# Cognition Models Bake-off: Lessons Learned from Creating Long-Running Cognitive Models

Siyu Wu [1], Amir Bagherzadeh [2] , Frank E. Ritter[1] and Farnaz Tehranchi [3]

1 College of Information Sciences and Technology
2 Department of Industrial and Manufacturing Engineering
3 School of Engineering Design and Innovation
Penn State, University Park, PA, 16802, USA

**Abstract.** We design and develop models capable of emulating human-like game playing behaviors through interaction with unaltered interface. Each model is designed to represent a specific type of human behavior observed during gameplay. These models were constructed using the ACT-R cognitive architecture and incorporate perceptual-motor knowledge of the virtual environment. Our implementation follows an "eyes and hands" approach, incorporating enhanced components for visual perception and robotic hand actions at the bitmap level. During the experimentation phase, one of the cognitive models ran for a prolonged period, spanning approximately 4 hours as it drove a simulated bus from Tucson to Las Vegas. In contrast, the other model demonstrated a shorter gameplay duration. We employed two different design approaches for these models, incorporating varied knowledge representations and actions. These approaches were either matched or unmatched with respect to the fine details of human behavior, such as the number of course corrections, average speed, and learning rate. Upon analyzing the performance of these models, it was discovered that the longer-running model did not exhibit the same level of fatigue as is typically observed in human subjects. Additionally, the shorter-running model and its behavior in this task revealed significant limitations in both the ACT-R framework and the extended hands and eyes framework. These limitations specifically pertained to the efficiency of visual pattern matching in a dynamic environment and the correspondence time of ACT-R and its extended eyes and hands components. We have documented these limitations and have already begun implementing them for further investigation.

**Keywords:** Cognition Computational Modeling; ACT-R; Driver Model

## 1 Introduction

Cognitive architectures, like ACT-R, serve as frameworks for creating cognitive models of various psychological phenomena and tasks. These architectures integrate procedures and structures that imitate human behavior, encompassing reaction times, error rates, and fMRI findings[1].

ACT-R is a cognitive architecture implemented as software, enabling the construction of models that can store, retrieve, process knowledge, and predict performance [2], [3]. Researchers and modelers have used ACT-R to create diverse models, ranging from cognition-focused tasks like Tower of Hanoi to comprehensive models incorporating perception and motor behaviors.

Different approaches exist for implementing perceptual and motor behaviors, classified based on their direct interaction with tasks [4]. Modified interface tasks, where models interact with altered interfaces, are commonly observed [5]. Another approach involves models interacting with unmodified tasks, using simulated eyes and hands visible to users [6]. Previous studies employing ACT-R have compared cognitive model behavior with human behavior, indicating closer alignment at the cognition level than at the perceptual-motor level [7]. However, future directions may involve extending ACT-R to model additional aspects of vision and motor behavior in uninstrumented interfaces [4], [8].

[7] developed an ACT-R model with an extended vision and motor management tool (JSegMan) for the Desert Bus video game, however deviated from human behavior with less than 20 minutes of operation. Building upon this and our previous work [9], which introduced a cognitive model with a different control strategy for prolonged performance, we developed models simulating distinct gameplay behaviors. Using diverse control mechanisms, we integrated varied visual patterns and extended functionalities of hands and eyes into different ACT-R models, aiming to replicate different types of human driving behaviors.

In this paper, we further develop our models and introduce two cognitive models demonstrating real-time control in a driving task. One model operates up to 4 hours, while the other runs for a shorter duration. These models interact with an unmodified interface, showcasing their adaptability within existing systems. To achieve this, we employ a human-centered design process to differentiate the models based on knowledge representation and their actions. The exhibited behavior of these models during the task highlights limitations within the ACT-R and extended hands and eyes framework. To present these limitations systematically, we compile them and establish a foundation for future implementation and improvement.

## 2 Components and Theoretical Foundation

We will outline the architecture and perceptual interface structure for interaction, followed by an overview of the engaged simulation and models.

### 2.1 The Architecture of Cognition

ACT-R, a cognitive architecture and theory, helps simulate human cognition, enabling the creation of models for knowledge storage, retrieval, and processing [3], [4]. It includes declarative knowledge, recognizing facts (e.g., an apple is a fruit), and procedural knowledge, involving operations and instruction execution (e.g., tasks in a driving scenario). Despite ACT-R's limitations, our research enhances its potential, incorporating new interaction knowledge and facilitating interaction with tasks via computer interfaces, displayed on a screen, and operated using a keyboard and a mouse.

### 2.2 The Architecture of Interaction

Models interact with the environment via visual and motor systems, perceiving items and performing actions like key pressing. The visual system stores object location in a "where" buffer, and object details in a "what" buffer. The central production system utilizes this information to reason and guide behavior. For instance, in the context of a driving model, the model may decide to move forward or steer based on the position data extracted from the visual buffer [2], [4].

To interact with the simulation, our approach involves directly examining the screen's bitmap to identify objects. The motor output is transmitted via the USB bus, emulating user actions such as typing characters on the keyboard or moving the mouse. VisiTor [8] is an open-source (GitHub)Python software that mimics hand and eye functionalities. It includes visual functions: "whatIsOnScreen" (detects specific visual patterns), "whereIs" (locates patterns), and "getMouseLocation" (retrieves mouse position). Motor functions encompass "click" (simulates a mouse click), "Keypress" (emulates keyboard press), "moveCursorTo" (replicates cursor movement to specified locations), and "moveCursorToPattern" (mimics cursor movement to a visual pattern).

### 2.3 The Simulation

Penn and Teller's Desert Bus is a deliberately monotonous and lengthy video game . It involves real-time driving from Tucson, AZ to Las Vegas, NV at a maximum speed of 45 mph, each eight-hour trip featuring a bus that continually drifts right. Going off-road results in stalling and restarts from Tucson. The game lacks virtual passengers or vehicles, allowing infinite replays after each journey completion, with the road appearing dark during in-game night. Available on platforms like Steam. Below is a screenshot of the game (see Fig. 1).

**Fig. 1.** A Screenshot of the driver’s view at approximately 10 min. into the game, oversteered.

In Desert Bus game, players enjoy a first-person view and a dynamic environment. We used the edition by Dinosaur Games and Gearbox Software, based on the unreleased "Smoke and Mirrors" Sega CD game. The environment, downloadable from Steam (https://store.steampowered.com/app/638110/Desert_Bus_VR/) for Windows, was unaltered for the model. Our study concentrates on the model's ability to drive from Tucson to Las Vegas.

## 3 Desert Bus Models

To enhance the performance of the bus-driving models, including gameplay duration and stability, we conducted a mini cognitive bake-off experiment. We observed two human subjects' gameplay and built models accordingly. This involved extending functionalities to emulate different human gameplay behaviors. We also explain how we enhanced ACT-R's perceptual-motor system, VisiTor, to support our model and delve into specific model components and mechanisms.

### 3.1 Extending ACT-R With VisiTor.

ACT-R 7, with a Perceptual-Motor module, interacts with Macintosh Common Lisp (MCL) interfaces, but not non-MCL ones like Desert Bus. We integrated VisiTor [8] into ACT-R 7, enabling the simulation of visual attention and user input, thereby providing access to other interfaces. VisiTor's capabilities can be further expanded by incorporating additional modules. ACT-R uses VisiTor to scan for specific pixel patterns, initiating program execution when found. An "if-then" rule then commands VisiTor to press and hold the "W" key to start and maintain bus speed. If VisiTor detects a deviation off the road center, a steering rule instructs it to hold "W" and "A" keys until the car re-centers, emphasizing real-time responses to the visual environment.

To fulfill this task, VisiTor required minor extensions, such as simplifying the description of visual objects and incorporating a range of visual objects. Additionally, it needed to facilitate the transfer of motor commands with variable durations to maintain key presses. To support the continuous driving of the bus, we implemented the "longpresskey" feature in VisiTor, enabling simulation of prolonged key-press actions by specifying a duration of time to hold the key.

### 3.2 Driver Models.

We identified different types of human gameplay behavior in this game and adopted a human-centered approach to develop corresponding models. Two authors played the game for 8 hours. They recorded their gameplay using the Recording User Input keystroke and mouse move logger [10]. Additionally, they provided field notes describing their gameplay strategies. Based on their observed gameplay behavior, we conducted a cognitive bake-off experiment to emulate their behaviors and create two cognitive models capable of real-time control in this simulation [11]. We will explain each model individually, in a sequence of control loop mechanisms, knowledge representations, production rules and the corresponding functions utilized in the eyes and hands approach.

**Model A.** Model A mimics the gameplay behavior of Type A. The human subject identified a strategy to optimize gameplay by leveraging the inherent game design. The subject observed that the game environment, characterized by its monotonous nature, features a stationary bus with the road moving within it. Notably, a visual cue indicating danger appears when the right white border of the road overlaps with the right visual

border of the bus. Whenever the subject sees this visual cue, the subject steers the bus back to the center of the road.

*Control loop*. Below we illustrate the mechanism for controlling model A (see Fig. 2(a)). Model A starts the bus by continuously pressing "W" upon seeing the yellow center lane. It monitors the environment for a visual danger cue, responding with a leftward steer ("A" key press) if detected. Without the cue, the model proceeds forward.

*Knowledge Representation*. The model has one type of chunk named “drive” and has two slots, “strategy” and “state”, with the state having parameters as object items, and a total of 7 declarative memories, which are working memories that tell the model to make the action based on the visual cues it saw.

*Actions to perform*. This model uses an explicit goal state to control the model. It contains 8 production rules. Table 1 (columns 1 and 2) list the high-level descriptions of the steps the model performs and the corresponding production rules. It shows the model initially scans the simulation environment to locate and acquire the required visual cue for gameplay initiation. It then uses the manual buffer to sustain the speed by maintaining key pressure. Concurrently, the model continuously assesses the environment for the presence of a “danger” visual cue (too far right). If such a cue is detected, the model maintains key pressure to steer the bus left. Otherwise, the model moves forward.

**Model B.** Model B replicates the gameplay behavior of Type B. The human subject has devised a strategy that emulates human driving behavior. The subject assesses the bus's position and measures the deviation from the center. When the deviation exceeds a certain threshold, the subject steers the bus back to the center of the road.

*Control loop*. Below we illustrate the mechanism for controlling model B (see Fig. 2(b)).

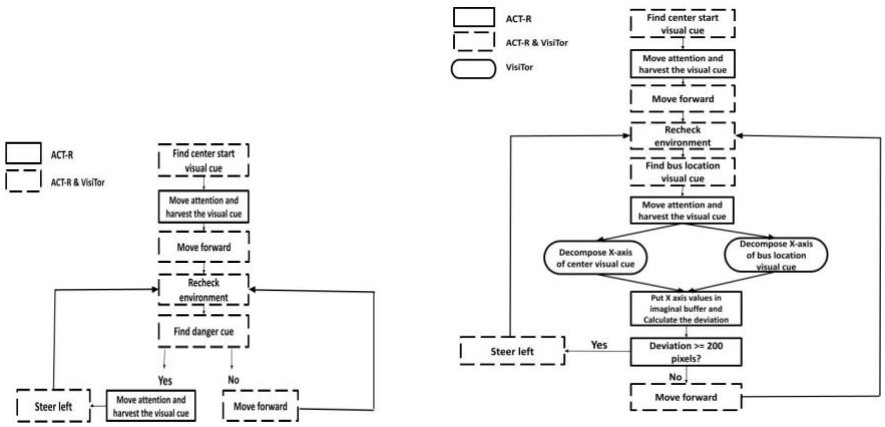

**Fig. 2.** The control loop of model A (a) and model B (b).

Figure 2 (b) shows the model's control loop. Using the visual buffer and simulated eyes, it focuses on two specific visual objects, encodes their screen-x locations with VisiTor's "whereis" function, calculates the deviation by subtracting these values, and determines if left steering is required when the deviation surpasses 200 pixels.

*Knowledge Representation.* The model consists of two types of chunks, with a total of 12 declarative memories. These memories serve as working memories that guide the model's actions based on the visual cues it has encountered. The first chunk, called "drive," has two slots named "strategy" and "state," with the state slot containing parameters related to object items. The other chunk type, "encoding," includes slots for the screen-x locations of the two visual cues and a deviation slot.

*Actions to perform.* This model employs an explicit goal state to regulate its behavior and consists of 13 production rules. The steps performed by the model and their corresponding production rules are summarized in Table 1 (columns 3 and 4). It shows that the model initiates by examining the simulation environment to locate and acquire the necessary visual cue for initiating gameplay. It uses the manual buffer to maintain forward motion by continuously pressing the key. Simultaneously, the model continually assesses the environment to identify and collect visual cues related to the bus's position. It then calculates the deviation between the x-axis of the center line and the bus's location, and if this deviation exceeds 200 pixels, the model holds down the key to steer the bus left. Otherwise, it continues driving forward.

**Table 1.** High-level description of the steps and the production rules used in models A and B. From left to right are columns 1, 2, 3, and 4.

| High level descriptions of steps for model A | Corresponding production rules | High level descriptions of steps for model B | Corresponding production rules |
|---|---|---|---|
| 1. When it detects a start visual cue, attend it, and press the “W” key using the manual buffer | Go<br>PerceiveEnvironment<br>Move-attention<br>Ahead | 1. When it detects a start visual cue, attend it, and press the “W” key using the manual buffer | Go<br>PerceiveEnvironment<br>Move-attention<br>Ahead |
| 2. Clear the visual buffer and shift model attention to recheck environment | Recheck-environment<br>Move-strategy | 2. Clear the visual buffer and attend to the bus location | Recheck-environment<br>Danger<br>Finding-danger<br>Move-at |
| 3. The danger cue visual pattern is detected, indicating the risk of driving off the lane, the manual buffer is cleared. | Finding-cue | 3. Calculate the bus deviation from the center lane | Where-is-danger<br>Where-is-center<br>Calculate-deviation |

| | | | |
|---|---|---|---|
| 4. Use the manual buffer by pressing “A” for 3 seconds | Steer | 4. Use the manual buffer by pressing “w” if the deviation is less than 200 pixels | Consider-ahead |
| 5. clear the visual buffer, Repeat steps 1,2,3,4 | Loop back to perceive-environment | 5. Clear the manual buffer if the deviation > 200 pixels. Using the manual buffer, align the bus by pressing the key for 6 seconds | Consider-steer |
| | | 6. After that, clear the visual buffer, Repeat steps 1,2,3,4,5 | Loop back to perceive-environment |

## 4 Demonstration Observations

The experiment simulated two models in the same environment to compare performances and gather ACT-R output. Key metrics included gameplay duration, hazard identification and steer, and response time. Compared to [7], both models showed longer operational capabilities, effectively meeting simulation requirements using declarative memories and production rules. An innovative feature was added to VisiTor to facilitate bus acceleration. However, each model exhibited unique characteristics and abilities.

### 4.1 Duration of Gameplay.

The simulation times were determined based on the necessary number of tests required to gain a comprehensive understanding of the models [12]. Due to the moderate presence of random elements within these two models, and the fact that unlike a non-real time model and unlike a short task model, running this model for a large-scale run would be difficult. We conducted multiple runs to ensure a thorough comprehension of their behaviors.

Model B efficiently operated for extended hours, with an average run of 1 hour and a maximum of 4 hours until the game environment drastically changed to night mode. It successfully implemented declarative memories, production rules, and extended motor and vision functions. Conversely, Model A had a shorter-term capacity, maintaining its road position for two control loop iterations before deviating due to a misinterpreted danger pattern. This misinterpretation led to an incorrect steering action.

### 4.2 Accuracy of Recognizing Visual Patterns and Steer Efficiency.

Model B contains three visual patterns, including one shared with Model A to start the bus. The other two in Model B coordinate the control loop throughout gameplay. In Model A, the second pattern appears selectively within the dynamic environment. The number of visual patterns doesn't greatly impact their identification accuracy; instead, pattern stability plays a pivotal role.

When the game environment undergoes minimal changes, the models consistently demonstrate the ability to identify the visual patterns. However, Model A struggles with correctly recognizing the visual pattern that intermittently shows up within the

dynamic environment. It is worth noting that to enable Model A to recognize the fleeting visual pattern even at the beginning of the gameplay, we adopted a utility learning approach and utilized the “Spp” parameter to set the reward value of the "finding-danger-cue" production to 0.5, thereby enhancing the model's ability to detect the danger cue. However, this adjustment increased Model A tends to “see” the visual pattern and influenced the accuracy of identifying this, resulting in the mistaken identification of the visual pattern. On the other hand, Model B maintains a stable accuracy in identifying the patterns due to their continuous presence throughout the gameplay if the simulation environment maintains day mode without drastic change.

Both models effectively employ control loops, based on human behavior observation, to guide the bus back onto the road. Specifically, Model A steers the bus towards the road center when the predefined danger cue is detected in VisiTor. Conversely, Model B steers when the deviation between two visual cues exceeds 200 pixels. Model A's steering stability, however, depends on continuous perception of the visual pattern. After two steering rounds, it tends to misidentify the danger cue, primarily due to the initial reward parameter set for perceiving fleeting visual cues. As a result, the bus gradually veers off the left side of the road.

### 4.3 Response Time.

As indicated in Table 5, Model B exhibited an average time of 0.235 s to effectively locate visual cues using the Visicon in conjunction with simulated eyes. Furthermore, it took approximately 0.05 s for attention to be directed towards the visual cue, while around 0.2 s was required to decompose the cue's location and store it in the imaginal buffer. The total decision-making time for the model, encompassing the assessment of deviation and subsequent action decision of pressing the keys, amounted to 0.9 s. In contrast, for Model A, it took an average of 0.315 s to verify the absence of a “danger” visual cue and an average of 0.135 s to identify and locate the danger cue. As we mentioned previously setting the reward parameter of finding danger cue production facilitated the model's faster detection of the danger visual cue in the first two rounds of production rule fire. The total decision-making time for model A, considering the deviation assessment and subsequent action decision amounted to 0.7 s.

Both models demonstrated efficient reflection time, enabling them to promptly identify danger cues and steer the bus back onto the road, even when traveling at the maximum speed of 45 mph. This behavior aligns with human performance in the context of minutes, while over hours, the models surpassed human capabilities. Their ability to operate without experiencing fatigue or committing errors provides them with a significant advantage in accomplishing the task of driving the bus (e.g., see [13] for a model of fatigue affecting a task).

Our model demonstrated notable improvements compared to the bus driving model developed by [7], particularly in the accuracy of identifying visual cues and sustained operation. The integration of VisiTor into ACT-R 7 enabled enhanced coordination between perceptual and motor behavior. The full cycle of signal transmission, visual cue recognition, location extraction, deviation calculation, and action decision take 0.7-0.9 s, a significant improvement from the previous 6.01 s average time using

JSegMan with ACT-R 6. VisiTor's extensibility supports task-specific functions like the long key press, which prevents ACT-R key presses from being misinterpreted, enhancing the model's performance for long-term tasks.

**Table 4.** The output script of the ACT-R models that shows buffers, fired productions, and VisiTor commands with time stamps.

Model A

```
CL-USER> (run 10)
0.000   GOAL           SET-BUFFER-CHUNK GOAL GOER NIL
0.000   VISION         SET-BUFFER-CHUNK VISUAL-LOCATION CHUNK0 NIL
0.050   PROCEDURAL     PRODUCTION-FIRED GO
Ready to go
0.100   PROCEDURAL     PRODUCTION-FIRED PERCEIVE-ENVIRONMENT
0.150   PROCEDURAL     PRODUCTION-FIRED MOVE-ATTENTION
0.150   VISION         SET-BUFFER-CHUNK VISUAL-LOCATION CHUNK0
0.200   PROCEDURAL     PRODUCTION-FIRED AHEAD
hiii
continuouspress a key!
0.200   MOTOR          PUNCH HAND RIGHT FINGER INDEX
0.235   VISION         SET-BUFFER-CHUNK VISUAL CHUNK2
0.285   PROCEDURAL     PRODUCTION-FIRED RECHECK-ENVIRONMENT-NO-CUE
0.285   VISION         SET-BUFFER-CHUNK VISUAL-LOCATION CHUNK0
0.550   PROCEDURAL     PRODUCTION-FIRED NO-FINDING-CUE
0.550   VISION         SET-BUFFER-CHUNK VISUAL-LOCATION CHUNK0
0.600   PROCEDURAL     PRODUCTION-FIRED PERCEIVE-ENVIRONMENT
0.650   PROCEDURAL     PRODUCTION-FIRED MOVE-ATTENTION
0.650   VISION         SET-BUFFER-CHUNK VISUAL-LOCATION CHUNK0
0.700   PROCEDURAL     PRODUCTION-FIRED AHEAD
 hiii
 continuouspress a key!
0.700   MOTOR          PUNCH HAND RIGHT FINGER INDEX
0.735   VISION         SET-BUFFER-CHUNK VISUAL CHUNK3
0.785   PROCEDURAL     PRODUCTION-FIRED RECHECK-ENVIRONMENT-CUE
0.835   PROCEDURAL     PRODUCTION-FIRED FINDING-CUE
0.835   VISION         SET-BUFFER-CHUNK VISUAL-LOCATION CHUNK1
0.900   PROCEDURAL     PRODUCTION-FIRED STEER
```

Model B

```
CL-USER>  (run 10)
0.000  GOAL        SET-BUFFER-CHUNK GOAL GOER NIL
0.000  VISION      SET-BUFFER-CHUNK VISUAL-LOCATION CHUNKO NIL
0.050  PROCEDURAL  PRODUCTION-FIRED GO
Ready to go
0.100  PROCEDURAL  PRODUCTION-FIRED PERCEIVE-ENVIRONMENT
0.150  PROCEDURAL  PRODUCTION-FIRED MOVE-ATTENTION
0.150  VISION      SET-BUFFER-CHUNK VISUAL-LOCATION CHUNKO
0.200  PROCEDURAL  PRODUCTION-FIRED AHEAD
hiii
continuouspress a key!
0.200  MOTOR       PUNCH HAND RIGHT FINGER INDEX
0.235  VISION      SET-BUFFER-CHUNK VISUAL CHUNK2
0.250  PROCEDURAL  PRODUCTION-FIRED RECHECK-ENVIRONMENT
0.285  VISION      SET-BUFFER-CHUNK VISUAL CHUNK3
0.300  PROCEDURAL  PRODUCTION-FIRED DANGER
0.350  PROCEDURAL  PRODUCTION-FIRED FINDING-DANGER
0.350  VISION      SET-BUFFER-CHUNK VISUAL-LOCATION CHUNK 1
0.485  VISION      SET-BUFFER-CHUNK VISUAL CHUNK4
0.535  PROCEDURAL  PRODUCTION-FIRED MOVE-ATTENTION-DANGER
0.585  PROCEDURAL  PRODUCTION-FIRED WHEREISDANGER
0.620  VISION      SET-BUFFER-CHUNK VISUAL CHUNK5
0.785  IMAGINAL    SET-BUFFER-CHUNK-FROM-SPEC IMAGINAL
0.835  PROCEDURAL  PRODUCTION-FIRED WHEREISCENTER
1.035  IMAGINAL    SET-BUFFER-CHUNK-FROM-SPEC IMAGINAL
1.085  PROCEDURAL  PRODUCTION-FIRED CALCULATE-DEVIATION
1.135  PROCEDURAL  PRODUCTION-FIRED CONSIDER-STEER
```

## 5 Discussion and Conclusion

This study employed ACT-R 7.X and VisiTor to conduct a bake-off experiment simulating driving behaviors, with Model B running for up to four hours and outperformed other models in all three categories and Model A for a shorter term. The game environment remained unaltered, with ACT-R's perceptual-motor module and VisiTor enabling interaction. Our models' extensive runs provided insights into model behavior. Analysis of ACT-R outputs illustrated effective coordination of motor and vision. Developed models successfully steered the bus within 1s at speeds below 45 mph, suggesting faster speeds might require shorter transport signals. Comparing the results of the models' execution, we observed that in the dynamically changing environment, the perceptual module in conjunction with VisiTor exhibits limitations in perceiving intermittent visual patterns and accurately identifying these patterns when significant alterations occur in the simulation environment.

There are also limitations in the central modules of ACT-R. These limitations include the lack of consideration for physiological factors such as fatigue or decreasing correction rates over time. [7] suggested that incorporating physiology with ACT-R could make the model more realistic. We agree with this point and plan to add that to our future work. This new approach can help with testing the compound effects of fatigue and learning rate on our model.

ACT-R + VisiTor playing *Drive the Bus* provides an excellent platform for studying the interaction of vision, attention, errors, and fatigue. It is in many ways a more naturalistic task than the PsychoMotor Vigilance task (PVT). We can now explore an existing fatigue model, and further examine fatigued driving, visual attention, the need for micuration, and modeling the details of interaction.

We could gain understanding about how long-term and repetitive physical activities, like driving a bus for an extended period, affect human performance. It remains to be seen if this task is more like the PVT or like motor control in how it is affected by sleep restriction. This task would also allow us to determine whether psychological factors could potentially harm or the increasing of learning rate due to the practice would enhance driving skills. We could also introduce additional variables, such as caffeine, to examine their combined impact.

## References


1. Newell, A.: Unified Theories of Cognition. Harvard University Press (1990).
2. Anderson, J. R.: How can the human mind occur in the physical universe? New York, NY: Oxford University Press (2007).
3. Ritter, F. E., Tehranchi, F., Oury, J. D.: ACT-R: A cognitive architecture for modeling cognition. WIREs Cognitive Science, 10(3), e1488 (2019).
4. Ritter, F. E., Baxter, G. D., Jones, G., Young, R. M.: Supporting cognitive models as users. ACM Transactions on Computer-Human Interaction, 7(2), 141–173 (2000).
5. Byrne, M. D., O'Malley, M. K., Gallagher, M. A., Purkayastha, S. N., Howie, N., Huegel, J. C.: A preliminary ACT-R model of a continuous motor task. In: Human Factors and Ergonomics Society Annual Meeting on Proceedings, 54(13) (2010).
6. Bagherzadeh, A., Tehranchi, F.: Comparing cognitive, cognitive instance-based, and reinforcement learning models in an interactive task. In: ICCM, The 20th International Conference on Cognitive Modeling on proceedings. 1-7 (2022).
7. Schwartz, D. M., Tehranchi, F., Ritter, F. E. Drive the bus: Extending JSegMan to drive a virtual long-range bus. In: 18th International Conference on Cognitive Modeling on proceedings. 241-246 (2020).
8. Pew, R. W., Mavor, A. S. (Eds.).: Human-system integration in the system development process: A new look. Washington, DC: National Academy Press. books.nap.edu/catalog/11893, checked May 2019 (2007).
9. Wu, S., Bagherzadeh, A., Ritter, F., Tehranchi, F.: Long Road Ahead: Lessons Learned from the (soon to be) Longest Running Cognitive Model. Paper accepted by 21st International Conference on Cognitive Modeling (ICCM) at the University of Amsterdam, the Netherlands (2023).
10. Kukreja, U., Stevenson, W. E., Ritter, F. E.: RUI—Recording User Input from interfaces under Windows and Mac OS X. Behavior Research Methods, 38(4), 656–659 (2006).
11. Carley, K. M.: Validating computational models (No. CMU-ISR-17-105). Pittsburgh, PA: School of Computer Science, Carnegie Mellon University (2017).
12. Ritter, F. E., Schoelles, M. J., Quigley, K. S., Klein, L. C.: Determining the number of model runs: Treating cognitive models as theories by not sampling their behavior. In L. Rothrock & S. Narayanan (Eds.), Human-in-the-loop simulations: Methods and practice pp. 97-116 . London: Springer-Verlag (2011).