A THEORY-BASED MODEL OF COGNITIVE WORKLOAD AND ITS APPLICATIONS

Christian Lebiere, Human-Computer Interaction Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania

ABSTRACT

We present a model of cognitive workload based on the ACT-R (Adaptive Character of Thought – Rational) cognitive architecture. That model has been validated in a synthetic air traffic control task according to a wide range of behavioral measures. Its cognitive workload predictions are sensitive to level of task embedding, interaction speed, level of interface decision support and individual differences. We sketch possible extensions of the model that support multiple workload dimensions and instantaneous workload ratings. We also discuss possible applications of this kind of fine-grained computational models to system design, operator training and selection and dynamic load balancing.

ABOUT THE AUTHOR

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.

A THEORY-BASED MODEL OF COGNITIVE WORKLOAD AND ITS APPLICATIONS

Christian Lebiere, Human-Computer Interaction Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania

INTRODUCTION

Cognitive workload is a primary indicator of human performance in many tasks, in particular those involving human control of a dynamic interactive environment. Precisely predicting which situations will lead to a high level of cognitive workload and dynamically modifying or restructuring the task or environment to avoid those situations could lower operator stress, reduce the likelihood of errors, enhance the efficiency of teamwork and generally improve performance.

We present a cognitive model of a human controller in a synthetic air traffic control task. The model is implemented in the ACT-R (Adaptive Character of Thought – Rational) cognitive architecture (Anderson & Lebiere, 1998). ACT-R is a hybrid architecture of cognition that combines a symbolic production system with a subsymbolic activation calculus that determines the performance of the symbolic level. The model makes precise predictions for cognitive activity at a subsecond scale and is validated against a broad measure of behavioral measures, including amount and type of errors, response latency, decision making and attentional focus.

We extended the theory to have the model predict the overall level of cognitive workload, as reported by the subjects in a post-experiment test. The model's workload prediction is based on the percentage of time spent in cognitively demanding unit tasks, and is also sensitive to the level of task embedding. The model correctly predicts not only the average workload level, but also its sensitivity to the speed of the scenario-driven interaction and the level of decision support embedded in the simulation interface. It also predicts the impact of individual differences in working memory capacity on performance and cognitive workload.

The model can be readily extended to account in a principled fashion for the impact on cognitive workload and performance of knowledge and strategies that can be vary with individual differences and level of training. The workload definition can also be straightforwardly extended to account for moment-to-moment workload and for different categories of workload measures.

Finally, we discuss potential applications of our model based on its capacity to predict cognitive workload as a function of a range of environmental conditions and system designs. Because the model interacts with the system in the same way as humans, it can be used to inform issues of system design. Because ACT-R captures the effect of individual differences on performance, it can provide a principled measure for operator selection. It can also be used to assist operator training by providing a source of feedback to the trainee. Finally the model can provide the input to warning systems and load-balancing algorithms for the operation of complex multi-operator systems such as air-traffic control.

ACT-R

ACT-R (Anderson & Lebiere, 1998) is a hybrid production system theory that models 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. 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.

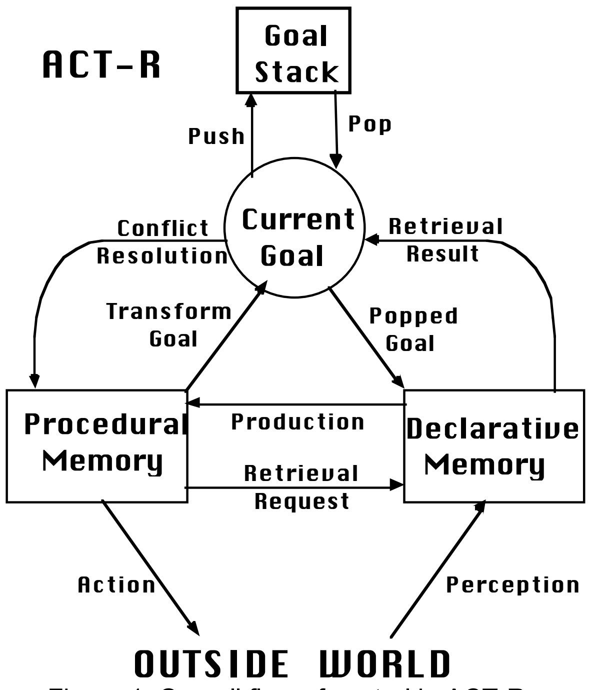


Figure 1: Overall flow of control in ACT-R.

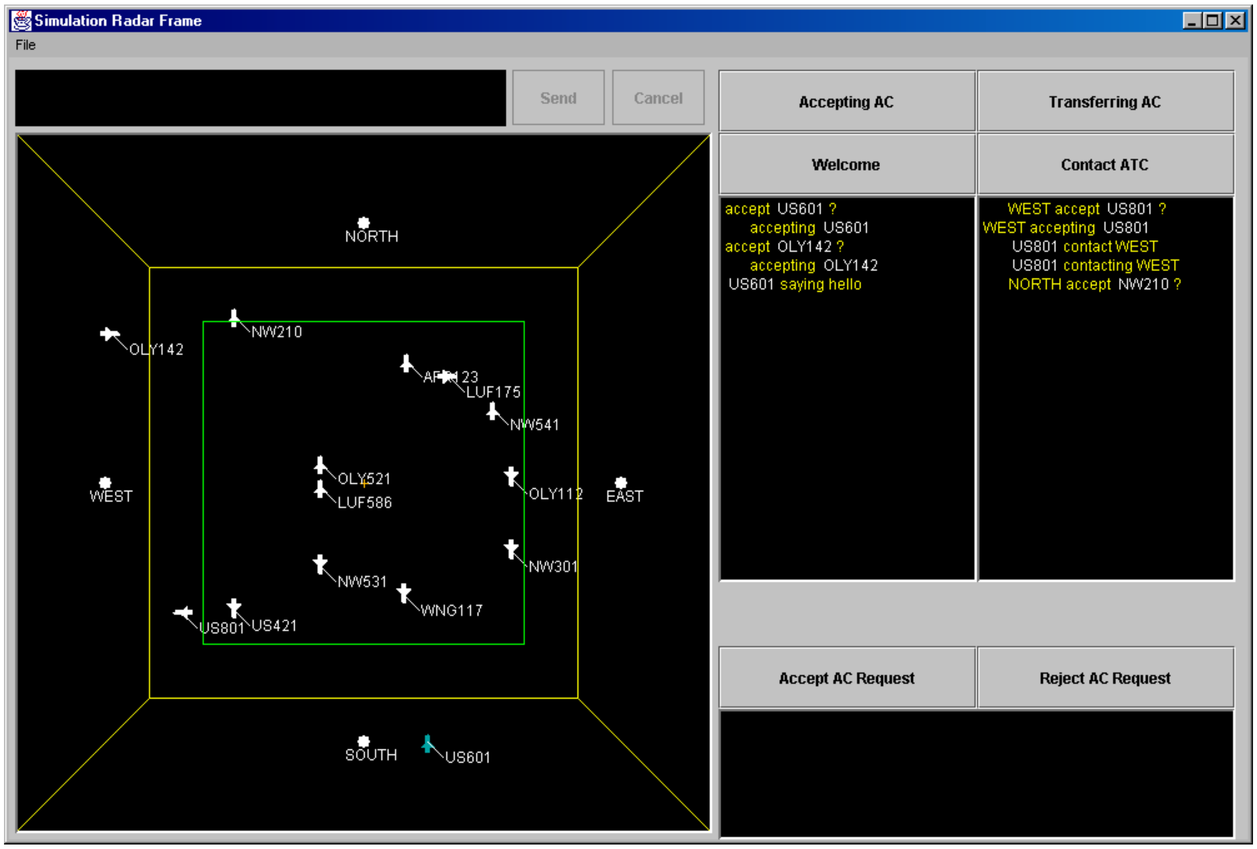
Figure 1 displays the information flow in the ACT-R system. There are essentially three memories -- a goal stack that encodes the hierarchy of intentions guiding behavior, a procedural memory containing production rules, and a declarative memory containing chunks. Productions are condition-action pairs 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 (Lovett, Reder & Lebiere, 1999). 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 (1991) have shown, this learning produces such basic features of human cognition as the Power Law of Forgetting (Rubin & Wenzel, 1990) and the Power Law of Practice (Newell & Rosenbloom, 1981).

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. Lebiere & West, 1999), as well as a similarity-based matching component, which provides generalization and robustness (e.g. Lebiere, 1998; Sanner, Anderson, Lebiere & Lovett, 2000). 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 (Anderson, Reder & Lebiere, 1996). 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 (Byrne & Anderson, 1998).

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.



accept US601 ?
accepting US601
accept OLY142 ?
accepting OLY142
US601 saying hello

WEST accept US801 ?
WEST accepting US801
US801 contact WEST
US801 contacting WEST
NORTH accept NW210 ?

Figure 2: Air Traffic Control simulation in D-OMAR

Task

The Agent-based Modeling of Behavior Representation (AMBR) comparison is structured as a sequence of one-year tasks each designed to emphasize a specific aspect of human behavior. The behavior targeted in the first phase was multi-tasking, an activity directly related to cognitive workload (Wickens, 1992). Pew and Mavor (1998) 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 (MacMillan, Deutsch & Young, 1997). This domain requires a controller to manage one sector of air space, especially the transition of aircraft into and out of the sector. Scenarios can vary the number, speed, altitude and type of aircraft requesting access to the sector and can be complicated by having them arrive from multiple directions and adjoining sectors. This is a rich enough infrastructure to create a variety of scenarios having variable task load levels and varying levels of planning complexity.

A simulation of the task was developed using the Distributed Operator Model ARchitecture D-OMAR (Deutsch, MacMillan, Cramer & Chopra, 1997) to facilitate setting up, running and collecting data from both human performance and computer models interchangeably. Figure 2 displays a screen shot of the simulation. The main part of the screen on the left contains a graphical representation of the entire airspace, with the part controlled by the human or model agent contained in the central yellow square. The rest of the airspace is divided by the yellow lines in four regions, named North, East, South and West, each managed by a separate controller. At any point during the simulation a number of airplanes (the exact number being a parameter controlling the difficulty of the task) are present in the airspace, flying through the central region or entering or exiting it. The task of the central controller is to exchange messages with the airplanes (each tagged with its identifying code, e.g. UAL344) and neighboring controllers to manage their traversal of its airspace. Those messages are displayed in the text windows on the right of the screen, with each window dedicated to a specific message category. The top left window concerns messages sent when a plane is entering the central controller's region while the top right window concerns messages sent when a place is exiting the central region. Both windows include messages exchanged between controllers as well as messages between the central controller and the plane itself. The bottom window concerns messages from and to planes requesting a speed increase, which should be granted unless that plane is overtaking another plane, which is the only airspace conflict that this simplified task allows.

A single event involves a number of messages being exchanged, all of which are appended to the relevant text window. For example, in the case of a plane about to enter the central region, a message requesting permission to enter will first be sent to the central controller from the controller of the neighboring region from which the plane originates. The central controller must reply to the other controller in a timely manner to accept the plane, then contact the plane to welcome it to the airspace. Those two cannot be performed in immediate succession, but instead require waiting for the first party contacted (in this case the other controller) to reply before taking the final action. This delay allows for the interleaving of unit tasks but also requires the maintenance of the currently incomplete tasks in working memory. Messages from other tasks can arrive when a task is being processed, thus requiring some search of the text window to identify the messages relevant to a task. A message is composed by clicking a button above the relevant text window (e.g. Accepting AC), then clicking in the graphical window on the intended recipient (e.g. another controller) and optionally the target of the message (i.e. a plane, unless it is the intended recipient in which case this is omitted), then the send button above the graphic window. The message being composed is displayed at the top left of the display in a text window.

To objectively measure performance on the task, penalties were assessed for a variety of failures to act in a timely manner. To develop on the experience accrued in MacMillan et al. (1997), a decision support condition contrasted with a support condition were implemented to dissociate two aspects of multi-tasking behavior. In the standard condition, subjects had to parse the messages printed in the text windows on the right side of the screen to determine which planes needed attention and which functions needed to be performed on them. In the assisted condition, planes that require assistance were color-coded in the graphical display on the left side of the screen according to the task needed to be performed (green for accept, blue for welcome, orange for transfer, yellow for contact, magenta for speed change and red for holding). This helped the subjects track visually which tasks need to be attended to and removed any necessity to parse the text windows on the left, a complex and time-consuming task. Therefore it dissociated the maintenance and updating of the queue of to-be-attended tasks from the resolution of conflicts between high-priority tasks.

Two sets of scenarios were created: one set was provided to the developers on which to base their designs, and another set was reserved to be used at the time of the competitive validation, a.k.a. the fly-off. Human performance data on the first set of scenarios were provided to the developers to fine-tune their model. The data from the second set of scenarios were withheld until after the fly-off for comparison with the model performance. The range of behavior requirements of both sets had the same scope, but the way in which those behaviors were exercised were not identical in order to test the robustness of the models.

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 Human-Computer Interaction (Card, Moran & Newell, 1983) as well as to the tasks in task network models (e.g. Allender et al, 1995). 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.

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, 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. Lebiere & West, 1999, Lerch, Gonzalez & Lebiere, 1999), 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.

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 3 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.

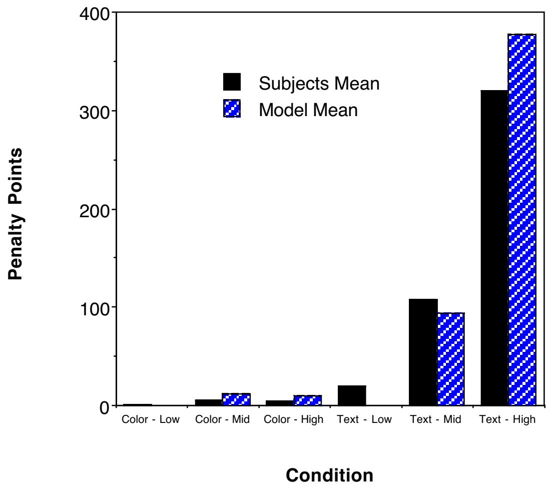


Figure 3: Mean performance for subjects vs. model

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 4 which plots the individual subject and model runs for the two conditions that generated a significant percentage

[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.

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.

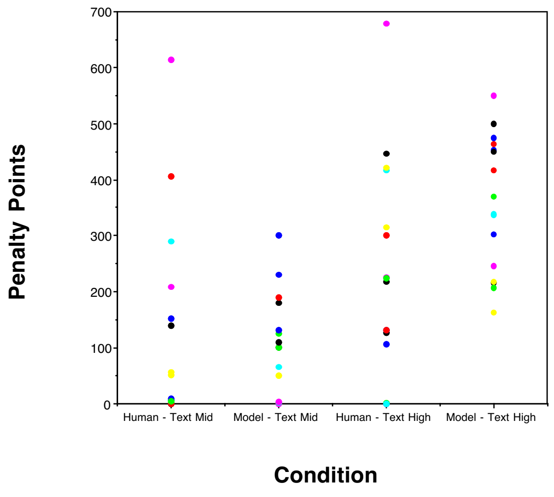


Figure 4: Performance for each subject vs. model run

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. 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 cognitive 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). This results because some **Process** goals are subgoals of **Scan-Text** goals, thus the time of the inner goals count twice, reflecting the cost of multi-tasking.

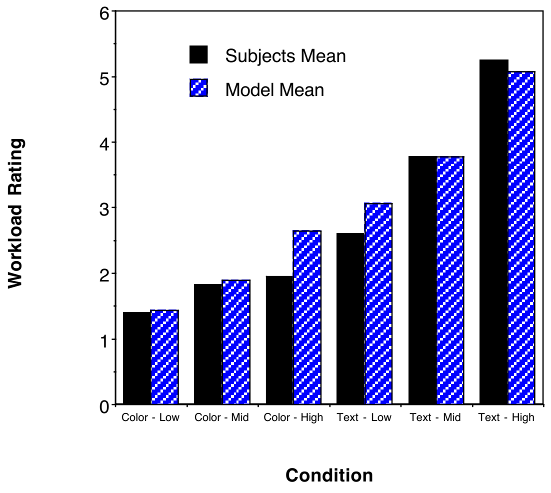


Figure 5: Mean workload for subjects vs. model

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.

WORKLOAD

The National Air and Space Administration Task Load indeX (NASA TLX) scale (Hart & Steveland, 1988) used by the subjects assesses workload according to six seven-point scales corresponding to the dimensions of mental demand, physical demand, temporal demand, performance, effort and frustration level. Because the subjects reports were highly correlated along these dimensions, we did not attempt to provide an estimate of workload for the ACT-R model along each dimension. It would seem possible to estimate workload along at least some of those dimensions. Mental workload would correspond to the more demanding cognitive operations, such as declarative memory retrieval, decision-making, etc. Physical workload would map more closely to the operations of the motor modules, e.g. how much time were these modules busy performing actions. Temporal demand, as a measure of the pace or time pressure, could be estimated from the amount of multi-tasking the model had to do, itself reflected in the amount of goal switch resulting from interruptions. Performance is an estimate of how well one accomplished the task. This is likely to be highly sensitive to the nature of the task and to the specific success criterion set for it, e.g. in this case the number of penalty points. Effort and frustration level also seem to be fairly subjective measures that are likely to vary from one individual to another according to poorly understood conditions. Any variation beyond the obvious correlation with the other dimensions (e.g. effort is likely to be positively correlated with the various demand dimensions and frustration is likely to be negatively correlated with performance) might be hard to capture in a purely computational model.

Our definition of cognitive workload is by no means limited to the unit task level. While that was a convenient level in which to ground the measure for our current purposes, the definition can be refined to the level of individual cognitive cycles and the use that they make of the various resources at a millisecond grain-scale. As such, it could be considered a theoretically grounded measure that could serve as the underlying basis for task-level formulas (e.g. Wickens, 1992, Allender et al, 1995). While the workload output by the model are average for the entire run, it could easily be generalized to provide a time-sensitive measure. A given, presumably short, time interval could be defined over which to apply the same formula to give the workload at the end of that interval. Perhaps more generally, the instantaneous workload at all points in the past could be decayed according to the same power law function used to compute the decay in ACT-R base-level activation. This would allow the entire past history to be taken into account, but would give precedence to the more recent events. In any case, better data on moment-to-moment variations in workload assessment would be needed to test this extension.

Wickens (1992) describe practical applications of research in cognitive workload. They involve predicting multiple-task performance for purposes of system design, operator training and operator selection. Our method of performance prediction differs from the methods described by Wickens (1992) in a fundamental way. Whereas they use workload measures to estimate performance in multiple-tasks settings, the performance generated by the ACT-R model originates directly from the cognitive constraints of the model and the demands of the task and is not a function of the model workload.[3] For system design, the ACT-R model can account for the effect of different design decisions, including interfaces, because ACT-R interacts with the system through the same interface as subjects thanks to its perceptual/motor modules (e.g. Salvucci, 2001). Assistance in operator training can also be provided through the ACT-R model by a technique called model-tracing used in cognitive tutors (Anderson, Corbett, Koedinger & Pelletier, 1995). Finally, operator selection can also be performed using the model through ACT-R's account of individual differences. Lovett, Reder & Lebiere (1999) showed that variation in a single parameter controlling activation spreading, called W, can account for individual differences in performance in a wide range of tasks. Reliable estimation of a subject's W on one or more tasks can then be used to predict the subject's performance on other, potentially more complex tasks.

Finally, fine-grained computational models such as this ACT-R model makes possible a new class of applications that dynamically use the models to determine future points of potential problems. For example, in settings such as air traffic control in which a composite task load is shared among multiple human operators, a dynamic load-

[3] Though it could if some of the model's decisions depended on its estimate of workload. There is anecdotal evidence that subjects sometimes make strategic adjustments on the basis of workload.

balancer could be designed that determines the best way to assign or re-assign tasks to operators based on projections of specific cognitive bottlenecks for individual operators at some point in the future. Such model-based projections would also be useful in providing advanced warning systems for periods of workload transitions that occur when cognitive demands suddenly increase. (Huey & Wickens, 1993). In all of these applications, fine-grained computational models are essential to provide the proper precision and generality of application to varying circumstances.

REFERENCES

Adams, M. J., Tenney, Y. J., & Pew, R. W. (1991). *Strategic Workload and the Cognitive Management of Advanced Multi-task Systems.* Crew System Ergonomics Information Analysis Center. Wright-Patterson AFB, OH.

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.

Anderson, J. R., Corbett, A. T., Koedinger, K., & Pelletier, R. (1995). Cognitive Tutors: Lessons learned. *Journal of Learning Sciences*, 4, 167-207.

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

Anderson, J. R., Reder, L. M., & Lebiere, C. (1996). Working memory: Activation limitations on retrieval. *Cognitive Psychology, 30*, 221-256.

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

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.

Card, S. K., Moran, T. P., & Newell, A. (1983). *The Psychology of Human Computer Interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Deutsch, S. E., MacMillan, J., Cramer, N. L., & Chopra, S. (1997). *Operator Model Architecture (OMAR) Final Report*. BBN Report No. 8179. BBN Corporation, Cambridge, MA.

Hart, S. G., & Staveland, L. E. (1988). Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In P. A. Hancock & N. MeshKati (eds.), *Human Mental Workload*. Amsterdam: North Holland.

Huey, B. M., & Wickens, C. D. (1993). *Workload Transition: Implications for Individual and Team Performance*. Washington, DC: NAP.

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/.

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.

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.

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.

MacMillan, J., Deutsch, S. E., & Young, M. J. (1997). A comparison of alternatives for automated decision support in a multi-task environment. *Proceedings the 41st Annual Meeting of the Human Factors and Ergonomics Society.*

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.

Pew, R. W., & Mavor, A. S. (1998). Modeling Human and Organizational Behavior. Application to Military Simulations. Washington, DC: National Academy Press.

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

Salvucci, D. D. (2001). Predicting the effects of in-car interfaces on driver behavior using a cognitive architecture. *CHI Letters*, CHI 2001 Conference Proceedings.

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.

Wickens, C. D. (1992). *Engineering Psychology and Human Performance*. New York, New York: Harper Collins.