few hundred milliseconds in performance of a task. ACT-R is situated at a level of aggregation above basic brain processes but considerably below significant tasks like air-traffic control. The newest version of ACT-R has been designed to be more relevant to tasks that are being performed under conditions of time pressure and high information-processing demand.

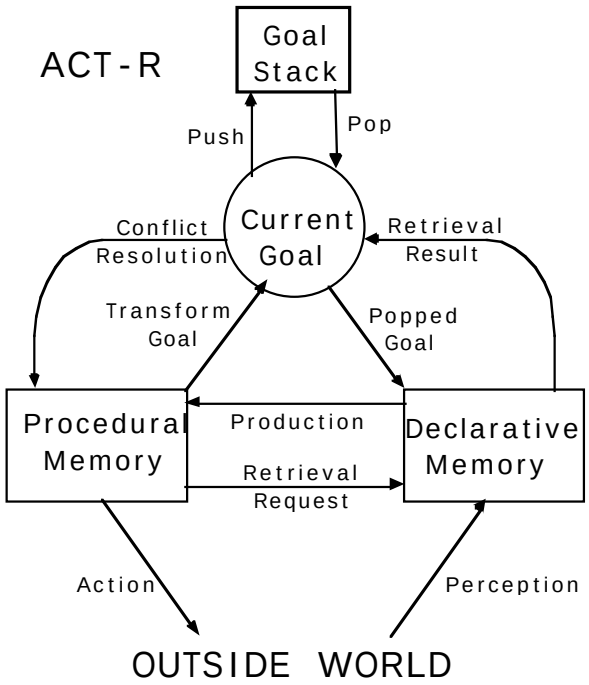

Figure 1: Overall flow of control in ACT-R.

Figure 1 displays the information flow in the ACT-R system. There are essentially three memories -- a goal stack that encodes the hierarchy of intentions guiding behavior, a procedural memory containing production rules, and a declarative memory containing chunks. Productions are condition-action pairs that determine which basic cognitive actions can be taken and when. Chunks are knowledge structures holding a small set of elements (e.g. 3+4=7) in labeled slots. Access to these memories is coordinated around the current goal that represents the focus of attention. The current goal can be temporarily suspended when a new goal is pushed on the stack. The current goal can be popped in which case the next goal will be retrieved from the stack. Productions are selected to fire through a conflict resolution process that chooses one production from among the productions that match the current goal. The selected production can cause actions to be taken in the outside world, can transform the current goal (possibly resulting in pushes and pops to the stack), and can make retrieval requests of declarative memory (e.g., "what is the sum of 3 and 4?"). The retrieval result (e.g., "7") can be returned to the goal. The arrows in Figure 1 also describe how new declarative chunks and productions are acquired. Chunks can be

added to declarative memory either as popped goals reflecting the solutions to past problems or as perceptions from the environment. Productions are created from declarative chunks through a process called production compilation which takes an encoding of an execution trace resulting from multiple production firings and produces a new production that implements a generalization of that transformation in a single production cycle.

ACT-R also has a subsymbolic level in which continuously varying quantities are processed in parallel to produce much of the qualitative structure of human cognition. These subsymbolic quantities participate in neural-like activation processes that determine the speed and success of access to chunks in declarative memory as well as the conflict resolution among production rules. ACT-R also has a set of learning processes that can modify these subsymbolic quantities.

The activation of a declarative memory chunk determines its availability. The context activation is a function of the attentional weight given to the current goal, which is thought to provide an individual difference parameter of working memory [11]. The base level activation of a chunk is learned by an architectural mechanism according to Bayesian statistics to reflect the past history of use of the information contained in the chunk. As Anderson and Schooler [12] have shown, this learning produces such basic features of human cognition as the Power Law of Forgetting [13] and the Power Law of Practice [14].

When trying to retrieve a chunk to instantiate a production, ACT-R selects the chunk with the highest activation. That activation includes a random component that provides stochasticity to memory retrieval and hence to the model's behavior (e.g. [15]), as well as a similarity-based matching component, which provides generalization and robustness (e.g. [16], [17]). Thus, ACT-R is capable both of errors of omission, in which a chunk cannot be retrieved because its activation cannot reach a threshold, and errors of commission, in which the wrong chunk is retrieved instead of the correct one (e.g. [18], [19]). The retrieval time of a chunk is an exponential function of its activation, providing a fine-grained account of the time scale of memory access. The total time of selecting and applying a production is determined by executing the actions of a production's action part, whereby a value of 50 ms is typically assumed for elementary internal actions. External actions, such as pressing a key, usually have a longer latency determined by the ACT-R/PM perceptual-motor modules [20].

ACT-R was developed at Carnegie Mellon University under sponsorship from the Office of Naval Research. ACT-R is implemented in Common Lisp and runs on MacOS, Windows and Unix platforms. A number of user tools are available, including a graphical environment to author and run ACT-R models, an adaptive web tutorial for learning how to model in ACT-R, a parameter optimizer that automates the task of model fitting and a multi-model extension that enables multiple ACT-R models to run concurrently and communicate with each other and with an interactive simulation. ACT-R is open-source and all software, models and tools are freely available on the web at the ACT-R web site http://act.psy.cmu.edu.

Model

If it is to justify its structural costs, a cognitive architecture should facilitate the development of a model in several ways. It should limit the space of possible models to those that can be expressed concisely in its language and work well with its built-in mechanisms. It should provide for significant transfer from models of similar tasks, either directly in the form of code or more generally in the form of design patterns and techniques. Finally, it should provide learning mechanisms that allow the modeler to only specify in the model the structure of the task and let the architecture learn the details of the task in the same way that human cognition constantly adapts to the structure of its environment. These architectural advantages not only reduce the amount of knowledge engineering required and the number of trial-and-error development cycles, providing significant savings in time and labor, but also improve the predictiveness of the final model. If the "natural" model (derived *a priori* from the structure of the task, the constraints of the architecture and the guidelines from previous models of related tasks) provides a good fit to the empirical data, one can be more confident that it will generalize to unforeseen scenarios and circumstances than if it is the result of *post hoc* knowledge engineering and data analysis. That is the approach that we have adopted in developing a model of this task, and indeed more generally our design and use of the ACT-R architecture.

Thus we did not try to reverse-engineer the subjects' strategies but instead tried to develop the simplest and most natural model for the architecture. We organized the model around a few goal types with their associated productions. Goal types correspond closely to the unit tasks in HCI [21] as well as to the tasks in task network models (e.g. [22]). Five goal types called **color-goal**, **text-goal**, **scan-text**, **scan-screen** and **process** were defined, together with a total of 36 very simple productions. Goals were simple and would hold just a

few elements, such as the aircraft currently being handled together with related information such as its position and the action to be performed.

Two basic modes of human interaction with the simulation were defined: one in which the operator had to rely mostly on text messages scrolling in windows to identify events that required action (the text condition), and one in which aircraft on the radar screen that required action would turn a color corresponding to the action (the color condition). The simulation also had three speeds (low, medium and high) that controlled how much time the subjects would have (10, 7.5 and 5 minutes respectively) to perform a given number of actions. For additional details on the simulation, see [3] and [5].

The goal type **color-goal** was the top goal for the color condition. Five productions were defined that applied to that goal. They scanned the radar screen continuously, identified an aircraft that had turned color, mapped the color into the required action by relying upon five simple memory chunks encoding the instructions that the subjects were given regarding the color-action mappings, then created a goal to perform the given action on the aircraft. The goal type **process** executed the sequence of mouse clicks required to perform the action. Twelve productions were defined to handle the five possible actions. This required clicking on a button identifying the action, then on the aircraft, then perhaps on a neighboring controller, then finally on the send button.

As expected, the text condition was both more difficult for the subjects and slightly more complicated for the model. The goal type **text-goal** was the top goal for the text condition. Four productions were defined to cycle through the three text windows and the radar screen looking for aircraft requiring action by creating goals of type **scan-text** and **scan-screen** respectively. A goal of type **scan-text** would handle the scanning of a single text window for a new message from another controller requesting action. A production was defined to systematically scan the window for such a message. If one was found, another production would attempt to retrieve a memory of handling such a request. Memories for such requests would be automatically created by the architecture when the corresponding goal was completed, but their availability was subject to their subsymbolic parameters, which were in turn subject to decay as well as reinforcement. If no memory could be retrieved, then the window would be scanned for another message indicating completion. If none could be found, then a **process** goal would be created to perform the action requested. Note that this is the same goal as in the color condition. A key component of the model was an additional production that would detect the onset of a new message in another window and interrupt the current goal to scan that window instead. This allowed the model to be sensitive to new events and handle them promptly. Scanning the radar screen was accomplished in a similar manner by goals of type **scan-screen** and their eight associated productions.

Finally, all the architectural parameters that control the performance of the simulation were left at their default values provided by previous models. Only two task-specific parameters were roughly estimated from the data: the production time to perform a mouse action was set at 1 second and the production time to perform a basic visual scan was set at 0.5 second.[1]

Finally, a key aspect of our methodology, which is also pervasive in ACT-R modeling [4], is the use of Monte Carlo simulations to reproduce not only the aggregate subject data (such as the mean performance or response time) but also the variation that is a fundamental part of human cognition. In that view, the model doesn't represent an ideal or even average subject but instead each model run is meant to be equivalent to a subject run, in all its variability and unpredictiveness. For that to happen, it is essential that the model not be merely a deterministic symbolic system but be able to exhibit meaningful non-determinism. To that end, randomness is incorporated in every part of ACT-R's subsymbolic level, including chunk activations, which control their probability and latency of retrieval, production utilities, which control their probability of selections, and production efforts, which control the time that they spent executing. Moreover, as has been found in other ACT-R models (e.g. [15], [23]), that randomness is amplified in the interaction of the model with a dynamic environment: even small differences in the timing of execution might mean missing a critical deadline, which results in an error condition, which requires immediate attention, which might cause another missed deadline and so on. To model the variation as well as the mean of subject performance, the model was always run as many times as there were subject runs. For that to be a practical strategy of model development, it is essential that the model run very fast, ideally significantly faster than real-time. Our model ran up to 5 times faster than real-time[2] on a desktop PC, making it possible to run a full batch of 48 scenarios in about an hour and a half, enabling a relatively quick cycle of model development.

[1] For software compatibility reasons, we did not use the perceptual-motor module of ACT-R/PM, which would have provided more precise estimates of those times.

[2] ACT-R models have run thousands of times faster than real-time. The limiting factor in this case was the simulation speed, especially the synchronization time between the air traffic control simulation and the model.

Results

As of this writing, the results from the HLA-enabled model have not yet been analyzed and thus we will report results from the first, non-HLA, phase of the comparison. The behaviorally significant components of the two model versions were identical, and based on a casual inspection of the HLA model runs there appears to be no significant difference between the two sets of results.

Because the variability in performance between runs, even of the same subject, is a fundamental characteristic of this task, we ran as many model runs as there were subject runs (48 total runs on 12 different scenarios). Figure 2 compares the mean performance in terms of penalty points for subjects and model for color (left three bars) and text (right three bars) condition by increasing workload level, i.e. simulation speed.

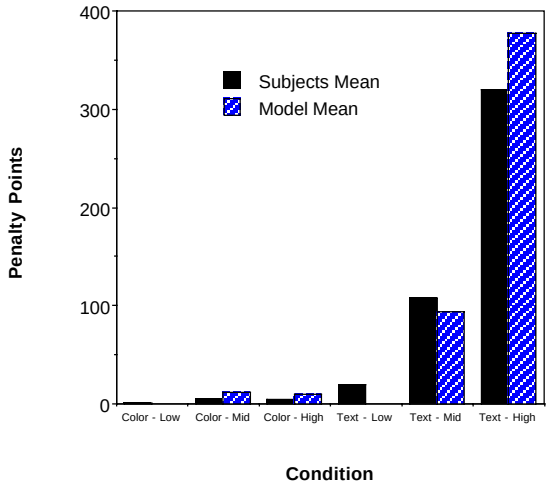

Figure 2: Mean performance for subjects vs. model

The model matches the data quite well, including the strong effects of color-vs-text condition and of workload for the text condition. Because ACT-R includes stochasticity in chunk retrieval, production selection and perceptual/motor actions, and because that stochasticity is amplified by the interaction with a highly dynamic simulation, it can reproduce a large part of the variability in human performance, as indicated by Figure 3 which plots the individual subject and model runs for the two conditions that generated a significant percentage of errors (text condition in medium and high workload). The range of performance in the medium workload condition is almost perfectly reproduced other than for two outliers and a significant portion of the range in the high condition is also reproduced, albeit shifted slightly upward. It should be noted that each model run is the result of an identical model that only differs from another in its runtime stochasticity. The model neither learns from trial to trial nor is modified to take into account individual differences.

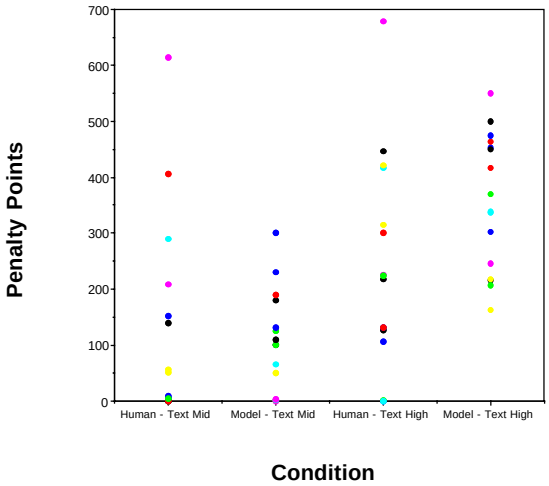

Figure 3: Performance for each subject vs. model run

The model reproduces not only the subject performance in terms of total penalty points, but also matches well to the detailed subject profile in terms of penalties accumulated under eight different error categories, as shown in Figure 4 for the same conditions:

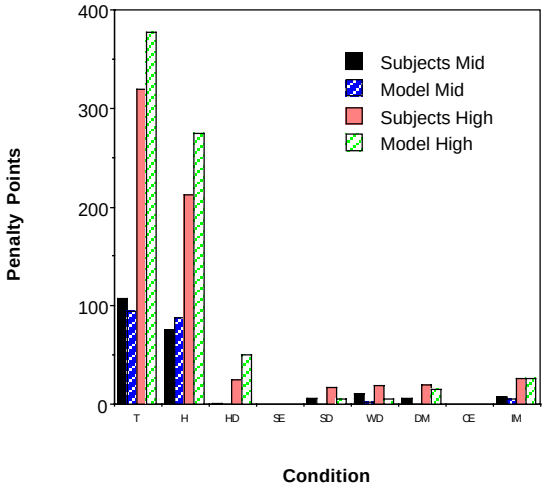

Figure 4: Penalty points for subjects vs. model runs

The model also fits the detailed pattern of latencies to perform a required action in terms of condition and number of intervening events. In a crucial test of the model's multi-tasking abilities, it also closely reproduces the pattern of response to a required action in terms of

number of intervening events before the action can be performed, a very sensitive measure of the ability to detect and process events immediately after they occur. That multi-tasking capacity results from the model's ability to detect event onsets and set the next goal to process those events. Thus, despite ACT-R's strong goal-directed behavior, it can exhibit the proper level of multi-tasking abilities without requiring any alteration to its basic control structure.

Finally, the model reproduces the subjects' answers to the self-reporting workload test administered after each trial. Since ACT-R does not have any built-in concept of workload, we simply defined the workload of an ACT-R model as the scaled ratio between the time spent in critical unit tasks to the total time on task. The critical unit tasks in which the model feels "pressured" or "busy" are defined as the **Process** goals, in which the model is busy performing a stream of actions, and those **Scan-Text** goals that are the result of an onset detection, in which the model feels "pressured" to find and process a new event requiring action. As shown in Figure 5, that simple definition captures the main workload effects, more specifically the effects of display condition and of schedule speed. Another quantitative effect that is reproduced is the higher rate of impact of schedule speed in the text condition (and the related fact that workload in the slowest text condition is higher than workload in the fastest color condition).

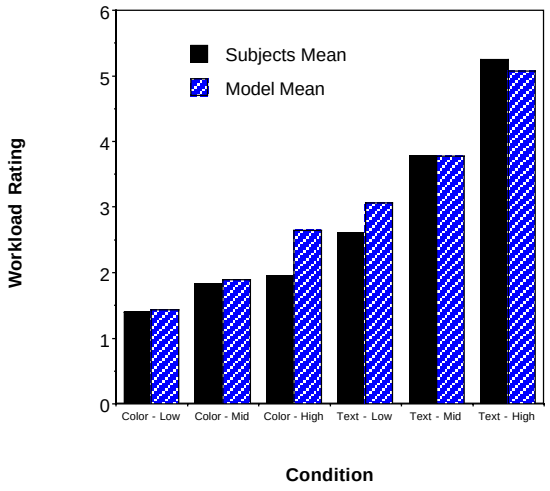

Figure 6: Mean workload for subjects vs. model

In summary, the advantages of this model are that it is relatively simple, requires almost no parameter tuning or knowledge engineering, provides a close fit to both the mean and variance of a wide range of subject performance measures as well as workload estimates, and suggests a straightforward account of multi-tasking behavior within the existing constraints of the ACT-R architecture.

HLA Implementation

Porting the ACT-R model to the High Level Architecture turned out to be relatively straightforward. The model itself did not require any changes. Only the code layer managing the interface between the simulation and the model, i.e. the code that makes the information about the environment (position of aircraft, text messages, etc) available to the model and relays the model's actions (mouse clicks) to the simulation, needed to be updated from calls to D-OMAR to equivalent calls to HLA.[3] Since D-OMAR and HLA contained similar time management schemes, few changes to the logic of the interface code were required. While the results of the HLA-enabled simulation runs have not yet been fully analyzed, they seem comparable to those of the original D-OMAR simulation.[4] Our relatively smooth experience with porting our existing cognitive model to HLA bodes well for its future as a standard for simulations involving human performance models.

There is however one note of caution regarding the speed of the simulation. The HLA-enabled simulation runs at about half the speed of (i.e. twice as slow as) real time. This represents a slowdown of about an order of magnitude over the D-OMAR simulation. It should be emphasized that HLA itself should not be blamed for all or even most of that difference. For example, since both ACT-R and D-OMAR are implemented in Lisp, the ACT-R model ran within the D-OMAR application, a huge advantage in terms of communication speed that is not available to modular simulation architectures such as HLA. Moreover, we might have failed to find the most efficient synchronization mode possible. For example, switching from the use of the Next Event Request command to the Time Advance Request command would allow greater parallelism between federates, as would the use of a larger lookahead value.[5] While the level of abstraction varies with each model, and with it the level of efficiency that might be achievable, it seems that all models could benefit from a less conservative time management strategy.

[3] Thanks to BBNT for providing hookups to HLA through the HLA-enabled D-OMAR simulation.
[4] See [3] and [6] for details of the Icarus federation.
[5] Thanks to Dave Prochnow and Ron King from MITRE for suggesting those and other ways of improving the simulation efficiency.

Rather, our intent is to highlight the need for maximum simulation efficiency when developing cognitive models. Because human performance varies unpredictably from trial to trial, many models (including the ACT-R model in the Icarus federation) are stochastic and their performance can only be assessed by repeated runs through a whole set of scenarios. Repeated model runs are also essential when one wants to reproduce seldom seen errors in the operation of a complex system, for example. However desirable, that modeling methodology makes strong demands on simulation efficiency. For example, in this simulation there were twelve different scenarios that were each run four times. In real time, this would take about eight hours, which would render model development very cumbersome. A simulation speed of five times faster than real time, which was attained in the previous implementation, reduced the time taken by a full model run to one to two hours, which is much more practical. An even higher efficiency would further reduce the model development cycle and make cognitive modeling an even more usable tool.

Conclusion

We presented a model of human performance in the control of a simplified air traffic control task. The model was developed using the ACT-R cognitive architecture, which accelerated model development, kept the model simple and improved its validation. The model accounted for both the average and distribution of a wide range of performance measures, including amount and type of errors, response latency choice percentages. The model provided an account of multi-tasking behavior and cognitive workload grounded in the architecture. Porting the model to the High Level Architecture proved relatively straightforward and transparent. However, the necessity for multiple simulation runs to account for the variability of human behavior puts strong demands on simulation efficiency.

References

[1] Pew, R. W., & Gluck, K. A. (2001). Overview of the Agent-based Modeling and Behavior Representation (AMBR) Model Comparison Project. In *Proceedings of the 10th Annual CGF-BR Conference*.

[2] Pew, R. W., & Mavor, A. S. (1998). *Modeling Human and Organizational Behavior: Application to Military Simulations*. Washington, D. C.: National Academy Press.

[3] Deutsch, S., & Benyo, B. (2001). The D-OMAR Simulation Environment, including Non-HLA and HLA Federate Architectures. In *Proceedings of the 10th Annual CGF-BR Conference*.

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

[5] Tenney, Y. J., & Spector, S. L. (2001). Comparisons of HBR Models with Human-in-the-loop Performance in a Simplified Air Traffic Control Simulation with and without HLA Protocols: Task Simulation, Human Data and Results. In *Proceedings of the 10th Annual CGF-BR Conference*.

[6] Feinerman, L. E., Prochnow, D. L., & King, R. A. (2001). Icarus, an HLA Federation of HBR Models. In *Proceedings of the 10th Annual CGF-BR Conference*.

[7] Anderson, J. R., Bothell, D., Lebiere, C. & Matessa, M. (1998). An integrated theory of list memory. *Journal of Memory and Language, 38,* 341-380.

[8] Anderson, J. R., Matessa, M., & Lebiere, C. (1997). ACT-R: A theory of higher level cognition and its relation to visual attention. *Human Computer Interaction, 12,* 439-462.

[9] Salvucci, D. D., & Anderson, J. R. (in press). Integrating analogical mapping and general problem solving: The path-mapping theory. *Cognitive Science*.

[10] Schunn, C. D., & Anderson, J. R. (in press). The generality/specificity of expertise in scientific reasoning. *Cognitive Science*.

[11] Lovett, M. C., Reder, L. M., & Lebiere, C. (1999). Modeling working memory in a unified architecture: An ACT-R perspective. In Miyake, A. & Shah, P. (Eds.) *Models of Working Memory: Mechanisms of Active Maintenance and Executive Control.* New York: Cambridge University Press.

[12] Anderson, J.R. & Schooler, L.J. (1991). Reflections of the environment in memory. *Psychological Science, 2,* 396-408.

[13] Rubin, D.C. & Wenzel, A.E. (1990). One hundred years of forgetting: A quantitative description of retention. *Psychological Review, 103,* 734-760.

[14] Newell, A. & Rosenbloom, P.S. (1981). Mechanisms of skill acquisition and the power law of practice. In J.R. Anderson (Ed.).*Cognitive skills and their acquisition* (pp. 1-56). Hillsdale, LEA.

[15] Lebiere, C., & West, R. L. (1999). A dynamic ACT-R model of simple games. In *Proceedings of the Twenty-first Conference of the Cognitive Science Society*, pp. 296-301. Mahwah, NJ: Erlbaum.

[16] Lebiere, C. (1998). The dynamics of cognition: An ACT-R model of cognitive arithmetic. Ph.D. Dissertation. *CMU Computer Science Dept Technical Report* CMU-CS-98-186. Pittsburgh,PA. Available at http://reports-archive.adm.cs.cmu.edu/.

[17] Sanner, S., Anderson, J. R., Lebiere, C., & Lovett, M. C. (2000). Achieving efficient and cognitively plausible learning in Backgammon. Proceedings of *The Seventeenth International Conference on Machine Learning*. San Francisco: Morgan Kaufmann.

[18] Lebiere, C., Anderson, J. R., & Reder, L. M. (1994). Error modeling in the ACT-R production system. In *Proceedings of the Sixteenth Annual Meeting of the Cognitive Science Society*, pp. 555-559. Hillsdale, NJ: Erlbaum.
[19] Anderson, J. R., Reder, L. M., & Lebiere, C. (1996). Working memory: Activation limitations on retrieval. *Cognitive Psychology, 30*, 221-256.
[20] Byrne, M.D. & Anderson, J.R. (1998). Perception and action. In J.R. Anderson & C. Lebiere (Eds.). *The atomic components of thought* (pp. 167-200). Mahwah: LEA.
[21] Card, S. K., Moran, T. P., & Newell, A. (1983). *The Psychology of Human Computer Interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates.
[22] Allender, L., Kelley, T. D., Salvi, L., Lockett, J., Headley, D. B., Promisel, D., Mitchell, D., Richer, C., & Feng, T. (1995). Verification, validation, and accreditation of a soldier-system modeling tool. In *Proceedings of the Human Factors and Ergonomics Society 29th Annual Meeting-1995* (pp. 1219-1223). San Diego.
[23] Lerch, F. J., Gonzalez, C., & Lebiere, C. (1999). Learning under high cognitive workload. In *Proceedings of the Twenty-first Conference of the Cognitive Science Society*, pp. 302-307. Mahwah, NJ: Erlbaum.

Author Biographies

CHRISTIAN LEBIERE is a Research Scientist in the Human-Computer Interaction Institute at Carnegie Mellon University. He received his B.S. in Computer Science from the University of Liege (Belgium) and his M.S. and Ph.D. from the School of Computer Science at Carnegie Mellon University. During his graduate career, he worked on the development of connectionist models, including the Cascade-Correlation neural network learning algorithm that has been used in hundreds of scientific, technical and commercial applications. Since 1990, he has worked on the development of the ACT-R hybrid cognitive architecture and is co-author with John R. Anderson of the 1998 book _The_Atomic_Components_of_Thought_. His main research interest is cognitive architectures and their applications to psychology, artificial intelligence, human-computer interaction, decision-making, game theory, and computer-generated forces.

JOHN R. ANDERSON received his B.A. from the University of British Columbia in 1968 and his Ph.D. from Stanford University in 1972. He has been at Carnegie Mellon University since 1978 where he is a professor of psychology and computer science. He is a member of the National Academy of Science. He has published a number of books including Human Associative Memory (1973 with Gordon Bower), Language, Memory, and Thought (1976), The Architecture of Cognition (1983), The Adaptive Character of Thought (1990), Rules of the Mind (1993) and The Atomic Components of Thought (1998 with Christian Lebiere). His current research is focused on three enterprises involved in testing various aspects of the ACT-R theory of cognitive architecture. One is to model the acquisition of cognitive skills, particularly those involving dynamic problem solving. The second is the application of the architectures to developing intelligent tutoring systems and cognitive agents for training. The third is research on brain imaging to identify the neural correlates of the cognitive architecture.

DAN BOTHELL is a Senior Research Programmer in the Psychology Department at Carnegie Mellon University. He received his B.S. in Math/Computer Science with a minor in Psychology from Carnegie Mellon University in 1996. Since then, he has worked for John R. Anderson in support of the ACT-R hybrid cognitive architecture.

Multi-tasking and cognitive workload in an ACT-R model of a simplified air traffic control task. In Proceedings of the 10t h Conference on Computer Generated Forces and Behavior Representation. Norfolk, Va.Google Scholar. Montemerlo, M. (2000).Â Cite this paper as: Morris A.C., Smart C.K., Thayer S.M. (2002) Adaptive Multi-Robot, Multi-Operator Work Systems. In: Schultz A.C., Parker L.E. (eds) Multi-Robot Systems: From Swarms to Intelligent Automata. Springer, Dordrecht.