Ritter, F. E., Tehranchi, F., Dancy, C. L., & Kase, S. E. (2020). Some futures for cognitive modeling and architectures: Design patterns that you can too. *Computational and Mathematical Organization Theory*, *26*, 278–306.

Some Futures for Cognitive Modeling and Architectures: Design Patterns for Including Better Interaction with the World, Moderators, and Improved Model to Data Fits (and So Can You)[1]

Frank E. Ritter (frank.ritter@psu.edu), Farnaz Tehranchi (farnaz.tehranchi@psu.edu),
Christopher L. Dancy (christopher.dancy@bucknell.edu), and
Sue E. Kase (sue.e.kase.civ@mail.mil)

5 jan 2020 Ritter-BRiMS-futuresV22.doc

Keywords: cognitive architecture; vision models; interaction; user models; emotions; moderators; model fit; model optimization; ACT-R; genetic algorithm; pattern matching

Abstract

We note some future areas for work with cognitive models and agents that as Colbert (2007) notes, "so can you". We present three approaches as something like design patterns, so they can be applied to other architectures and tasks. These areas are: (a) Interacting directly with the screen-as-world. It is now possible for models to interact with uninstrumented interfaces both on the machine that the model is running on as well as remote machines. Improved interaction can not only support a broader range of behavior but also make the interaction more accurately model human behavior on tasks that include interaction. Just one implication is that this will force models to have more knowledge about interaction, an area that has been little modeled but essential for all tasks. (b) Providing the cognitive architecture with more representation of the body. In our example, we provide a physiological substrate to implement behavioral moderators' effects on cognition. Cognitive architectures can now be broader in the measurements they predict and correspond to. This approach provides a more complete and theoretically appropriate way to include new aspects of behavior including stressor effects and emotions in models. And (c) using machine learning techniques, particularly genetic algorithms (GAs), to fit models to data. Because of the model complexity, this is equivalent to performing a multi-variable non-linear stochastic multiple-output regression. Doing this by hand is completely inadequate. While there is a danger of overfitting using a GA, these fits can help provide a better understanding of the model and architecture, including how the architecture changes under moderators such stress. This paper also includes some notes on model maintenance and reporting.

Acknowledgments

Stephen Colbert's (2007) *I am America (and so can you!)* provided inspiration for the title. This report was supported by ONR (N00014-15-1-2275). It is based on a plenary presented at BRIMS 2018. The work reported has been supported by a wide range of sponsors noted in the individual reports. Ritter would like to thank his collaborators, including the ACS Lab, Agent Oriented Systems (Lucas, Evertsz, Pedrotti), Jeanette Bennett, Jen Bittner, Robert Hester, Laura Klein, Drew Pruett, Robert St. Amant, Mike Schoelles, Courtney Whetzel, and folks at Charles River Analytics (Weyhrauch, Niehaus, Lynn). This report was improved by comments from Cesar Colchado, Joseph DiPalma, Raphael Rodriguez, David Schwartz, and two kind, helpful, anonymous reviewers. Steve Crocker provided particularly

[1] Colbert (2007) provided inspiration for the title.

helpful comments on what we thought was a clean manuscript. Any errors, of course, remain the fault of the authors.

Introduction

In this paper, we note some future areas for work with cognitive models created within cognitive architectures that may also be useful for AI agents. These models and agents are important for cognitive science and also important for agent-based modeling and computational organizational theories because they provide broader cognitive social agents. This report is based on a plenary talk at the 2018 International Conference on Social Computing, Behavioral-Cultural Modeling & Prediction and Behavior Representation in Modeling and Simulation (SBP-BRiMS) conference and has been slightly expanded and extended. This work helps fulfill the vision outlined in Newell (1990, Figures 8.1 and 8.2).

These future areas are ones that we have done work in, and we provide examples of what progress can be made in these areas. In each area, the example can either be duplicated or modified. The examples can be seen as a type of design pattern for work in that area. (A design pattern is a re-usable form of a solution to a problem; this concept has been used in computer science, Gemma et al., 1995.) The areas are ones that also appear in previous reviews (e.g., Pew & Mavor, 1998, 2007; Ritter et al., 2003), but now are further realized in these examples.

The three areas are (a) Interacting directly with the screen-as-world. It is now possible for models to interact with uninstrumented interfaces both on the machine that the model is running on as well as remote machines. Improved interaction can not only support a wider range of behavior but also make the interaction more accurately model human behavior on tasks that include interaction. Just one implication is that this will force models to have more knowledge about interaction, an area that has been little modeled, but is essential for all tasks.

(b) Providing the cognitive architecture with more representation of the body. In our example, we provide a physiological substrate to implement behavioral moderators' effects on cognition. Cognitive architectures can now be broader in the measurements they predict, the mechanisms they include, and effects they correspond to. This approach provides a more complete and theoretically appropriate way to include new aspects of behavior including stressor effects and emotions in models.

(c) Using machine learning techniques to fit the model to data. Because of model complexity, this fitting process is equivalent to performing a multi-variable non-linear stochastic multiple-output regression. Doing this complex alignment by hand seems completely inadequate. While there is a danger of overfitting, these fits, particularly across multiple fits, can help provide a better understanding of the model, including how the architecture changes under stress or other moderators and how people vary. We will present how we used a genetic algorithm to do this.

These approaches are presented essentially as design patterns, so they can be applied to other architectures in addition to the ones we used. Overall, these approaches provide areas for improving models drastically if they are widely applied. They also will make our jobs harder, in that we will have to take account of a wider range of human behavior. In addition to the high-level view throughout the paper there will be some low-level advice about modeling.

Before describing each of these areas, we start with a very brief overview of cognitive architectures for readers not familiar with them and to ground what we mean by cognitive architecture.

Cognitive Models and Architectures

Cognitive models are simulations that predict human cognition and provide possible explanations of human cognition or an application for practical use. Cognitive architectures are infrastructures that

provide a fixed set of computational mechanisms that represent the fixed mechanisms of cognition used to generate behavior for all tasks (Newell, 1990). As a fixed set they provide a way for combining and applying cognitive science theory. Cognitive architectures are essentially programming languages specifically designed for modeling cognition, such as Soar (Laird, 2012; Newell, 1990) or ACT-R (Anderson, 2007; Anderson et al., 2004; Ritter, Tehranchi, & Oury, 2019). When knowledge is added to a cognitive architecture, essentially providing a program, a cognitive model is created.

Thus, cognitive modeling can be seen as a form of task analysis and, as such, is congenial to many areas and aspects of human factors. The primary purpose of cognitive models is to summarize and explain human cognition. By summarize, we mean: to explain a wide range of regularities of human behavior within a single theory. Cognitive models outside of cognitive architectures encode different aspects of human cognition with singular, one-use computer programs not necessarily intended for composition or reuse.

A user model within a cognitive architecture is thus a combination of (a) task knowledge and (b) a cognitive architecture with its fixed mechanisms to apply the task knowledge to generate behavior. Running these models simulates human cognition and predicts human performance based on information processing. Cognitive architectures have been both a research tool for theory building and an engineering tool for applying the theory. As an engineering tool they provide a way to use simulations of humans in applications, such as example users, realistic opponents in games, or teammates.

These architectures may also include interaction with the world, either as a function or with theoretical intent. They may attempt to include affect or emotions, and there is continuing work to extend the architectures to include more mechanisms (e.g., hearing, physiology, temporal and spatial reasoning).

ACT-R is an example cognitive architecture. It has two basic types of knowledge, declarative and procedural. Declarative knowledge (called “chunks” in ACT-R) contains facts, images, locations, or sounds. Procedural knowledge (production rules) represents behavioral aspects of performance with goals, operators, methods, and selection rules. ACT-R runs a pattern matching process that finds production rules with conditions that match the current state of the system (conditions are the so-called left-hand side, LHS, of production rules). When the conditions match the actions of the matching production rule will be fired or applied (so-called right-hand side, RHS). Productions can make a request to an ACT-R buffer and then that buffer's module will perform that function. For example, it may place a chunk into another buffer. The combined model, knowledge+architecture, can be tested by comparing the resulting model’s performance, such as time to perform the task, accuracy on the task, and the order of task actions, with the results of people doing the same task.

With this summary in hand for readers not familiar with cognitive models and architectures, we now turn to the three approaches that can be extended.

Better Interaction with the World

Ritter et al. (2001) and others (e.g., Gray, 2002; Gray, 2007) have argued previously that models need access to a world to perform interactive tasks. Models have often performed an abstracted version of the task or having access to the whole world at once. For simple problems like the Tower of Hanoi, this may be sufficient. For complex tasks requiring where the state of the world must be scanned and learned or that varies with actions, where active vision (Findlay & Gilchrist, 2003; Gray, 2008) is required, it is far less appropriate. A similar problem arises when action is abstracted too far.

Figure 1 shows an approach to providing cognitive models access to the same interfaces that a user sees. This extends how human hands and eyes interact with a User Interface Management System

(UIMS), to provide a Cognitive Model Interface Management System (CMIMS) to allow a cognitive model to interact with an interface.

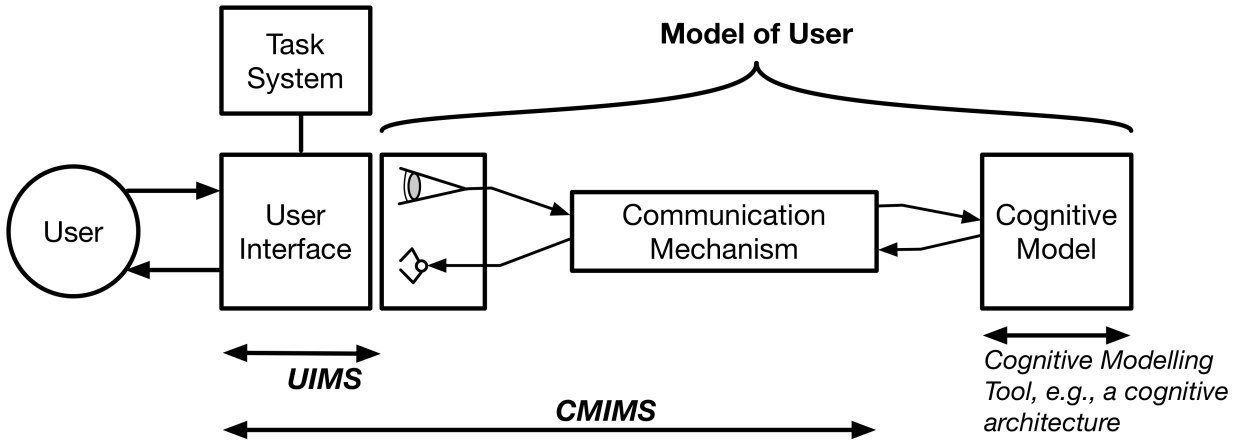


Figure 1. How a cognitive model interacts with the interface through a cognitive model interface management system, similar to a user interface management system. (Revised from figure in Ritter, Baxter, Jones, & Young, 2000.)

There are several ways to create a CMIMS. (a) You can instrument the task system. In this approach, the model connects to the system so that the model can pass commands to the task and be passed information about the task's state—the model's eyes and hands may or may not see what the user sees and can do. This is useful if you have access to the system internals, and do not care about the cognitive plausibility of the connection (or perhaps its details). Agents that are used in military simulations are often examples of this (e.g., Tambe et al., 1995). The cognitive plausibility of the connection in these situations is often not important. For this approach, Ong (1994) created MONGSU as a general solution to the problem of connecting ACT-R to a wide-range of task environments. His work proposed the idea of communicating between two processes through a UNIX socket. However, establishing the socket connection requires an additional application and advanced knowledge of programming.

Hope et al. (2014) presented a simplified interaction scenario between cognitive models and task environments through a JSON network interface. To use this connection, a new ACT-R device module (JNI module) for handling the communication must be added to the cognitive model. This method has been tested in different environments, such as the REACT-R module that uses the JNI module to connect ACT-R with a Unity 3D simulation (Salt, Wise, Sennersten, & Lindley, 2016). Yet, the current JNI API version does not support all the ACT-R motor and visual functions and does not have an independent interface. Thus, this approach is more useful for advanced users and where the interest is not in the interaction per se, but in having a behavior generator or in aspects of the behavior such as learning or reaction to the behavior by other agents.

(b) You can modify the graphics library that the system uses to draw the interface and have the modified version pass information to the model. This enables interfaces that use a graphics library like Swing to be able to pass information to and from a model about what is on a display. Bass, Baxter, and Ritter (1995) created the first models using this approach in Garnet and Tcl/Tk. The approach of modifying the interface or working in an interface language has also been implemented in ACT-R/PM (Byrne & Anderson, 1998). ACT-R/PM allows ACT-R models to interact with interfaces built within a special Common Lisp window.

Another successful version of this method is Salvucci's (2006) Cognitive Code, an extension of ACT-R that provides access to external environments. He also introduced a new implementation of ACT-R in

Java that can be used as a library in other ACT-R projects (Salvucci, 2009, 2013). Here, the ACT-R motor and vision module have been hard-coded, not configurable, and limited to a Java applet in the Cognitive Code. Antetype (http://www.antetype.com) also appears to work in this way communicating with a user interface design tool (Wallach, Fackert, & Vladimir Albach, 2019). However, there are drawbacks to this approach. It becomes difficult to create an eye and hand for each graphics library, not all interfaces use a modifiable graphics library, and some models need to interact with an interface that is not implemented in a way that it is modifiable in this way.

(c) Build a whole new task that the model can interact with. This approach is additional work, but has been done on numerous occasions (e.g., Dancy & Ritter, 2017a; Taatgen, 2002).

And (d) interact based on what is displayed on the screen and pass actions into the operating system. This final approach is the most general and works for all interfaces. This is the approach that SegMan uses. SegMan was initially developed by St. Amant, and was extended through use by Ritter and his colleagues working with St. Amant and his students (Ritter, Van Rooy, St. Amant, & Simpson, 2006; St. Amant, Horton, & Ritter, 2007; St. Amant & Riedl, 2001; St. Amant, Riedl, Ritter, & Reifers, 2005).

SegMan is based on general vision algorithms and a theory of how an interface may be an easier world for models to interact with than other environments (St. Amant, 2000). For example, interfaces are more regular than other environments and often have a small number of objects to recognize. Figure 2 shows how SegMan, where it provides the simulated eyes and hands, connects the ACT-R cognitive architecture to interfaces of computer environments. We have primarily used SegMan with ACT-R but this approach and SegMan would work with any cognitive architecture.

SegMan works by segmenting the screen to find different segments, pixel groups, and structures it knows about (e.g., buttons) and applying multiple vision algorithms to find different types of objects. It gets the pixel map from the operating system, and mouse and keystrokes are created by passing them to the operating system's input queue.

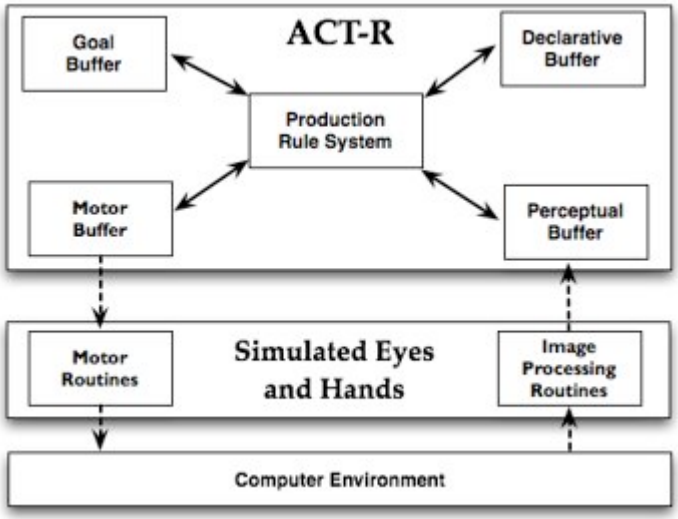
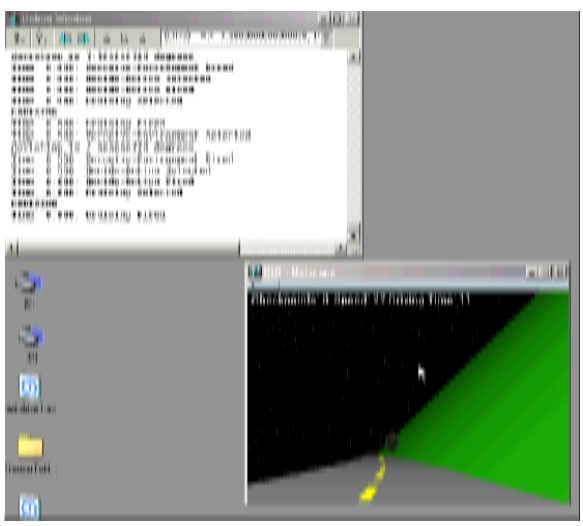


Figure 2. Simulated eyes and hands mediate ACT-R's interaction with the world.

SegMan provided eyes and hands to several models and was used to interact with a wide range of interfaces (Ritter, Kukreja, & St. Amant, 2007; St. Amant & Riedl, 2001; St. Amant et al., 2005). It was never fully completed but was used by several models, including ones that drove simulated cars (Ritter et al., 2006), robots (Ritter, Kukreja, et al., 2007), and a variety of online systems including driving games, robot operation, and other screen-based interfaces, including one designed not to be used by agents (St. Amant & Riedl, 2001; St. Amant et al., 2005). Examples are shown in Figure 3

where (a) an ACT-R model drives in an uninstrumented Java-based driving game (Ritter et al., 2006), (b) an ACT-R model drives a robot (Ritter, Kukreja, et al., 2007), and (c) an agent plays an online casino's practice roulette table to test a theory of gambling (St. Amant et al., 2005).



(a) Driving in a simple Java game, (b) Driving a robot

(c) Playing an offshore casino's practice roulette.

Figure 3. Example applications of SegMan.

Current Version, JSegMan

The latest version of this approach, JSegMan, updates SegMan. JSegMan creates a way to interact with all interfaces using an extended Java library (Robot package) to input motor commands

(keystrokes, mouse moves, and mouse clicks), and an open source library to help with image processing (Sikuli package) based on OpenCV.

Visual patterns are small images that represent visual objects in cognitive architectures—visual chunks in ACT-R. JSegMan parses the screen and uses the Template Matching method to find the target, the visual pattern, and area. Template Matching is a pattern-matching algorithm that compares a template (small image) against the overlapped image regions (the computer screen) pixel by pixel; the area that has the maximum matching score is the target area. JSegMan can identify pre-defined visual patterns, which are defined in a similar way as memory chunks (Tehranchi & Ritter, 2018).

As the most recent and perhaps most extensive example of using a simulated eye and hand to interact with an interface, we extended an existing large ACT-R model (Paik, Kim, Ritter, & Reitter, 2015) to perform the Dismal spreadsheet task. The Dismal task has 14, non-iterated subtasks and takes about 20 min. to perform the first time (Kim & Ritter, 2015). The model has initially 617 rules in fully expert mode. In novice mode, it has 29 rules and 1,152 declarative memory task elements. Figure 4 shows the unmodified task on the left-hand side of the figure that the model interacts with, and the running ACT-R+JSegMan model on the right-hand side.

This model makes several contributions. JSegMan allows the model to not just model doing the task, but to actually perform the task. That is, it actually does the complete, uninstrumented task in real time. It performs a large, 20 min., non-iterated task with 14 subtasks. By non-iterated, we mean that each subtask is not repeated (unlike repeating a decision for multiple, similar stimuli). In implementing the task, we adjusted 162 declarative chunks in the original Dismal model by adding a new slot for visual objects. In addition, to model eye movements, we added 52 new visual objects and visual locations. When we ran the resulting model we found one missing click in the original model. Also, most of the key press requests to the motor module required a hand/finger adjustment. These differences were visible because the results in the interface did not match the expected output. Matching behavior in an interface can be an important way to validate models in the future.

The use of JSegMan also provided a better fit to the human data. Table 1 shows that the model with JSegMan predicted the response times more accurately while, importantly, using the same, unmodified interface that the human subjects used. The correlation improvement is probably not a reliable increase, but the difference in MSE appears to be (Tehranchi & Ritter, 2018).

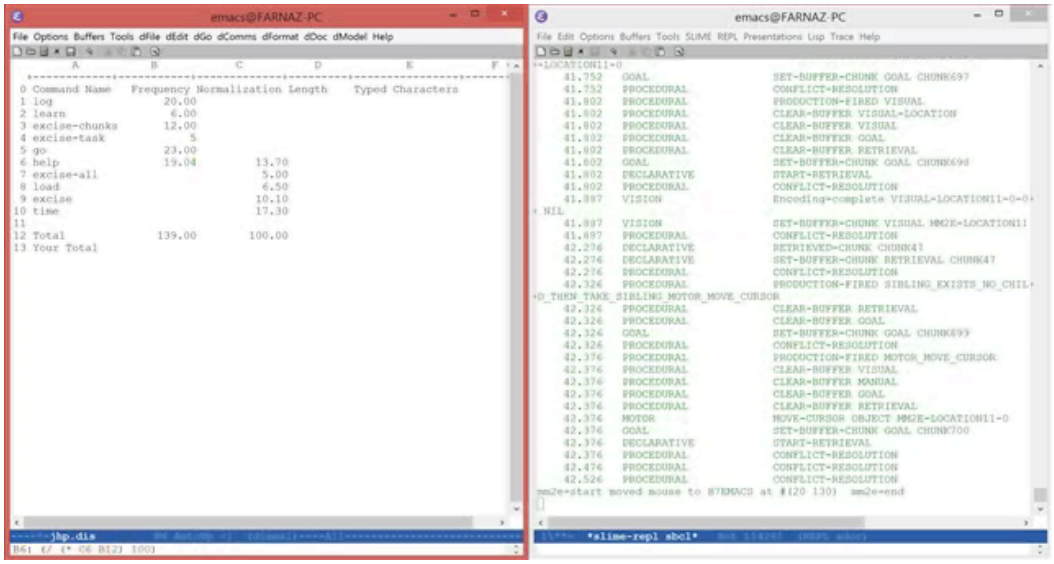


Figure 4. On the left side, the Dismal spreadsheet task. On the right side, the trace of ACT-R+JSegMan.

Table 1: The mean (SD) task completion time in seconds for the four learning sessions for the original Dismal mouse-interface task for humans, an ACT-R model (Paik et al., 2015) and the ACT-R model with JSegMan (Tehranchi & Ritter, 2018), as well as the correlations (R2) and MSE between the human data and the models.

Day	Human (N=30)	Original ACT-R Model (N=100)	ACT-R Model with JSegMan Hands+Eyes (N=30)
1	1366 (60.8)	1326 (12.1)	1339 (11.7)
2	894 (26.6)	891 (6.1)	894 (6.5)
3	727 (25.5)	693 (4.5)	704 (5.0)
4	659 (22.7)	594 (5.8)	614 (4.4)
R2		.997	.9984
MSE		1745	820

Summary: Interaction with the World

There are several approaches used by modelers to interact with the world. Previous systems illustrating these approaches provided lessons for our work. The most satisfying approach in the end, we believe, will be interacting with the user's interface to parse the screen bitmap and generating keystroke and mouse move actions to the operating system.

Environmental interaction will remain an interesting task for models because it is a core aspect of human behavior on nearly all tasks. A colleague who should remain anonymous once commented that you do not need to create a general way for models to interact with a task because if you do not "It only takes 25% of every project"[2]. If this aspect of behavior is created for each model-interface pair, it is not a fixed mechanism, and thus interaction for this model is not architectural. A general way to interact is worth providing to make interaction part of the architecture. Creating a fixed set of mechanisms and reusing them as part of cognitive architectures is thus not just good science in that we have a fixed architecture and can use it to summarize findings, it is also good engineering because the system is reusable and widely applicable.

By providing a reusable way to interact with tasks, we will receive better fits to human data and better summaries of human data because we are modeling more of the task. We will be able to see errors, mistakes, and slips more easily (as we did in moving the Dismal model to interact directly with the task) (e.g., Gray, 2000), and should be able to see and model error correction strategies. We should also be able to unify more of human modeling, including vision, visual search, visual recognition, and even finding the mouse (Gray, 2008). We will see better applications, from models to test interfaces to models that can act as teammates or assist users, because they can see what the user sees, using the system to compute what is visible (Ritter, 2019). This approach provides a whole new world for models, even the world itself through web browsers and video cameras.

[2] Just to be clear, this seems like a horrible tax to pay.

Moderators of Cognition

After modelers started to work with fixed sets of mechanisms to be shared across models—architectures (Newell, 1990)—the question arose about how did the mechanisms change? The mechanisms might change as children grow into adults (Jones, Ritter, & Wood, 2000). The mechanisms might also change their processing behavior due to behavioral moderators such as stress, fatigue, sleep deprivation, drugs such as caffeine and nicotine, or emotions. This raised the question about how to represent these changes.

Ritter (1993) in a workshop paper for Sloman's workshop on representing emotions in cognition noted that these effects might arise in three ways: (a) simple knowledge about emotions, where the agent reasons about emotions, (b) thoughts that effect physiology (e.g., becoming scared at a noise because of what it might represent), and (c) physiology changes that lead to changes in the mechanisms of cognition (e.g., being hungry or consuming caffeine and having this modify processing). The second two are more interesting to those interested in how cognition changes due to moderators.

Modeling Moderators with Overlays

The first approach we took to modeling moderators was to modify the parameters of ACT-R to represent changes due to moderators (Ritter, Reifers, Klein, & Schoelles, 2007). For example, we represented arousal by increasing the processing speed, and represented sleep fatigue by decreasing processing speed and increasing working memory decay. We used this approach to summarize the effect of caffeine on cognition (G. P. Morgan, Ritter, Stine, & Klein, 2006), and this work also led to creating a phone app (Ritter & Yeh, 2011) to display the pharmacokinetics and pharmacodynamics of caffeine that has had over 100k downloads (caffeinezone.net). We also used this approach to model the effects of moderators in architectures, two of which we use to illustrate what it can do. But, with further work, we moved from representing moderators as fixed, changes to architecture parameters throughout a run of a model to dynamic modifying parameters during the run of a model to represent state changes caused by the model's thinking, body, and the environment.

We used overlays within the Cognitive Jack (CoJACK) project to create moderated behavior. Jack is the Java Agent Construction Kit (Busetta, Rönnquist, Hodgson, & Lucas, 1999). We added aspects of ACT-R's memory equations and some aspects of Soar's trace (Norling & Ritter, 2004) to modify an agent architecture to be more cognitive. We did this as an exploration of usability because one review suggested that it would be easier to build a new architecture than to make Soar easy to use (Shakir, 2002).

Figure 5 shows a schematic of how CoJACK uses moderators. The basic Jack architecture is a Belief-Desires-Intentions (BDI) architecture (Busetta, Howden, Rönnquist, & Hodgson, 1999). CoJACK might be better understood as an architecture that tries to apply plans to its knowledge and beliefs. The programmer creates plans, relatively large structures (which helps make CoJACK usable) that the architecture attempts to start to apply (or *instantiate* in that literature) based on what is in working memory. Working memory is made up of what previous plans have put there, instantiated plans, and what the agent has learned from the world. As more actions become possible in the plans (they have conditional subtasks in them), they produce further behavior. The timing and the errors in these steps in CoJACK are based on memory and other timing and error equations taken from ACT-R (Norling & Ritter, 2004).

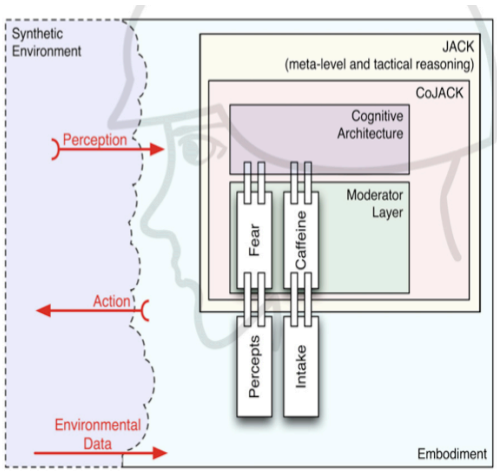


Figure 5. Schematic of CoJACK and how it uses moderators.

We created several overlays to the architecture to represent the effects of caffeine, and the effect of challenging (which might be seen as confident) and threatening (which might be seen as fear) task appraisals (Ritter et al., 2012). These changes lead to different behavior in a tank game.

Figure 6 shows another environment we put moderated agents into (Evertsz, Pedrotti, Busetta, Acar, & Ritter, 2009). In this world, the agents had varied levels and causes of fear and motivation. These agents exhibited a wider range of behavior than previous agents, including acting in a variable way when walking, based on their surroundings.

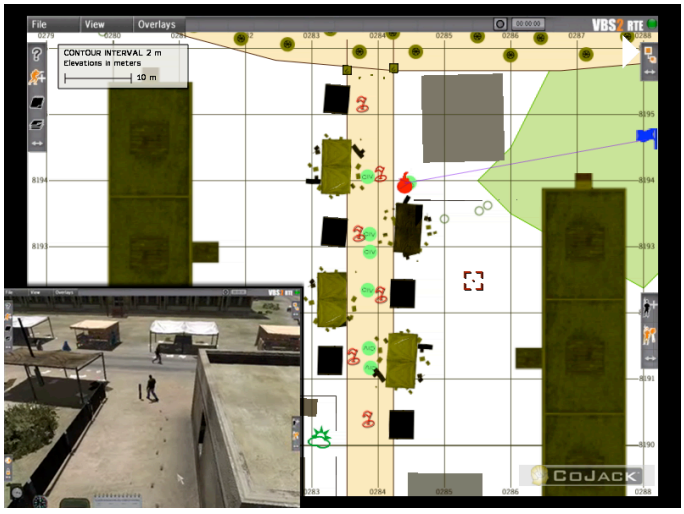


Figure 6. A VBS2 simulation of walking through a market area.

Modeling Grossman's Theory of Participation

We also used an overlay approach to implement Grossman's (1996) formula for participation (J. H. Morgan, Morgan, & Ritter, 2010; J. H. Morgan, Morgan, Ritter, & Poncelin de Raucourt, 2009). Grossman's formula originally just noted several factors that influenced participation in battle. This equation is shown in Eq 1. The factors are the participant's group size (larger leads to more participation), group composition (including how close the group is to the participant physically and socially, closer leads to higher participation), social distance from the opponent (further social distance leads to greater participation), the distance to the target (further away leads to more participation), mutual support (greater group cohesion leads to more participation), how close and clear the

participant's commander is (closer and clearer leads to greater participation), and target attractiveness (closer task alignment with the participants internal motivations leads to greater participation) and that some personality types are more likely to participate (tendencies).

(Equation 1) **Participation = (group size) x (group composition) x (social distance) x (spatial distance) x (mutual support) x (demands of authority) x (target attractiveness) x (tendencies)**

We implemented this as a logistic formula to provide participation prediction probabilities between 0 and 1 for simple agents in a tank game with fixed team sizes (we did not include social distance or target attractiveness because they did not change, and did not include tendencies) (J. H. Morgan et al., 2010). We used the resulting values to moderate when agents participated in a tank game.

The results of using this equation are shown in Figure 7. The agents that are moderated in this way vary in how they participate in ways that other agents typically do not, including running away.

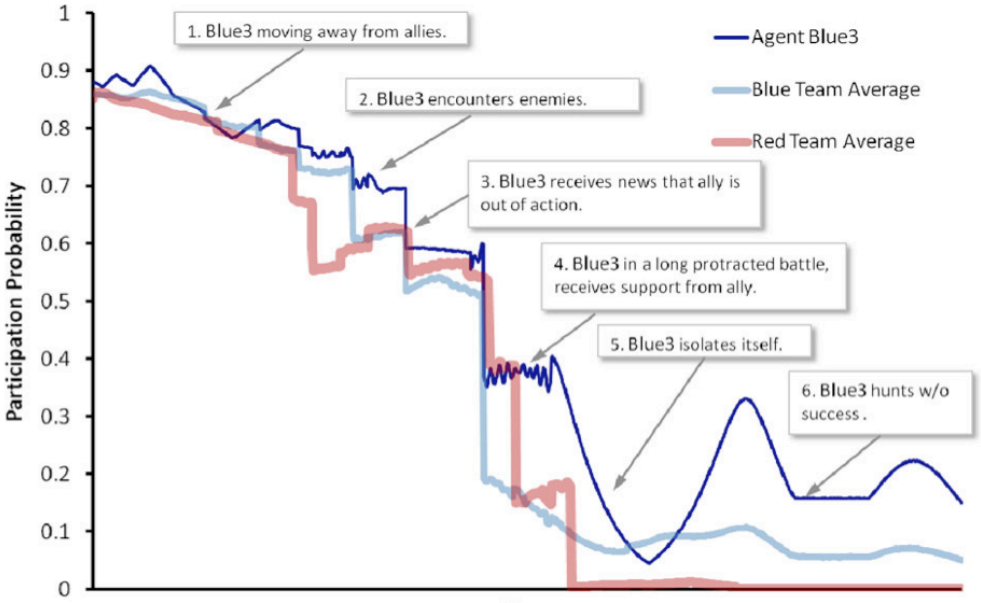


Figure 7. The probability of participating over the course of a game.

These projects in two architectures show that applying overlays to cognitive architectures is useful for modeling the effect of single moderators. These overlays allow a modeler to explore how an individual moderator may affect behavior in various contexts.

There are problems with this approach, though. The combination of individually developed overlays can be very difficult. For example, the overlays in CoJACK give different behavior depending on the order that they are loaded, and do not otherwise provide a theoretically grounded way to combine the effects of multiple moderators. The use of the participation equation leads to moderated behavior but does not provide a principled way to moderate how it interacts with other moderators. A better, more generalizable approach appeared to be to model the underlying physiology.

Modeling Moderators with Physiology: ACT-R/Φ

A more sophisticated way to approach modeling moderators of cognition is to represent overlays within models of physiology. This provides a more straightforward and tractable way to combine moderators because many of the moderators that are modeled are, at some level, modulating physiological processes, which can then have bottom-up effects on cognitive processes.

By including systematic models of physiology (based on *many* equations) instead of more simple overlays (essentially, one equation), an architectural perspective can be more directly used when including multiple moderators. Thus, using computational models of physiology allows one to explore how bottom-up (i.e., physiology to cognition) and top-down (i.e., cognition to physiology) influences may interact with moderators. With more encompassing models of physiology (e.g., see Hester et al., 2011, for a discussion of models of physiology) one can explore how two moderators may affect different physiological processes initially, but cause downstream effects to moderate the same physiological variables, which then affect cognition.

One such example of combining a systematic model of physiology with a cognitive architecture is that of ACT-R/Φ ("act-are-phi", Dancy, Ritter, & Berry, 2012; Dancy, Ritter, Berry, & Klein, 2015). It is a combination of ACT-R and a model of physiology, thus, phi from the Greek word physiology. This hybrid system combines ACT-R with HumMod (Hester et al., 2011). As a way to understand HumMod we have created a HumMod manual (Brener et al., 2019). Initially ACT-R/Φ was used to simulate the effects of stress on mental arithmetic but it has also been used to simulate other interactions between physiological systems and cognitive processes (Dancy, 2019; Dancy & Kim, 2018; Dancy, Ritter, & Gunzelmann, 2015; Kim, Dancy, & Sottilare, 2018).

In work with mental arithmetic, Dancy et al. (2015) used ACT-R/Φ to further explore potential within-task state changes. Using these simulations, they demonstrate how one may use this type of model and architecture (with dynamic physio-cognitive interactions) to understand how a model's state at any given point in time may result in different task performance if such a state was kept constant.

Figure 8 shows the results from using an ACT-R/Φ model to supply parameters for an ACT-R model that does serial subtraction. The ACT-R/Φ model does four 4-minute blocks of repeatedly subtracting 7 or 13 from a running total. During each second of the model simulation runs (N=200), the value of the declarative memory noise parameter (which is modulated by changes in the simulated physiology) was recorded. We then used each of these recorded declarative memory noise parameter values to run a cognitive model in ACT-R (i.e., with physiology turned off) that had static parameters while the simulation ran. Over multiple runs (N=200) of each "static" ACT-R model, the mean and standard deviation in performance were computed. This allowed us to understand how physiology affected the models second-by-second performance during the task, including how stress from previous blocks compounds in later blocks to modulate performance.

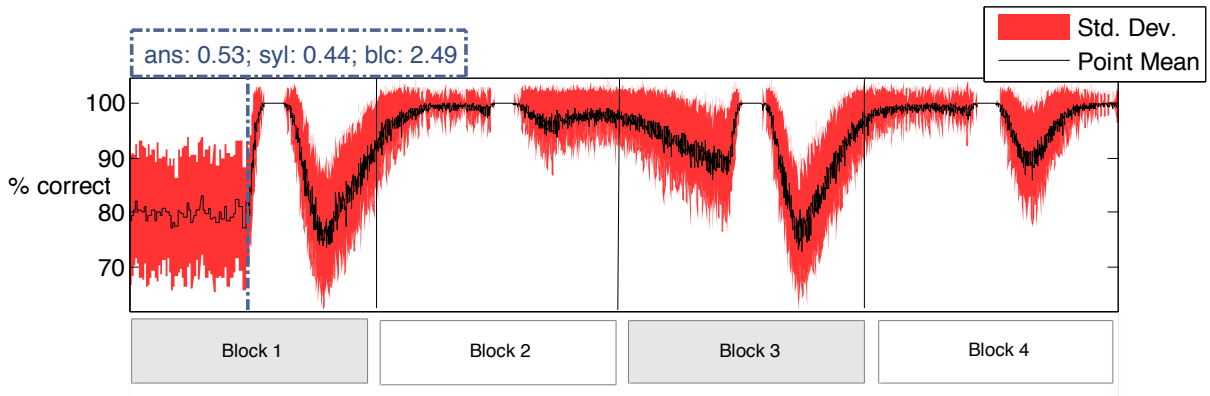


Figure 8. An example of how one can use ACT-R/Φ to understand the effects of dynamic physio-cognitive states on behavior and performance. The black line is the mean task performance, given the physio-cognitive state of the model at that point in time (e.g., roughly half-way through the block 1, the ACT-R/Φ model showed a declarative memory noise value of 0.53, which would result in it performing nearly 80% of the serial subtractions correct). The red area above and below the black line represents the standard deviation of that model's performance. For more explanation of this model, simulation, and figure, see Dancy et al. (2015).

[This figure will be supplied separately to maintain quality]

Dancy, Ritter, and Gunzelmann (2015) compare using ACT-R/Φ with a version of ACT-R combined with a more specific mathematical model of sleep fatigue to simulate the effects of sleep deprivation on behavior during the psychomotor vigilance task. Dancy and Kim (2018) further expand upon that work to simulate the effects of slow-breathing on physiology and cognition. Both use bottom-up physiological changes to cause changes to cognitive processes, and thus behavior in the models (albeit through different initial perturbations of physiological systems).

These physio-cognitive interactions also have implications for how we may treat affective or emotional behavior (e.g., Dancy, 2013; Dancy & Ritter, 2017b; Larue et al., 2018). For example, the physiological representations in ACT-R/Φ have allowed the development of a tractable physio-cognitive process model that simulates the effects of physiological changes (e.g., hunger or thirst) on choice (Dancy & Kaulakis, 2013). In addition, the architecture has been used to explore other interactions between affective and cognitive processes, and their effects on decision-making (Dancy & Schwartz, 2017).

Though this architecture still has many possible connections between physiological, affective, and cognitive systems that can be implemented, it represents a useful start to integrating more robust and encompassing models to develop process models and simulations of physio-affective-cognitive interactions. The architecture allows for more tractable representations in process models of human behavior and initial exploration of scenarios and physio-cognitive states that may otherwise be too dangerous or unethical to explore experimentally.

Summary: Modeling Moderators

There are several ways one may choose to model and simulate the effects of moderators on behavior. We have given some examples of ways that architectures have been used in concert with more focused models to simulate particular moderators using overlays and ways an architecture can be used with a more general model of physiology to simulate how the same moderators influence cognition over time (ACT-R/Φ), but in ways that are perhaps more generalizable. Both ways have their own advantages and disadvantages, and their use will likely depend upon the use-case (e.g., do you need to use a lighter

model that will use less computational time and memory, or do you want a model that can represent more moderators simultaneously).

Both approaches will likely require a continued move towards more instances of experiments that collect more diverse datasets (e.g., including more physiology variables with cognitive performance), but this is particularly true for using systems like ACT-R/Φ. Though verification of mechanisms can occur by using existing data from separate sources, ACT-R/Φ becomes a more powerful architecture and the modeling more straightforward when there are both physiological and behavioral data. These data may allow one to verify that models and mechanisms mediating models are behaving reasonably. If these data do not exist, ACT-R/Φ and related systems can tell us what data may be useful to collect in the future and may point to new areas of inquiry that relate to these complex and dynamic physio-affective-cognitive interactions. Such systems can give us encouragement to look beyond what we can see underneath the proverbial street-lamp.

While the architecture does well to model and simulate some physio-cognitive behaviors, ACT-R/Φ will need to be expanded upon in the future to represent more moderators. This will likely have to happen both in the connections between ACT-R and HumMod and within the HumMod model itself. Good examples of this evolution are represented in observing the expansion in the physiological model (HumMod) and connections seen between the initial work on simulating stress effects on declarative memory (Dancy et al., 2012) and more recent work on understanding how physio-cognitive changes from behavior like slow-breathing may mediate behavior (Dancy & Kim, 2018; Kim et al., 2018). We will also have to include more cognitive task appraisals such as those EMA used (Gratch & Marsella, 2004).

Understanding the Fit Between Model and Data Better

The final topic is better understanding models' fit to data. Fitting a cognitive model to data is the same task as fitting a stochastic multivariable non-linear multi-value optimization function. The function is also often stochastic because models typically include noise in their processes, so they must be run multiple times to obtain a stable prediction and to understand the model. The function to be fit is multivariable because there are multiple parameters and model aspects to modify to adjust the fit. The fit is non-linear in that how the fit varies with parameters is not linear on most parameters and definitely not linear with their interactions. The task is multi-value because there are multiple aspects of data to be fit, including, for example, timing and errors. In addition, some parameters are also binary or categorical parameters, which further complicates the fitting process. Thus, most cognitive model fitting is more complex than a logistic regression (Pampel, 2000), and therefore, is not a problem to be solved using manual optimization techniques or even hill climbing (Kase, 2008). Even exploring the functions that the models represent are best done with high-performance computing (Moore, 2011).

Due to the complexity of this process, we argue that an automated approach should perform model fitting. We start this section by noting the effects of stochasticity on model understanding, and then we present an example showing how automatic model fitting can be done.

Running Models until They Provide Stable Predictions

A question we addressed on the way to improving the fit of our model was accessing the fits. With a stochastic model, when you have (or think you have) a set of parameters that seem reasonable, you are left with the question of how many times to run a model to obtain stable performance? Ritter et al. (2011) examined this question using a model of subtraction in ACT-R.

Figure 9 shows that as the number of model runs increases for this model of subtraction the mean adjusts to a more stable value. At Trial 15 and on for this model the mean is relatively stable. At the same time, the standard deviation appears to stabilize.

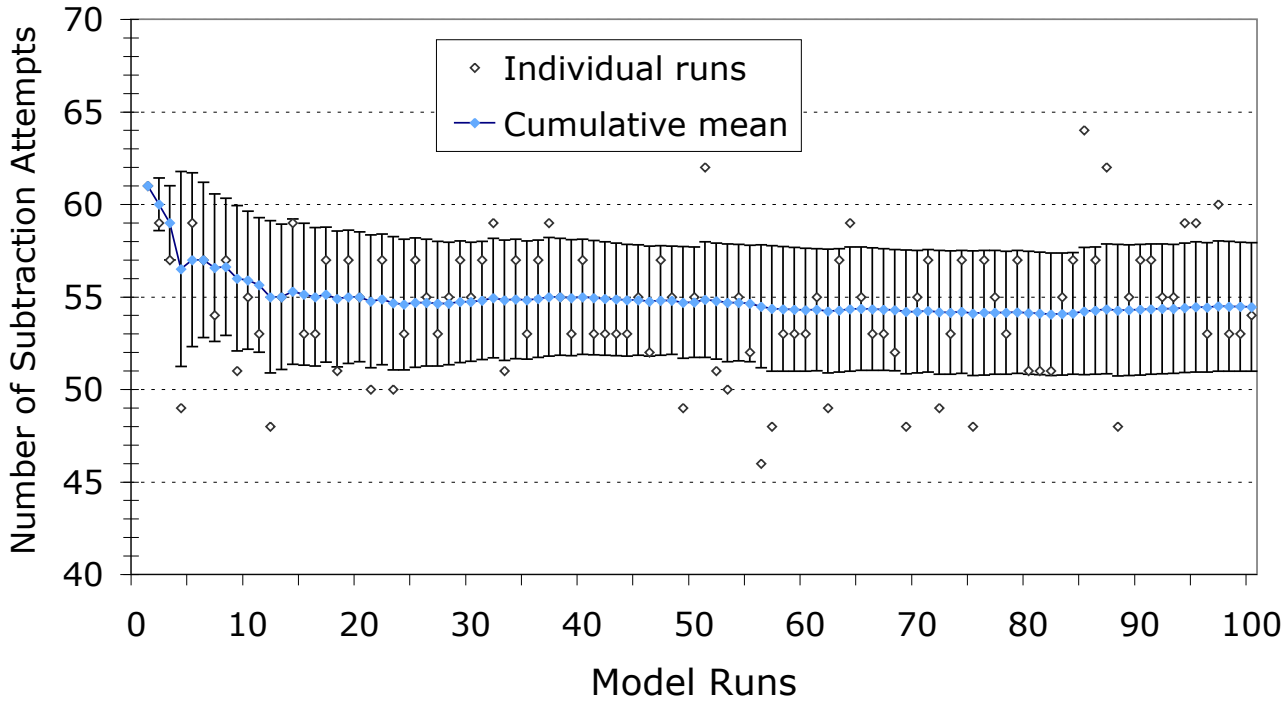


Figure 9. How the mean number of subtraction attempts varies with additional runs in a model of subtraction.

Figure 10 shows how the standard deviation and the standard error of the mean vary with additional runs. The changes in standard deviations are more visible in this graph. For this model, the standard deviation continues to stabilize over 100 runs. A standard error of the mean can be computed at Trial 3 and basically continually drops from a few runs after that. As Figures 9 and 10 show, this model's performance is noisy, that multiple runs can help understand its behavior, and human or simple optimization algorithms will have difficulty fitting it to data (Kase, 2008). Ritter et al. (2011) conclude that models should not be run a set number of times, but run until the models are understood (although they do provide heuristics for the number of runs).

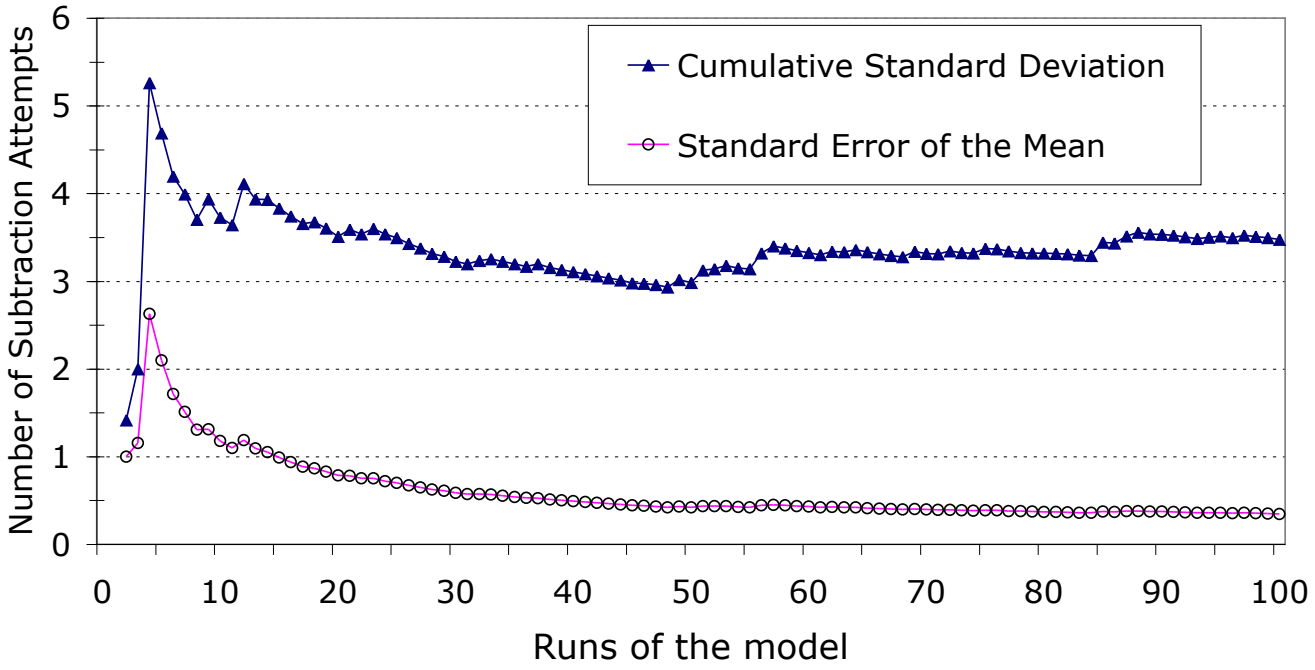


Figure 10. How the number of subtraction attempts and the cumulative standard deviation and for the standard error of the mean (SEM) varies with additional runs.

Fitting a Complex Model to Stressed and Unstressed Users

One automated approach that has been used for model fitting for complex functions is Genetic Algorithms (GAs), a type of machine learning. GAs are a subset of evolutionary algorithms that are broadly classified as meta-heuristic search algorithms imitating aspects of biological evolution. GAs represent an intelligent exploitation of a random search used to solve optimization problems by applying bio-inspired operators such as mutation, crossover, and selection to generations of genotypes. GAs are robust search algorithms, unlike conventional AI systems that can break if the inputs change slightly or in the presence of noise. Both these characteristics are advantageous for fitting computational cognitive models' performance to human data.

Early work on this approach was done as class projects, and BS and MS theses. It showed that the approach of using a GA to fit a model to data had promise (Cornwell, 2001; Ritter, 1990, 1991; Tor & Ritter, 2004), but did not use a large dataset. Further work continued to show that GAs could fit larger data sets (Lane & Gobet, 2005; Peebles, 2016). As an example of applying a GA to automate the model fitting process, this section overviews an investigation on how cognition changes with stress and caffeine while performing a serial subtraction task (Kase, Ritter, Bennett, Klein, & Schoelles, 2017). We start by describing the task and data, and then explain how a model was fit by modifying architectural parameters to explain what changed under stress.

The Serial Subtraction Task

Human performance data from a serial subtraction task was collected after one of three doses of caffeine (placebo, 200 mg, 400 mg) was administered to 45 male participants (Klein et al., 2008). The serial subtraction task is part of the Trier Social Stressor Task (TSST) that has been often used in physiology studies to cause stress in laboratory environments (Kirschbaum, Pirke, & Hellhammer,

1993). The task requires participants to mentally subtract 7s and 13s from 4-digit starting numbers without paper or visual cues. Participants verbally speak the answers of each subtraction problem to a laboratory assistant holding a timekeeping device and answer sheet. Figure 11 shows a portion of the serial subtraction task with example starting numbers for the four blocks (two blocks subtracting by 7s; two blocks subtracting by 13s). Participants are informed of incorrect answers and then directed to start over at the previous correct answer. Participants are instructed to go faster halfway into each four-minute block. Performance was recorded as a number of attempted subtraction problems and the percent correct for each participant in each block.

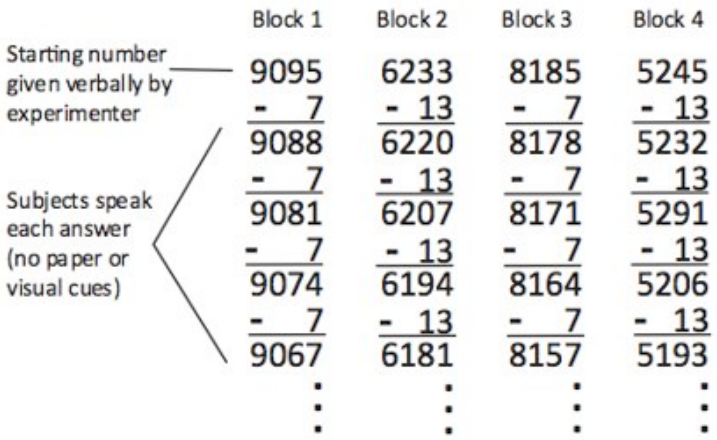


Figure 11. An example of the serial subtraction task stimuli for each of four blocks in a Trier Social Stressor Task.

Human Performance Analysis

Before and after the serial subtraction session, participants completed pre- and post-task appraisals based on Lazarus and Folkman's (1984) theory of stress and coping. Each participant answered questions reporting their perceived resources to deal with the serial subtraction task and how stressful they thought the task would be to perform. A ratio of perceived task requirements to perceived coping resources was created from the appraisal survey results. If perceived task requirements were less than or equal to perceived task resources (ability to cope), this equated to a *challenge condition*. If perceived needs was greater than the perceived resources, this equated to a *threat condition*.

Performance differences between the challenge and threat conditions were most pronounced in the 200 mg (LoCAF) group with an increase of 20 more attempted subtraction problems per 4-minute block and a 13.5% increase in subtraction accuracy by challenge participants over threat participants. For the 400 mg (HiCAF) group the challenge and threat conditions differences were less than LoCAF but still substantial: 13 more attempted problems and a 7.7% increase in subtraction accuracy in the challenge condition over the threat condition. Differences between the challenge and threat condition were least visible in the placebo (PLAC) group, 10 more attempted problems and only a 5.4% increase in accuracy.

Figure 12 visualizes these performance differences with the caffeine treatment groups (PLAC, LoCAF, HiCAF) labeled along the x-axis and the plot subdivided into three sections: averages across treatment groups (not by appraisal condition) in the leftmost section, and averages across treatment groups subdivided by appraisal condition in the center (challenge) and rightmost sections (threat). The plotted data should be viewed from the perspective of a pattern that visualizes potential trends, not as a test of statistical significance. It does match the expected effect of caffeine having an inverted U-shaped curve for those in a challenge condition, and a relatively smaller effect on those that appraised the condition as threat.

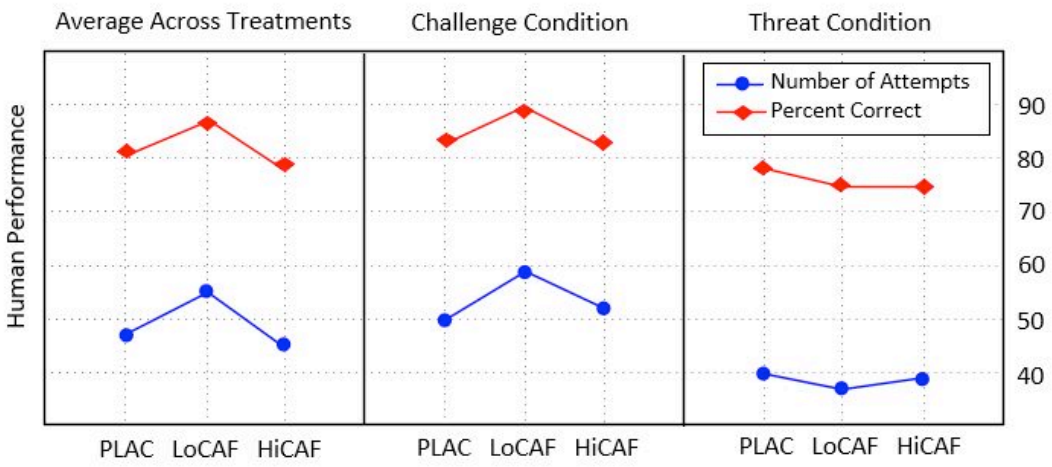


Figure 12. Comparing human performance differences in number of attempts and percent correct by treatment group (x-axis) and appraisal condition: treatment groups not accounting for appraisal (leftmost section), and averages across treatment groups divided by appraisal condition, challenge (middle section) and threat (rightmost section).

Optimizing the Fit of a Serial Subtraction Model

Based on the analysis of the human performance results, a simple cognitive model of the serial subtraction task was developed using the ACT-R cognitive architecture (Anderson, 2007; Anderson et al., 2004) to provide a description of how the task is performed. During model development, psychological theories of mental arithmetic performance were combined with observations gleaned during the experiment. The serial subtraction model performs a block of subtracting by 7s or 13s similarly to that of the human participants using a column-by-column calculation and verbalization strategy (Ritter, Reifers, et al., 2007; Ritter, Schoelles, Klein, & Kase, 2007). The model's declarative knowledge uses approximately 650 arithmetic facts and goal-related information to solve the subtractions. ACT-R's vocal module enables the model to verbalize the answers to each subtraction problem, similar to what the participants do.

Several architectural parameters in ACT-R appeared important in performing serial subtraction. Three parameters were selected to explore out of more than 80 parameters available in the ACT-R architecture: the activation noise (ANS) parameter that affects variance in retrieving declarative memory information and the error rate for retrievals; the seconds-per-syllable (SYL) parameter that controls the rate the model speaks; and the base level constant (BLC) parameter that affects declarative knowledge access. The search space for fitting the serial subtraction model was defined by the three parameter value boundaries: ANS [0.1 – 0.9], SYL [0.1 – 0.9], and BLC [0.1 – 3.0].

Because of the large search space, a GA approach was used to automate the serial subtraction model fitting process. GAs have been used to optimize the fit of functions to data for a long time (Davis & Ritter, 1987; Goldberg, 1989). GAs use a set of genotypes, the genes for a solution, and compute the fitness of a phenotype, a solution created by the genotype. In cognitive modeling, a genotype could represent a set of parameters applied to the cognitive model. For this investigation, a parallel genetic algorithm (PGA) was used to overcome the large combinatory parameter search spaces and substantial computational and time resources associated with fitting the ACT-R model to the human performance on the serial subtraction task. We could imagine using other approaches with other models and tasks.

The ACT-R cognitive architecture and serial subtraction model are written in the Lisp programming language. To utilize the parallel processing of the PGA in the fitting process, ACT-R and the model were packaged into an executable Lisp image or core file for use in a high-performance computing (HPC) cluster. This image file can be executed by a system call from a C program on each processor in

parallel while using a message passing interface (MPI) to communicate genotypes and fitness values among the processors.

To run a generation of the models, the population of genotypes (ACT-R parameter sets consisting of ANS, BLC, and SYL values) is distributed to the processors using the MPI. Each processor executes the Lisp image file that runs the model within the ACT-R architecture and then calculates a fitness value based on the model's performance predictions compared to the human performance statistics (number of attempts and percent correct). To produce a fitness value, a sum of squares error function uses both statistics from each block of serial subtraction performance. Based on the fitness values, genetic functions (mutation, crossover, and selection) are applied to the corresponding 'population' of genotypes. This process of generating and evaluating genotypes is repeated through a number of generations with the effect of evolving a set of candidate solutions (the best parameter values that fit the model).

For this investigation, the optimization setup consisted of nine PGAs executing 100 generations of 200 binary-encoded genotypes. Each PGA optimized the serial subtraction model to participant group performance data by caffeine and appraisal conditions. The PGAs used genotypes formed by one 36-bit chromosome divided into three 12-bit substrings each representing the value of the three ACT-R parameters ANS, BLC, and SYL. The termination condition for the PGAs was a specified number of generations (100) instead of proximity to each participant's performance statistics. These choices are somewhat arbitrary but show an example GA.

Model-to-data fit was determined by the fitness function defined by the sum of squared discrepancies between model performance (number of attempts and percent correct) and the corresponding human performance (e.g., $(47.3 - 47.2)^2 + (81.5\% - 81.4\%)^2 = 0.02$). In this case, the fitness represents a difference between model performance and the data. A fitness value of 0 representing perfect correspondence between the model predictions and the human data; values less than 1.0 represent a fit less than 1 difference in a number of attempts and percent correct. Other functions could have been used and would have perhaps found slightly different results. One can also weight the fit to improve response time or error rate or error type or other measures.

Employing this type of automated optimization approach allowed for 20,000 different sets of parameter values to be tested in a directed manner each time the PGA was executed. Using this approach, the serial subtraction model was optimized to the nine rows of human performance data shown in Table 4.

Results of the PGA Optimization

Table 4 summarizes the PGA optimization results by caffeine level and appraisal. The first column denotes the appraisal condition with CH for the challenge, TH for threat, and ALL for the average across challenge and threat. The next two columns, Human Performance and Avg. Model Prediction, list the number of attempts (first value) and percent correct (second value) for the human (second column) and the model (third column). The model's performance is an average across the number of best fitting parameter sets. For example, the (3) in the first row, last column, means the PGA found three parameter sets producing fitness less than 1.0 and that the parameter values (last column), model predictions (third column), and fitness value (fourth column) are averaged across those three best PGA runs. The fitness value column shows the PGA optimizations were able to find good solutions for all the caffeine and appraisal conditions within a fractional part of a single subtraction problem. Considering the complexity of the serial subtraction task and how the human performance varied across conditions, these are exceptional model-to-human data fits that suggest how cognition changed.

Table 4. Optimization results for three treatment groups (PLAC, LoCAF, HiCAF) and appraisal groups (CH = challenge, TH = threat) comparing human performance and model predictions in number of attempts and percent correct in a four minute block, and average fitness value associated with the best fitting (less than 1.0 fitness values, shown in parentheses).[3]

Condition	Human Performance	Avg. Model Prediction	Avg. Fitness Value	ACT-R Parameters ANS, BLC, SYL (N)
PLAC (no caffeine)				
ALL	47.3 / 81.5%	47.2 / 81.4%	0.53	0.64, 2.45, 0.51 (3)
CH	50.7 / 83.3%	50.8 / 83.2%	0.47	0.62, 2.44, 0.52 (6)
TH	40.4 / 77.9%	40.2 / 77.9%	0.36	0.68, 2.47, 0.56 (5)
LoCAF (200 mg caffeine)				
ALL	54.9 / 86.5%	54.9 / 85.9%	0.52	0.61, 2.45, 0.40 (4)
CH	57.8 / 88.3%	57.7 / 87.7%	0.54	0.61, 2.47, 0.36 (3)
TH	37.5 / 74.8%	37.7 / 74.9%	0.38	0.59, 2.34, 0.60 (6)
HiCAF (400 mg caffeine)				
ALL	45.7 / 79.2%	45.7 / 79.2%	0.54	0.58, 2.38, 0.52 (4)
CH	51.6 / 82.8%	51.7 / 82.7%	0.51	0.52, 2.30, 0.43 (3)
TH	38.9 / 75.1%	38.9 / 75.1%	0.57	0.65, 2.46, 0.62 (4)

Several trends can be observed within the parameter values producing best fits in Table 4. Beginning with the seconds-per-syllable parameter, SYL, shown in the last column and last value in the triple in Table 4, the model predictions indicate that overall challenge subjects speak a syllable more quickly than threat subjects. This is true for all treatment groups. LoCAF shows the greatest difference in speech rate (0.24) while HiCAF differences in SYL are less (0.19), and PLAC differences are the least (0.04). Challenge subjects self-report less stress and are generally confident that they can perform the serial subtraction task well. With less stress and a low dose of caffeine more fluid speech appears to occur.

Overall, the activation noise parameter values (ANS, the first value in triple) are high compared to what would be manually assigned to the model in the ACT-R modeling community. This occurrence could be due to the task being more stressful than typically found in psychology experiments (i.e., this task as part of the Trier Social Stressor Task has been purposively used to elicit a stress response). Although abnormally high, the ANS value range in Table 4 is narrow (a difference of 0.16), which hints at the fact that caffeine may not greatly effect this parameter's role in the model's performance of serial subtraction. The ANS values are basically equivalent across PLAC (a difference of 0.06) and LoCAF (a difference of 0.02). The greatest variability in ANS (a difference of 0.13) is in HiCAF where challenge predictions (0.52) show a lower value then threat predictions (0.65). ANS values that are slightly higher in predicting threat subjects correspond to lower performance (fewer attempts and lower accuracy), and the self-reports where subjects do not believe they will perform well.

Similar to the ANS value range, the base level constant parameter (BLC, the middle value in the triple) value range in Table 4 is narrow (a difference of 0.17). Caffeine may not effect this parameter's role in the model's performance of serial subtraction either. The BLC values are basically equivalent across PLAC (a difference of 0.03). LoCAF shows a difference in BLC values of 0.13 with challenge predictions showing a higher BLC value than threat predictions. In this case, caffeine may be causing a 'boost' in the base level activation value of facts in declarative memory promoting the higher probability of selection in response to a retrieval request and lower declarative fact retrieval time which would lead to better overall performance. Surprisingly HiCAF exhibits a reverse trend with challenge

[3] This table represents a correction to similar tables published in Kase et al. (2017) and Ritter, Kase, Klein, Bennett, and Schoelles (2009) in the human and model performance columns but not the fitness or parameters columns, which remain basically the same; the implications also remain the same.

predictions showing a lower BLC value (2.30) compared to threat predictions (2.46). This trend is unexpected, requiring further investigation.

Summary: Optimizing Fit of Models Automatically

This investigation showed how a cognitive model was fit to three different caffeine treatments and appraisal groups using a PGA. The fits were very close, revealing several patterns in the parameter values. The changes to the model to fit these data increased understanding of the cognitive mechanisms in the ACT-R architecture that lead to differences in behavior. There appear to be systematic changes in cognition due to caffeine and task appraisal—most notable talking faster when challenged and with the low caffeine level and talking slower when threatened or with no or high caffeine, and more noise and higher basic activation in declarative memory processes when threatened with high caffeine.

Finally, this line of research has not been actively pursued by the cognitive modeling community but should be as these results point out. The PGA optimization approach was successful in producing excellent model-to-human data fits and shows promise for replacing the cognitive modeling community's traditional manual optimization technique—an iterative step-by-step process that encourages modeler bias in selecting parameter values that at best support a chosen hypothesis, and at worst lead to lack of understanding of a complex process. Overfitting can be ameliorated by running this stochastic approach several times to see if the same parameters sets are consistently found or if there are multiple good fits, gathering additional data, and testing the results to see if the parameter sets continue to provide good fits and examining the range of behavior that they lead to.

Conclusions and Future Work

This paper reviewed some promising futures for cognitive modeling and architectures. These were providing simulated eyes and hands for models to interact directly with unmodified interfaces, providing a physiology substrate to the mind to support modeling how cognition is changed by having a body, and using genetic algorithms to fit models to data to understand the model better. Work that has been done show that work in these directions can be fruitful, but more importantly, in each case in the work shows that much further work can be done.

Including these approaches in modeling cognition will make it harder in many cases (but not all, the use of simulated eyes and hands will mean that additional interfaces for models will not have to be built), but will make the models more representative of how humans behave in the world and are modified by it. You might wish to pursue these directions or work to make them easier to include.

If these directions are viewed as design patterns, and general approaches to performing a task or extending an architecture, they suggest that there are many other projects in these areas that you can approach too. In addition to these results, we would like to close by noting where you can learn more, and to provide some comments on modeling in general.

Further Resources on Modeling

There are a couple of resources to mention to provide further modeling design patterns. Pew has led several reviews (Gluck & Pew, 2005; Pew, 2007; Pew & Mavor, 1998, 2007) that provide example models, architectures, and uses. A response to Pew and Mavor's (1998) report also has further example projects (Ritter et al., 2003). To learn more about the basics of behavior for modeling, we can recommend several textbooks for psychology (Anderson, 2000, 2014, any editions; Ritter, Baxter, & Churchill, 2014) and for physiology (Hall, 2016, also known as Guyton and Hall, any edition).

Comments on How to Model

This paper concludes with some comments on modeling in general based on these projects. These projects often reused data and models (and updated the models while doing so). Each of these projects reused multiple programs. It is probably the case that reuse was essential because work in this area is often complex. So it is worth explicitly noting that it is helpful to reuse models, data, and tools because it is often very difficult to create them.

So, if you are going to reuse materials, we encourage you to document your systems and code in several ways. This is easier when the system has been documented with comments and README files and stored separately from the investigator in an archive. It was also useful to have graphs, analyses, pictures, and movies of the models running. We recommend creating these documentation items as you iterate your model, and to note when and how they were done, the software used, the processes used, who created or performed each one, and what each column or variable means.

Newell also often mentioned in discussions that you should not build tools for others—you should build tools for yourself and use them. If your use is promising, then others will help build a community. In this case, you are the user, and your use is the proof. We have partially fulfilled Newell's vision in this area with these projects.

These projects and their results, those of interaction, modeling the physiology underlying cognition, and machine learning optimized fits are presented as design patterns. You do not have to use the details, but recreate these types of processes with your own resources, and for your own architecture, and for your own situation—so can you!

References

[NOTES to COPYEDITOR/TYPESETTER:
1. Colbert: The publisher is: Grand Central Publishing Hachette Book Group
2. St. Amant: *intelligence* [no caps] is the name of the journal
2. Grossman: *on Killing* [no initial cap] is the name of the Grossman book]

Anderson, J. R. (2000). *Learning and memory: An integrated approach*. New York, NY: John Wiley and Sons.
Anderson, J. R. (2007). *How can the human mind exist in the physical universe?* New York, NY: Oxford University Press.
Anderson, J. R. (2014). *Cognitive psychology and its implications* (8th ed.). New York, NY: Worth Publishers.
Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review, 111*(4), 1036-1060.
Bass, E. J., Baxter, G. D., & Ritter, F. E. (1995). Creating models to control simulations: A generic approach. *AI and Simulation of Behaviour Quarterly, 93*, 18-25.
Brener, M., Becera, N., Pruett, D., Ritter, F. E., Dancy, C. L., & Webster, I. (2019). *Manual for HumMod (Salt Version 3.0.4)* (Tech. Report No. ACS 2019-1, Version: 2.9). Applied Cognitive Science Lab, College of Information Sciences and Technology, Penn State. acs.ist.psu.edu/reports/brenerBBPRDW19.pdf.
Busetta, P., Howden, N., Rönnquist, R., & Hodgson, A. (1999). Structuring BDI agents in functional clusters. In *Proceedings of the Sixth International Workshop on Agent Technologies, Architectures and Languages*, 149-161.
Busetta, P., Rönnquist, R., Hodgson, A., & Lucas, A. (1999). JACK intelligent agents—Components for intelligent agents in JAVA. *AgentLink News Letter, 2*(Jan.), www.agent-software.com/white-paper.pdf.
Byrne, M. D., & Anderson, J. R. (1998). Perception and action. In J. R. Anderson & C. Lebiere (Eds.), *The atomic components of thought*. Mahwah, NJ: Erlbaum.
Colbert, S. (2007). *I am America (and so can you!)* New York, NY: Grand Central Publishing Hachette Book Group.
Cornwell, J. B. (2001). *Using genetic algorithms to create and optimize cognitive models of development*. Unpublished BS honors thesis, The Pennsylvania State University, University Park.

Dancy, C. L. (2013). ACT-RΦ: A cognitive architecture with physiology and affect. *Biologically Inspired Cognitive Architectures, 6*, 40–45

Dancy, C. L. (2019). A hybrid cognitive architecture with primal affect and physiology. *IEEE Transactions on Affective Computing*.

Dancy, C. L., & Kaulakis, R. (2013). Towards adding bottom-up homeostatic affect to ACT-R. In *Proceedings of the 12th International Conference on Cognitive Modeling*, 316-321. Ottawa, Canada.

Dancy, C. L., & Kim, J. W. (2018). Towards a physio-cognitive model of slow-breathing. In *Proceedings of the 40th Annual Conference of the Cognitive Science Society*, 1587-1592. Madison, WI.

Dancy, C. L., & Ritter, F. E. (2017a). IGT-Open: An open source, computerized version of the Iowa Gambling Task. *Behavioral Research Methods, 49*(3), 972-978.

Dancy, C. L., & Ritter, F. E. (2017b). A standard model of the mind needs a body. In *Proceedings of the AAAI Fall Symposium Series*, 316-320. Arlington, VA.

Dancy, C. L., Ritter, F. E., & Berry, K. (2012). Towards adding a physiological substrate to ACT-R. In *Proceedings of the 21st Conference on Behavior Representation in Modeling and Simulation*, 12-BRIMS-014, 078-085. Amelia Island, FL: BRIMS Society.

Dancy, C. L., Ritter, F. E., Berry, K., & Klein, L. C. (2015). Using a cognitive architecture with a physiological substrate to represent effects of psychological stress on cognition. *Computational and Mathematical Organization Theory, 21*(1), 90-114.

Dancy, C. L., Ritter, F. E., & Gunzelmann, G. (2015). Two ways to model the effects of sleep fatigue on cognition. In *Proceedings of the 13th International Conference on Cognitive Modeling 2015*, 258-263.

Dancy, C. L., & Schwartz, D. M. (2017). A computational cognitive-affective model of decision-making. In *Proceedings of the 15th International Conference on Cognitive Modeling*, 31-36. Coventry, United Kingdom: University of Warwick.

Davis, L. W., & Ritter, F. (1987). Schedule optimization with probabilistic search. In *The Third IEEE Computer Society Conference on Artificial Intelligence Applications*, 231-236. Washington, DC: IEEE Press.

Evertsz, R., Pedrotti, M., Busetta, P., Acar, H., & Ritter, F. E. (2009). Populating VBS2 with realistic virtual actors. In *Proceedings of the 18th Conference on Behavior Representation in Modeling and Simulation*, 09-BRIMS-04.

Findlay, J. M., & Gilchrist, I. D. (2003). *Active Vision: The psychology of looking and seeing*. Oxford, UK: Oxford University Press.

Gemma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns: Elements of reusable object-oriented software*. Boston, MA: Addison-Wesley.

Gluck, K. A., & Pew, R. W. (Eds.). (2005). *Modeling human behavior with integrated cognitive architectures: Comparison, evaluation, and validation*. Mahwah, NJ: Erlbaum.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Reading, MA: Addison-Wesley.

Gratch, J., & Marsella, S. (2004). A domain-independent framework for modeling emotion. *Cognitive Systems Research, 5*(4), 269-306.

Gray, W. D. (2000). The nature and processing of errors in interactive behavior. *Cognitive Science, 24*(2), 205-248.

Gray, W. D. (2002). Simulated task environments: The role of high-fidelity simulations, scaled worlds, synthetic environments, and microworlds in basic and applied cognitive research. *Cognitive Science Quarterly, 2*(2), 205-227.

Gray, W. D. (2007). Composition and control of integrated cognitive systems In W. D. Gray (Ed.), *Integrated models of cognitive systems* (pp. 3-12). New York, NY: Oxford University Press.

Gray, W. D. (2008). Cognitive architectures: Choreographing the dance of mental operations with the task environment. *Human Factors, 50*(3), 497-505.

Grossman, D. (1996). *on killing: The psychological cost of learning to kill in war and society*. New York, NY: Back Bay Books, Little Brown and Company.

Hall, J. E. (2016). *Guyton and Hall Textbook of medical physiology* (13th ed.). Philadelphia, PA: Elsevier.

Hester, R. L., Brown, A. J., Husband, L., Iliescu, R., Pruett, D., Summers, R., et al. (2011). HumMod: A modeling environment for the simulation of integrative human physiology. *Frontiers in Physiology, 2*(12), Article 12.

Hope, R. M., Schoelles, M. J., & Gray, W. D. (2014). Simplifying the interaction between cognitive models and task environments with the JSON Network Interface. *Behavior Research Methods, 46*(4), 1007-1012.

Jones, G., Ritter, F. E., & Wood, D. J. (2000). Using a cognitive architecture to examine what develops. *Psychological Science, 11*(2), 93-100.

Kase, S. E. (2008). *Parallel genetic algorithm optimization of a cognitive model: Investigating group and individual performance on a math stressor task*. Unpublished PhD thesis, College of IST, Penn State University, University Park, PA.

Kase, S. E., Ritter, F. E., Bennett, J. M., Klein, L. C., & Schoelles, M. (2017). Fitting a model to behavior reveals what changes cognitively when under stress and with caffeine. *Biologically Inspired Cognitive Architectures, 22*(October), 1-9.

Kim, J. W., Dancy, C. L., & Sottilare, R. A. (2018). Towards using a physio-cognitive model in tutoring for psychomotor tasks. In *Proceedings of the AIED Workshop on Authoring and Tutoring for Psychomotor, Mobile, and Medical Domains*. London, UK.

Kim, J. W., & Ritter, F. E. (2015). Learning, forgetting, and relearning for keystroke- and mouse-driven tasks: Relearning is important. *Human-Computer Interaction, 30*(1), 1-33.

Kirschbaum, C., Pirke, K.-M., & Hellhammer, D. H. (1993). The Trier Social Stress Test—A tool for investigating psychobiological stress responses in a laboratory setting. *Neuropsychobiology, 28*, 76-81.

Laird, J. E. (2012). *The Soar cognitive architecture*. Cambridge, MA: MIT Press.

Lane, P. C. R., & Gobet, F. (2005). Discovering predictive variables when evolving cognitive models. In *Pattern Recognition and Data Mining, Third International Conference on Advances in Pattern Recognition, ICAPR 2005, Bath, UK, August 22-25, 2005, Proceedings, Part I*, 108-117. Heidelberg, Germany: Springer.

Larue, O., West, R. L., Rosenbloom, P. S., Dancy, C. L., Samsonovich, A. V., Petters, D., et al. (2018). Emotion in the Common Model of Cognition. *Procedia Computer Science, 145*, 740-746.

Lazarus, R. S., & Folkman, S. (1984). *Stress, appraisal and coping*. New York, NY: Springer Publishing.

Moore, L. R., Jr. (2011). Cognitive model exploration and optimization: A new challenge for computational science. *Computational and Mathematical Organization Theory, 17*, 296–313.

Morgan, G. P., Ritter, F. E., Stine, M. M., & Klein, L. C. (2006). The cognitive effects of caffeine: Implications for models of users: ACS Lab, College of IST, Penn State. Unpublished manuscript.

Morgan, J. H., Morgan, G., & Ritter, F. E. (2010). A preliminary model of participation for small groups. *Computational and Mathematical Organization Theory, 16*, 246-270.

Morgan, J. H., Morgan, G. P., Ritter, F. E., & Poncelin de Raucourt, V. (2009). A preliminary model of participation. In *Proceedings of the 18th Conference on Behavior Representation in Modeling and Simulation*, 129-136. 109-BRIMS-127.

Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA: Harvard University Press.

Norling, E., & Ritter, F. E. (2004). *CoJACK Software Specification*. Human Variability within Computer Generated Forces Project, Project No: RT/COM/3/006: Agent Oriented Software Limited.

Ong, R. (1994). *Mechanisms for routinely tying cognitive models to interactive simulations*. Unpublished MSc thesis, U. of Nottingham. Available as ESRC Centre for Research in Development, Instruction and Training Technical report #21 and as http://acs.ist.psu.edu/misc/nottingham/ccc/mongsu-2.1.tar.gz.

Paik, J., Kim, J. W., Ritter, F. E., & Reitter, D. (2015). Predicting user performance and learning in human-computer interaction with the Herbal compiler *ACM Transactions on Computer-Human Interaction, 22*(5), Article 25.

Pampel, F. C. (2000). *Logistic regression: A primer*. Thousand Oaks, CA: Sage.

Peebles, D. (2016). Two methods for optimising cognitive model parameters. In *Presentations at the ACT-R Post-graduate Summer School*.

Pew, R. W. (2007). Some history of human performance modeling. In W. Gray (Ed.), *Integrated models of cognitive systems* (pp. 29-44). New York, NY: Oxford University Press.

Pew, R. W., & Mavor, A. S. (Eds.). (1998). *Modeling human and organizational behavior: Application to military simulations*. Washington, DC: National Academies Press.

Pew, R. W., & Mavor, A. S. (Eds.). (2007). *Human-system integration in the system development process: A new look*. Washington, DC: National Academy Press. books.nap.edu/catalog/11893, checked May 2019.

Ritter, F. E. (1990). Mendel-DP: Optimizing PDP learning rates. In *Eighth Annual Pitt-CMU conference, June and the CMU PDP seminar, June*.

Ritter, F. E. (1991). Towards fair comparisons of connectionist algorithms through automatically generated parameter sets. In *Proceedings of the 13th Annual Conference of the Cognitive Science Society*, 877-881. Hillsdale, NJ: Erlbaum.

Ritter, F. E. (1993). Three types of emotional effects that will occur in cognitive architectures. In *Workshop on architectures underlying motivation and emotion (WAUME93)*. School of Computer Science and Centre for Research in Cognitive Science, University of Birmingham, UK.

Ritter, F. E. (2019). Modeling human cognitive behavior for system design. In P. Gunther, S. Nightingale & A. C. A. Garcia (Eds.), *Digital human modelling and posturography* (pp. 517-525). London: Academic Press.

Ritter, F. E., Baxter, G. D., & Churchill, E. F. (2014). *Foundations for designing user-centered systems: What system designers need to know about people*. London, UK: Springer.

Ritter, F. E., Baxter, G. D., Jones, G., & Young, R. M. (2000). Supporting cognitive models as users. *ACM Transactions on Computer-Human Interaction, 7*(2), 141-173.

Ritter, F. E., Baxter, G. D., Jones, G., & Young, R. M. (2001). User interface evaluation: How cognitive models can help. In J. Carroll (Ed.), *Human-computer interaction in the new millennium* (pp. 125-147). Reading, MA: Addison-Wesley.

Ritter, F. E., Bittner, J. L., Kase, S. E., Evertsz, R., Pedrotti, M., & Busetta, P. (2012). CoJACK: A high-level cognitive architecture with demonstrations of moderators, variability, and implications for situation awareness. *Biologically Inspired Cognitive Architectures, 1*(1), 2-13.

Ritter, F. E., Kukreja, U., & St. Amant, R. (2007). Including a model of visual processing with a cognitive architecture to model a simple teleoperation task. *Journal of Cognitive Engineering and Decision Making, 1*(2), 121-147.

Ritter, F. E., Reifers, A. L., Klein, L. C., & Schoelles, M. J. (2007). Lessons from defining theories of stress for architectures. In W. Gray (Ed.), *Integrated models of cognitive systems* (pp. 254-262). New York, NY: Oxford University Press.

Ritter, F. E., Schoelles, M. J., Klein, L. C., & Kase, S. E. (2007). Modeling the range of performance on the serial subtraction task. In *Proceedings of the 8th International Conference on Cognitive Modeling*, 299-304. Taylor & Francis/Psychology Press: Oxford, UK.

Ritter, F. E., Schoelles, M. J., Quigley, K. S., & Klein, L. C. (2011). 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.

Ritter, F. E., Shadbolt, N. R., Elliman, D., Young, R. M., Gobet, F., & Baxter, G. D. (2003). *Techniques for modeling human performance in synthetic environments: A supplementary review*. Wright-Patterson Air Force Base, OH: Human Systems Information Analysis Center (HSIAC).

Ritter, F. E., Tehranchi, F., & Oury, J. D. (2019). ACT-R: A cognitive architecture for modeling cognition. *Wiley Interdisciplinary Reviews: Cognitive Science, 10*(3), Paper e1488.

Ritter, F. E., Van Rooy, D., St. Amant, R., & Simpson, K. (2006). Providing user models direct access to interfaces: An exploratory study of a simple interface with implications for HRI and HCI. *IEEE Transactions on System, Man, and Cybernetics, Part A: Systems and Humans, 36*(3), 592-601.

Ritter, F. E., & Yeh, K.-C. (2011). Modeling pharmacokinetics and pharmacodynamics on a mobile device to help caffeine users. In *Augmented Cognition International Conference 2011, FAC 2011, HCII 2011, LNAI 6780*, 528-535. Berlin Heidelberg: Springer-Verlag.

Salt, L., Wise, J., Sennersten, C., & Lindley, C. A. (2016). REACT-R and Unity Integration. In *Proceedings on the International Conference on Artificial Intelligence (ICAI)*, 31.

Salvucci, D. D. (2006). Modeling driver behavior in a cognitive architecture. *Human Factors, 48*, 362-380.

Salvucci, D. D. (2009). Rapid prototyping and evaluation of in-vehicle interfaces. *ACM Transactions on Computer-Human Interaction, 16*(2), Article 9, 33 pages.

Salvucci, D. D. (2013). Integration and reuse in cognitive skill acquisition. *Cognitive Science, 37*(5), 829-860.

Shakir, A. (2002). Assessment of models of human decision-making for air combat analysis. [Unpublished technical report. DERA Farnborough. Abstract, put on the web with permission, at http://acs.ist.psu.edu/papers/shakir02-abstract.pdf].

St. Amant, R. (2000). Interface agents as surrogate users. *intelligence, 11*(2), 28-38.

St. Amant, R., Horton, T. E., & Ritter, F. E. (2007). Model-based evaluation of expert cell phone menu interaction. *ACM Transactions on Computer-Human Interaction, 14*(1), Paper 1. 24 pages.

St. Amant, R., & Riedl, M. O. (2001). A perception/action substrate for cognitive modeling in HCI. *International Journal of Human-Computer Studies, 55*(1), 15-39.

St. Amant, R., Riedl, M. O., Ritter, F. E., & Reifers, A. (2005). Image processing in cognitive models with SegMan. In *Proceedings of HCI International '05*, Volume 4 - Theories Models and Processes in HCI. Paper # 1869. Mahwah, NJ: Erlbaum.

Taatgen, N. A. (2002). A model of individual differences in skill acquisition in the Kanfer-Ackerman Air Traffic Control Task. *Cognitive Systems Research, 3*(1), 103-112.

Tambe, M., Johnson, W. L., Jones, R. M., Koss, F., Laird, J. E., Rosenbloom, P. S., et al. (1995). Intelligent agents for interactive simulation environments. *AI Magazine, 16*(1), 15-40.

Tehranchi, F., & Ritter, F. E. (2018). Modeling visual search in interactive graphic interfaces: Adding visual pattern matching algorithms to ACT-R. In *Proceedings of the 16th International Conference on Cognitive Modeling (ICCM 2018)*. 162-167. Madison, WI.

Tor, K., & Ritter, F. E. (2004). Using a genetic algorithm to optimize the fit of cognitive models. In *Proceedings of the Sixth International Conference on Cognitive Modeling*, 308-313. Mahwah, NJ: Erlbaum.

Wallach, D. P., Fackert, S., & Vladimir Albach, A. (2019). Predictive prototyping for real-world applications: A model-based evaluation approach based on the ACT-R cognitive architecture. In *DIS '19: Proceedings of the 2019 on Designing Interactive Systems Conference*, 1495–1502.

Author Bios

Frank Ritter is a professor of IST, CSE, and Psychology at Penn State. He has won several paper awards at the Behavior Representation in Modeling and Simulation (BRIMS) conference, co-chaired three BRIMS conferences, and edited three special issues of *Computational and Mathematical Organization Theory* (CMOT) based on the best BRIMS papers. He has been a Fulbright Scholar to TU/Chemnitz. He currently edits the Oxford series on cognitive models and architectures for Oxford University Press. His work has been funded by ARL, DARPA, DMSO, Dstl (UK), DSTO (Australia), DTRA, Harris, NSF, SSRC (UK), and ONR. He has published a textbook on what psychology systems designers need to know with the Director of User Experience at Google, which has repeatedly won awards at the HCI Consortium annual meeting.

Farnaz Tehranchi is a PhD student in Computer Science and Engineering at the Pennsylvania State University. She works in the Applied Cognitive Science Lab where she is advised by Dr. Frank E. Ritter and Dr. Rebecca Passonneau. Her research is focused on cognitive computing at the intersection of human-computer interaction and cognitive science. In her research, she is extending cognitive architectures to generate a cognitive model that can be used as a user model. She earned her MS and BS in Computer Science from Sharif University of Technology.

Christopher L. Dancy received a B.S. in Computer Science in 2010, and PhD in Information Sciences and Technology, with a focus on artificial intelligence and cognitive science in 2014, both from The Pennsylvania State University (University Park). He is an assistant professor of computer science at Bucknell University. His research involves the computational modeling of physiological, affective, and cognitive systems in humans. He studies how these systems interact and what these interactions mean for human-like intelligent behavior and interaction between humans and artificial intelligent systems. His work has been funded by NSF, ONR, ARL, and The Social Science Research Council. Chris Dancy has previously chaired the Behavior Representation in Modeling and Simulation Society and is currently a member of ACM, AAAI, the Cognitive Science Society, NSBE, IEEE SMC, and the IEEE Computer Society.

Sue Kase is a computer scientist for the Army Research Laboratory (ARL) at Aberdeen Proving Ground in Maryland. She completed a PhD at the Pennsylvania State University (PSU) in information sciences and technology and an MS at State University of New York in computer science. She has held post-doctoral positions at the Defense Threat Reduction Agency (DTRA), Mitre Corporation, and a visiting professor position at Richard Stockton University in New Jersey before moving to her present position at ARL. She continues to teach computer science courses part-time at local community colleges and the PSU World Campus.