COLLEGE OF INFORMATION SCIENCES AND TECHNOLOGY
THE PENNSYLVANIA STATE UNIVERSITY

Some Frontiers of Cognitive Modeling: A Modest Research Agenda Exploring Emotions and Usability

Frank E. Ritter

frank.ritter@psu.edu

Technical Report No. ACS 2008-1

1 November 2008

Applied Cognitive Science Lab
College of Information Sciences & Technology
University Park, PA 16802
acs.ist.psu.edu

Phone +1 (814) 865-4455   Fax +1 (814) 865-6426

Some Frontiers of Cognitive Modeling: A Modest Research Agenda Exploring Emotions and Usability

Frank E. Ritter

frank.ritter@psu.edu
Applied Cognitive Science Lab
College of Information Sciences & Technology
University Park, PA 16802
acs.ist.psu.edu
Phone +1 (814) 865-4455 Fax +1 (814) 865-6426

Technical Report No. ACS 2008-1

1 November 2008

Abstract

This paper reviews hybrid cognitive architectures that include both symbolic and non-symbolic components, a currently very active area of cognitive modeling. These architectures support new research directions including models of emotions. This paper reviews existing work with three hybrid architectures and notes some exciting problems that are now tractable. There remain problems creating models in these architectures, which itself remains a research and engineering problem. Thus, it introduces the term cognitive science engineering as an area that would support making models easier to create, understand, and re-use.

Acknowledgements

Support was provided by ONR Contract N00014-03-1-0248 and N00014-06-1-0164, and by the UK MOD's Directorate of Analysis, Experimentation and Simulation corporate research programme (Project No: RT/COM/3/006). Olivier Georgeon and Colin Sheppard provided useful comments that improved this version of this paper.

A much revised version of this report will appear as: Ritter, F. E. (invited, accepted October, 2008). Two cognitive modeling frontiers: Emotions and usability. *Journal of Japanese AI Research*.

Table of Contents

1. Introduction

Hybrid architectures combine the strengths of symbolic architectures with the strengths of lower level, sub-symbolic architectures. The symbolic processing based on what is essentially an AI or agent architecture allows the models to perform relatively large and complicated tasks, and the sub-symbolic components support modifying the performance in subtle but important ways, sometimes with the modifications happening gradually and sometimes more quickly. These architectures are increasingly available and usable.

This paper reviews cognitive modeling based on these cognitive architectures. This hybrid approach opens new research directions including models of emotions, models of individual differences, and models of the brain. It examining models of emotions in more detail.

There remain problems creating, reusing, and understanding these (and other models), which itself remains a research problem. This review is intended to inspire further research in what I see are interesting directions.

This review represents my thinking, and draws too much on projects that I am working on. Longer and more complete reviews (e.g., Morrison, 2003; Pew, 2007; Pew & Mavor, 1998, 2007; Ritter et al., 2003) include further work from a wider geographical and intellectual area, further projects, and include work on a wider range of architectures.

This review is intended, however, to be useful to a wide range of modelers and architects. The problems noted here are large enough that others could pursue them as well, indeed, they will require multiple researchers, there are plenty of applications, and we would appreciate the help.

2. Hybrid architectures

Cognitive architectures are an approach to describing the mechanisms of cognition that are fixed across users and across tasks. Cognitive architectures use cognitive psychology concepts like working memory and implement them in a computer program that takes in knowledge representations (typically rules, but it could include other types of task knowledge such as plans) to produce behavior. They are typically implemented as a computer program that applies task knowledge to produce behavior. Thus, the language has a representation for knowledge and a way to apply it. The application of knowledge in a cognitive architecture takes time, suffers from simulated limitations of human knowledge application, and often includes ways for errors to arise from not applying the correct knowledge or not applying the knowledge correctly.

Hybrid architectures can be defined as cognitive architectures with two types of knowledge representations that are at different levels or are of fundamentally different types. These architectures can be created in three ways. One way is by adding symbols to a connectionist representation (e.g., Sun & Bookman, 1994; Touretzky, 1986). Work in this area provides a way to either create symbols or structures or to find them amongst the connectionist representation.

Another way is by adding sub-symbolic representations to a symbolic architecture, for example, declarative memory strength in ACT-R and reinforcement learning in Soar 9. Here, the weighting you give to the rules and declarative memory elements is essentially a way to create a symbolic architecture that has changes that are small and gradual.

Most hybrid architectures are realized in these two ways, but there are also examples where one of the levels is a genetic algorithm, fuzzy logic, or other representation. Hybrid in this case does not mean symbolic and sub-symbolic, but typically symbolic and some higher level or orthogonal representations or process is used to supplant the strengths of the base architecture. Reviews of hybrid architectures (e.g., Kandel & Langholz, 1992) and further examples of hybrid architectures are available at conferences and workshops (e.g., Lewis, Polk, & Laird, 2007; Sun & Bookman, 1994).

I review three high-level hybrid architectures briefly to support readers not familiar with cognitive architectures or with these in particular. Additional architectures are noted further in the paper to illustrate particular points. These architectures include Clarion (Sun & Zhang, 2006) and EPIC (Kieras & Hornof, 2004).

2.1 Soar and ACT-R

Soar and ACT-R are two of the most commonly used cognitive architectures. They can be seen as theories of cognition realized as a set of principles and constraints on cognitive processing, a cognitive architecture (Newell, 1990). They both provide a conceptual framework for creating models of how people perform tasks. This review draws upon and updates a previous description of Soar and ACT-R (Ritter et al., 2003) for use here.

Both Soar and ACT-R are supported by a computer program that realize these theories of cognition. There are debates as to whether and how the theory is different from the computer program, but it is fair to say that they are at least highly related. It is generally acknowledged that the program implements the theory and that there are commitments in the program that must be made to create a running system that are not in the theory—places where the current theory does not say one thing or another.

As cognitive architectures, their designers intend for them to model the full breadth and width of human behavior. Such cognitive architectures, including the ones discussed in this report, do so to a greater or lesser extent, usually with the areas covered increasing monotonically over time. This approach to modeling human cognition is explained in books by Newell (1990) and Anderson (Anderson, 1993; Anderson & Lebiere, 1998). They are similar to other cognitive architectures such as PSI (Bach, in press) and the Cogent meta-architecture (Cooper, 2002).

Further comparisons of Soar and ACT-R are available (Johnson, 1997, 1998; G. Jones, 1996; Kennedy & Trafton, 2006; Ritter, 2003), and online from the Soar and ACT-R home pages, the Soar and ACT-R Frequently Asked Questions lists, and the Soar Wiki (available through Google).

2.1.1 Background of Soar and ACT-R

Soar and ACT-R are each based on a set of different theoretical assumptions, reflecting, largely, their different conceptual origins. Soar was developed by combining three main elements: (a) the heuristic search approach of knowledge-lean and difficult tasks; (b) the procedural view of routine problem solving; and (c) a symbolic theory of bottom-up learning designed to produce the power law of learning (Laird, Rosenbloom, & Newell, 1986). However, many of the constraints on Soar's theoretical assumptions consist of general characteristics of intelligent agents, rather than detailed behavioral phenomena. Soar's outlook is more biased towards performance because it arose out of a more AI-based tradition.

In contrast, ACT-R grew out of detailed phenomena from memory, learning, and problem solving (Anderson, 1983, 1990, 2007; Singley & Anderson, 1989). ACT-R is thus suited more for slightly lower-level phenomena, and is more suited for predicting reaction times, particularly for tasks under 10 seconds. ACT-R's outlook is more biased towards predicting reaction time means and distributions because it arose out of a more experimental psychology tradition. These differences are relative; both architectures have been used for both high and low level models, with attention paid to both performance and time predictions. Plenty of examples models are referenced on their home pages.

2.1.2 Similarities between Soar and ACT-R

Soar and ACT-R can be seen as similar in numerous ways. They both have two kinds of memory, declarative (facts) and procedural (rules), although they represent these items differently. Typical instantiations of them now have input provided through a model of perception and output buffered through a model of motor behavior (Byrne, 2001; Chong, 2001; Ritter, Baxter, Jones, & Young, 2000).

Both Soar and ACT-R model behavior by reducing much of human behavior to problem solving. Soar does this rather explicitly, being based upon Newell's information processing theory of problem solving, the Problem Space Computational Model (PSCM, Newell, 1968), whereas ACT-R merely implies it by being goal directed.

In both architectures memories are conceptually infinite, with no provision being made for the full removal of any memory item in ACT-R, although elements become less active with time and lack of use (the Soar architecture does perform removal of declarative memory in its goal stack, which therefore can be seen as a type of short-term memory). Manipulation of declarative memory can be accomplished by adding new items or changing existing ones. For procedural memory, rules may only be added to both architectures but not removed.

The course of processing involves moving from an initial state to a specified goal state. ACT-R has only one possible goal state (as a memory structure, there may be multiple internal and external situations where the goal is satisfied), whereas Soar may have several of them arranged in a stack. Movement between the initial and goal states usually involves the creation of sub-goals to accomplish the various steps leading up to the satisfaction of the goal.

Both ACT-R and Soar maintaining a goal hierarchy where each subsequent sub-goal becomes the focus of the system. In ACT-R, these must be satisfied in a serial manner, and in the reverse of the order they appear in the hierarchy (which is not directly visible to both the model and the modeler). Soar generally proceeds in a serial way as well, but is capable of removing (or solving) intermediate sub-goals should the current problem solving resolve a sub-goal that is much higher in the goal hierarchy. This difference makes ACT-R potentially less reactive, although work is in progress to make ACT-R more reactive (Lebiere, 2001; Salvucci, 2006).

2.1.3 Differences between Soar and ACT-R

There are also fundamental differences between the two architectures. Soar only moves between states through changing the state as part of a decision procedure, which rules can vote on but cannot directly cause. In Soar, when no more productions can fire, an operator is selected or a state is modified. This whole process is called a decision cycle. Where an operator cannot be selected (e.g., due to preferences for the set of operators conflicting each other or not being complete), a sub-goal is created with a goal to choose the next operator. Movement between states is done in ACT-R by firing productions, which may directly change the state and goal stack. Where there is no rule that matches, many versions of ACT-R just stop.

Soar allows multiple rules to fire in parallel. This may lead to impasses because the knowledge in the rules may suggest different operators, but problem solving is available to resolve this. In ACT-R, when the conditions of several productions are met, a conflict resolution mechanism selects the production that it estimates to have the highest gain; if no rule is available, many versions of ACT-R have just stopped (and thus, there may be a default rule added that waits).

Learning in Soar occurs only for production memory. New rules are created by the architecture whenever a sub-goal is resolved, such that when next encountering the same situation, the new production fires without the need to enter a new sub-goal. This type of information can include which operator to select, or how to implement an operator. These rules tend to be atomic, and in nearly all cases until recently can be seen as immediately fully learned. This learning mechanism (chunking) can implement a wide range of learning effects, including long-term declarative memory learning—for long-term declarative information is represented solely as the result of procedural memory. Soar 9, the latest version, includes reinforcement learning.

ACT-R learning involves both declarative and procedural memory. When rules fire they become stronger, and as declarative memories are used more they are strengthened as well. Each production also has an expected gain value based on its probability of success and its cost and the current goal's value. The expected gain is used for conflict resolution; the production with the highest expected gain is selected when several productions are possible matches. The more often the production meets with later success (e.g., the sub-goal ends up being solved), the higher this probability for the rule will become. This strength also influences the activation of the declarative memory items that are matched by the condition of the production.

Each item in declarative memory has an associated activation that changes based upon how often it has been used, and how strongly it is associated with other items that are being used. The more often an item is used, the higher its base level activation will become. The more strongly associated an item is with ones that are being used, the more chance that item has for having its activation raised.

A rule learning mechanism is less often used in ACT-R models, and when it has been used, the resulting rules are typically created in a nascent state such that they have to be created several times before they are fully learned (e.g., G. Jones, Ritter, & Wood, 2000; Taatgen & Lee, 2003).

2.2 CoJACK

CoJACK combines BDI agents' high-level knowledge representation and usability with several aspects of low level cognitive architectures, including traceability, processing time predictions, and errors (Norling & Ritter, 2004a; Ritter & Norling, 2006). CoJACK attempts to include lessons from ACT-R and from Soar (Norling & Ritter, 2004b). CoJACK also includes important aspects of macrocognition, including variability in its performance of a long-term task, and demonstrates the effects of behavioral moderators. It also allows aspects of situation awareness (SA) to be explored in a cognitive architecture by labeling the source of beliefs. Its behavior and the effects of moderators on performance are demonstrated in a simple adversarial environment. It provides lessons for other architectures including how to define, measure, and control variability that arises from individual and temporal aspects of cognition (Evertsz, Busetta, Pedrotti, Ritter, & Bittner, 2008); the importance of SA and knowledge representations necessary for complex SA; and some of the complexities that will arise when we try to add aspects of SA to cognitive architectures (Evertsz, Ritter, Russell, & Shepherdson, 2007).

2.2.1 JACK

To understand CoJACK, it is necessary to briefly describe JACK, the Java Agent Construction Kit (Busetta, Rönnquist, Hodgson, & Lucas, 1999). JACK is based on the so-called Belief-Desires-Intentions (BDI) model, a paradigm that explicates rational decision-making about actions. Drawing on Bratman's (1987) work on situated rational agents, this approach has been applied to a wide range of problems, including fault diagnosis for Space Shuttle missions (Georgeff & Ingrand, 1989). The BDI paradigm was developed in response to a perceived problem with existing AI approaches to planning. Agents are typically situated in a dynamic environment and must constantly review their goals and activities. They should also be aware of the resource-bounded nature of their reasoning. Earlier AI approaches to planning only addressed the offline planning problem—how to achieve the goal given infinite time and resources. They failed to address the temporal pressures that apply when trying to achieve goals within a fluctuating environment that presents a multitude of interacting, conflicting, and changing opportunities.

From the perspective of a cognitive architecture, the key programming constructs of JACK are:

Event—the central motivating factor in agents that are generated in response to external stimuli or as a result of internal computation.

Plan—a procedure that defines how to respond to an event. When an event is generated, JACK computes the set of plans that are applicable to the event. The agent selects the plan that will form its next intention. Plans have a body that defines the steps to be executed in response to the event. Non-deterministic choice allows the agent to try alternative plans to achieve the goal.

Beliefsets—the agent's declarative beliefs in a first order, tuple-based relational form. Beliefsets are analogous to the working memory of a production system.

Intentions—the currently active plan instantiations, that is, the plan instances that the agent is committed to. A plan becomes an intention when the agent instantiates it with symbolic references to concrete entities in the environment and commits to its execution.

JACK represents and executes tactics in a manner that maps well to SMEs' (Subject Matter Experts) introspections about their own reasoning processes. The tactics representation includes a front-end that allows analysts to specify tactics graphically at a high-level (as shown in Figure 1). The graphical representation is useful for visualizing the logical structure of tactics and discussing them with SMEs. This representation language provides higher level constructs and thus direct support for usability. Therefore, a key design goal for CoJACK was to retain JACK's high-level representation and ease-of-use.

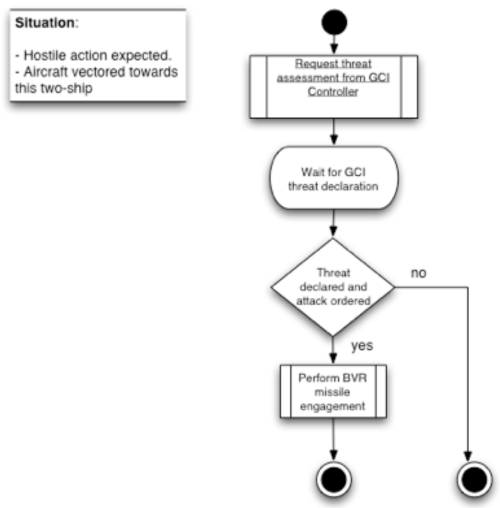

Situation:
- Hostile action expected.
- Aircraft vectored towards this two-ship
Request threat assessment from GCI Controller
Wait for GCI threat declaration
Threat declared and attack ordered
no
yes
Perform BVR missile engagement

Figure 1. A simple plan represented in JACK.

2.2.2 CoJACK extends JACK with timing, errors, and moderators

CoJACK augments JACK with a set of cognitive architectural constraints and parameters, as well as a moderator layer. The major activities in the JACK architecture, such as adding a belief or instantiating a plan have a (simulated) time cost associated with them. Based upon cognitive parameters' values, the architectural constraints add latency to the current intention's reasoning steps and to memory access.

CoJACK adds noise to several of the decision processes, which then affects the choice of beliefs retrieved in response to a memory access attempt; this includes effects such as failure to retrieve a matching belief, retrieval of a belief that only partially matches, and retrieval of an alternative matching belief (i.e., not the one that JACK would have chosen first). A similar mechanism affects the selection of the next intention to execute. Thus, the agent can choose an unanticipated intention or even fail to retrieve one of its current intentions, and it has less resources creating a more limited focus of attention.

CoJACK adjusts weights of rules and declarative memory objects, so it also learns. The learning is not of new plans, but different behaviour can arise in it, which is somewhat unusual for an agent architecture.

The cognitive parameters can be moderated at runtime, leading to further variation in behaviour. For example, a caffeine moderator can be added that decreases the time taken to perform reasoning steps, leading to shorter response times, which we demonstrate later.

Three cognitive models have been created in CoJACK. We have created a model of serial subtraction, a model of how Rules of Engagement will influence and be influenced by behavioural moderators (Evertsz, Ritter, Russell, & Shepherdson, 2007), and a very simple model that plays in a simple tank game (Evertsz, Busetta, Pedrotti, Ritter, & Bittner, 2008).

2.3 Summary

The review described three hybrid architectures each with a strong symbolic component. These architectures have been used to create cognitive models, although fewer with CoJACK. With their descriptions in mind, I examine how they have been used to create models of emotions and high-level modeling languages.

3. Models of emotions

An area of increasing interest in science, design, cognitive science, and cognitive modeling is emotions. There is interest in defining and understanding emotions (e.g., Picard, 1997) and in applying this understanding to design (Norman, 2006). Cognitive science and cognitive modeling are interested in creating models that have emotions in them as a way to define and understand emotions, and these models provide useful definitions of emotions and provide a way to explore what emotions are, how they may be related, and how they influence cognition and performance.

I would like to note two ways emotions have been created in cognitive architectures (out of several ways they can be, Ritter, 1993). They have been created by modifying an existing architecture, and they have been created by incorporating emotions directly into a new architecture.

3.1 Overlays

A very direct way to include emotions in a cognitive architecture is to modify an existing architecture. With an architecture like ACT-R, which includes multiple parameters and which is built upon an interpreted language, the changes can be represented as an overlay to the architecture. These overlays can be broken into types similar to the way that Gray (2007) proposes architectural components can be represented: Changes can be made to parameters, changes can be made to architectural components, and changes can be made to strategies or knowledge representations. Overlays have been created in each of the three architectures discussed above, and I provide examples taken from the Soar and ACT-R communities as well present a single example in CoJACK.

3.1.1 Soar

There have been several attempts to include emotions in Soar (Chong, 1999; Marinier & Laird, 2007). The largest attempt is to include appraisal theory. Gratch and Marsalla (2004) created a theory of how task appraisal influences cognition as an overlay to Soar. This overlay both made use of existing mechanisms in Soar, as well as adding additional levels of analyses that can support more complex problem solving and reasoning about emotions including coping. It included additions to the architecture to provide more appropriate input and output, and also additions to knowledge to reason about emotions and emotional aspects of decision making.

It is probably an extreme case of an overlay in that the authors probably see their theory not as an extension to the Soar architecture, but as using Soar as a component. Nevertheless, it is a nice example of how an emotional theory can be added to an architecture.

They learned some lessons from this work that are generally applicable. They found that additional types of information were required to create their models with emotions. They found that emotional content could be useful when generating interactions (e.g., answering the question 'what happened?' can be informed by the emotional aspects of the situation). They note that the appraisal process and behavior and the physiology of the agent will interact. This interaction has been noted before, but this work shows that having a running model will make the work more interesting, and modeling this interaction remains a very interesting problem.

3.1.2 ACT-R

ACT-R has been used numerous times to create overlays of emotions and behavioral moderators related to emotions. We have created a set of overlays for ACT-R to model the effects of stress (Ritter, Reifers, Schoelles, & Klein, 2007; Ritter, Schoelles, Klein, & Kase, 2007). Others have created overlays representing the effects of development on cognition (G. Jones, Ritter, & Wood, 2000), fatigue (Jongman, 1998), stress (Belavkin & Ritter, 2000), arousal (Cochran, Lee, & Chown, 2006), and sleep loss (Gunzelmann, Gross, Gluck, & Dinges, in press).

In each case, these overlays implement theories of how cognition changes. Typically, they modify a parameter within the existing architecture. In a few cases they modify the equations that the architecture uses to select a rule to fire or how declarative memories are retrieved.

This approach is incremental, but it provides a way for theories of emotions and behavioral moderators to be formally represented. And the resulting theories generally are available and will work with other models, for example, a driving model, to provide predictions of how driving is influenced by fatigue, or sleep loss, or other moderators.

3.1.3 CoJACK

CoJACK, which includes nearly all of the ACT-R equations, has had emotional overlays created for it as well. These overlays include a stressful overlay and an overlay representing the effects of caffeine (submitted), and a fear overlay (Evertsz, Ritter, Russell, & Shepherdson, 2007). A schematic of how overlays interact with JACK and CoJACK is shown in Figure 2.

The overlays modify the parameters in CoJACK to represent changes such as working memory capacity and access speed, processing speed, and noise in the decision process (Evertsz, Busetta, Pedrotti, Ritter, & Bittner, 2008).

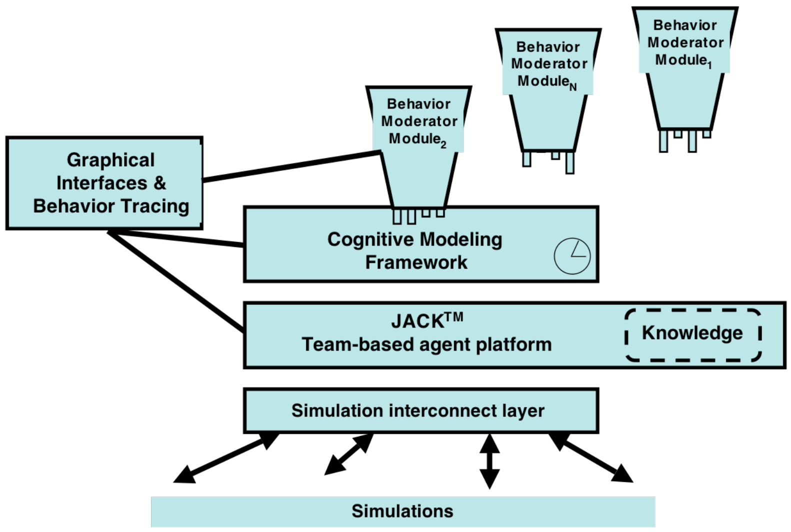

Behavior Moderator Module$_1$
Behavior Moderator Module$_N$
Behavior Moderator Module$_2$
Graphical Interfaces & Behavior Tracing
Cognitive Modeling Framework
JACK™ Team-based agent platform
Knowledge
Simulation interconnect layer
Simulations

Figure 2. Schematic of CoJACK's relationship to JACK, and how moderators influence performance by adjusting parameters in CoJACK.

We have explored how to include a model of stress and of caffeine in CoJACK as it plays in the dTank environment. Figure 3 shows how a team of 4 CoJACK tanks with variable amounts of simulated caffeine play against four unmoderatored tanks. This graph shows that, averaged over 30 runs at each level, that as the simulated amount of caffeine goes up, performance first improves and then decreases, simulating the inverted U-shaped curve that is often found for caffeine on performance.

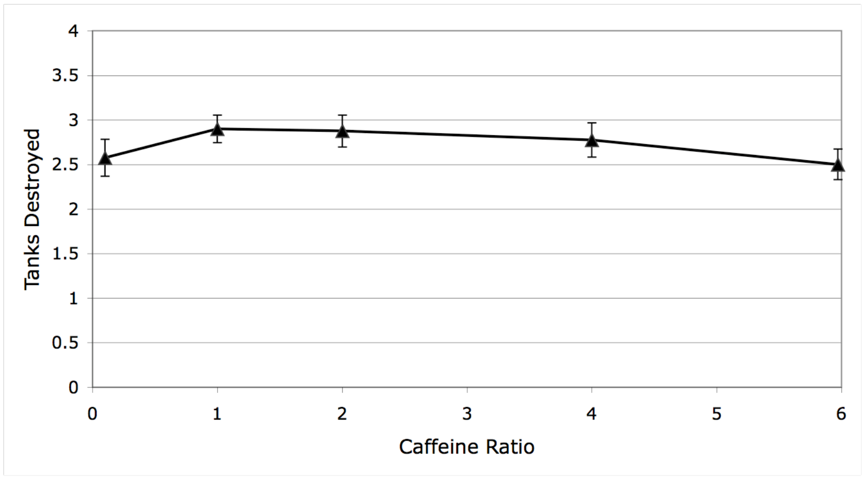


Figure 3. The effects of increasing amounts of simulated caffeine on the performance of a CoJACK agent in dTank.

Summary

Overlays have been created in each of these three architectures. These overlays have been used to implement theories across a wide range of effects and suggest that most emotions and moderators can be explored in this way.

These overlays have already raised some interesting questions. For example, the work on modeling sleep deprivation in ACT-R (Gunzelmann, Gross, Gluck, & Dinges, in press) has raised questions about how to generalize results across tasks and across populations. The work modeling caffeine have left us slightly stumped without additional data—yes, people with caffeine get faster, but how to apportion the speed up among the multiple mechanisms? Additional data from additional tasks will be needed.

Initial work has started with relatively simple and in most cases static overlays. In time, the overlays will have to represent a more dynamic system that is being modeled—emotions interaction with cognition, and behavior moderators influence strategy choices, which in turn influence further changes to cognition.

Similar work is possible in many other architectures that may be appropriate for a particular task. It would be straight forward to implement similar overlays to EPIC (Kieras, Wood, & Meyer, 1997), Clarion (Sun & Zhang, 2006), and even GOMS (Kieras, 1998)!

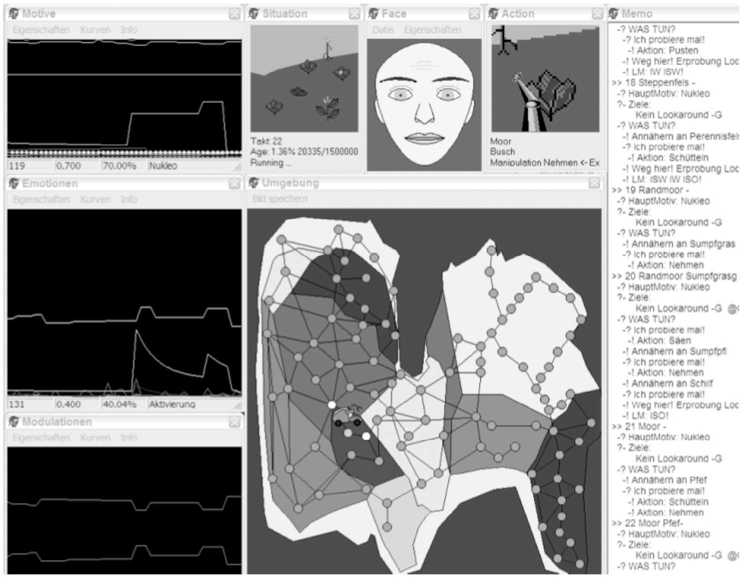
3.2 PSI and other systems with drives

Dietrich Dörner (2000; , 2003) proposed the PSI architecture. This architecture includes drives that interact with most cognitive mechanisms. The drives help clarify how emotions could arise and change the way that the architecture works on a fundamental level, providing an architecture more suited for behaving autonomously in a simulated world, which it does.

PSI includes three types of drives, physiological (e.g., hunger), social (i.e., affiliation needs), and cognitive (i.e., reduction of uncertainty and expression of competency). These drives routinely influence goal formation and knowledge selection and application. The resulting architecture generates new kinds of behaviors, including context dependent memories, socially motivated behavior, and internally motivated task switching.

Joscha Bach (in press) has recently implemented this theory as the MicroPSI architecture. This architecture illustrates a way that emotions and physical drives can be included in an embodied cognitive architecture. The PSI architecture, while including perceptual, motor, learning, and cognitive processing components, also includes several novel knowledge representations, including temporal structures, spatial memories, and several new information processing mechanisms and behaviors, including progress through types of knowledge sources when problem solving (the Rasmussen ladder), and knowledge-based hierarchical active vision. These mechanisms and representations suggest ways for making other architectures more realistic, more accurate, and easier to use.

The architecture is demonstrated in the Island simulated environment, as shown in Figure 4. While looks like a simple game, it was carefully designed to allow (and require) multiple tasks to be pursued and provides ways to satisfy the multiple drives. It would be useful in its own right for developing other architectures interested in multi-tasking, long-term learning, social interaction, embodied architectures, and related aspects of behavior that arise in a complex but tractable real-time environment.



```
  -? WAS TUN?
    -? Ich probiere mal!
      -! Aktion: Pusten
    -! Weg hier! Erprobung Loc
    -! LM: IW ISW!
>> 18 Steppenfels -
  -? HauptMotiv: Nukleo
  ?- Ziele:
        Kein Lookaround -G
  -? WAS TUN?
    -! Annähern an Perennisfel
    -? Ich probiere mal!
      -! Aktion: Schütteln
    -! Weg hier! Erprobung Loc
    -! LM: ISW IW ISO!
>> 19 Randmoor -
  -? HauptMotiv: Nukleo
  ?- Ziele:
        Kein Lookaround -G
  -? WAS TUN?
    -! Annähern an Sumpfgras
    -? Ich probiere mal!
      -! Aktion: Nehmen
>> 20 Randmoor Sumpfgrasg
  -? HauptMotiv: Nukleo
  ?- Ziele:
        Kein Lookaround -G  @
  -? WAS TUN?
    -? Ich probiere mal!
      -! Aktion: Säen
    -! Annähern an Sumpfpfl
    -? Ich probiere mal!
      -! Aktion: Nehmen
    -! Annähern an Schilf
    -? Ich probiere mal!
    -! Weg hier! Erprobung Loc
    -! LM: ISO!
>> 21 Moor -
  -? HauptMotiv: Nukleo
  ?- Ziele:
        Kein Lookaround -G
  -? WAS TUN?
    -! Annähern an Pfef
    -? Ich probiere mal!
      -! Aktion: Schütteln
      -! Aktion: Nehmen
>> 22 Moor Pfef-
  -? HauptMotiv: Nukleo
  ?- Ziele:
        Kein Lookaround -G  @
  -? WAS TUN?
```

Figure 4. A PSI agent in the Island simulation (taken from Bach, in press), showing how emotions vary during processing (left hand side), views of the agent and the agents' emotional state as a face, and the agent's location on the island.

While there have been tests of the architecture's predictions to human data (Detje, 2000), the resulting models in PSI are generally presented as validated cognitive models, but as theoretical explorations in the space of architectures for generating behavior. The sweep of the architecture can thus be larger—it presents a new cognitive architecture attempting to provide a unified theory of cognition.

Similar systems that include emotions more directly have been created, such as Barry Silverman's PMServe (2004), Ron Sun's Clarion (Sun, 2006), and by Eva Hudlicka's (2002; , 2000) MAMID. These architectures with emotions included in them and PSI in particular have several lessons for other architectures and models. This is not a typical cognitive modeling work, but one that I believe that we can learn much from. These architectures with drives in them suggest that there are mechanisms related to drives, important aspects of human behavior, that have been abstracted out of many models. For simple systems, this is not a problem. But as models attempt to model a wider range of behavior including the effects of drives will become more important, and overlays will eventually not be able to cope with the breadth of effects.

3.3 Summary and open problems

So, with time, models will come to include the effects on cognition of emotions and behavioral moderators like stress, caffeine, and lack of sleep that most experimental methods have attempted to remove. These are important for applications and important theoretically as well. They are important for applications because we would like to predict not ideal behavior by users, but actual behavior. And while many users are fresh and alert, not all are, and many are important users in high-stakes positions like truck drivers, pilots, and power plant operators. In addition to predicting behavior for design, it will also be important for simulations to have models that do not do the right thing in a simulation, as these will be opponents to take advantage of and colleagues that need support.

Architectures will need to include both changes to information processing that arise from emotions and moderators, such as through overlays, and they will have to include drives, such as PSI. Including these effects will be important for theoretical reasons. Not all of cognition occurs in a laboratory setting. Implementing behavioral moderators and emotions in architectures will help codify theories in these areas, and can help generate more accurate predictions Initial efforts are likely to be way off, but with further work we should be able to develop more theoretically strong definitions of emotions and the effects of moderators. In the end it will require creating a simulated body to hold the physiology of the mechanisms, including fatigue, neurotransmitters, and energy and hormonal systems. This is consistent with Sloman's view, for example, about supporting cognition by providing a body [cite]. Which will be at least interesting and quite likely complicated. As embodying cognition progresses it will also offer a way to unify the life sciences. Before that happens, we will need summaries of how specific moderators influence cognition on multiple tasks. These summaries will take time, but be invaluable for creating overlays to architectures.

4. Usable models

It has been noted several times and by at last a few commentators that cognitive models are not always easy to create and use (Pew & Mavor, 1998, 2007; Ritter et al., 2003). It is the case that they are not as easy to use as we would all like them to be. There are probably multiple causes for this, but what this means is that processing cognitive models are not used as theoretical constructs as often as they deserve to be. Whenever theory cannot be applied because it is too difficult, it might be viewable as an engineering problem, which I now think the usability of models is.

I would like to introduce the term cognitive science engineering, or perhaps engineering cognitive science. This is a new area for studying how to make and apply cognitive science. This area cannot be called cognitive engineering, because that name is already taken and refers to using cognitive psychology and cognitive science to improve the design of systems like power plants. Cognitive science tends to reject the engineering of models as cognitive science in the same way that psychology tends to reject software for psychology as psychology (although tools for physics and chemistry is physics and chemistry).

The usability of models and the ability to create and understand large models appear to be the first and perhaps most important type of cognitive science engineering. Other examples of cognitive science engineering are now clearly visible in a previous review (Ritter, et al., 2003), such as connecting models to simulations, where both engineering and science interact to create plausible, accurate connections between a model and a simulated or real world (Byrne, 2001; Ohno & Ogasawara, 1999; Ritter, Baxter, Jones, & Young, 2000), and optimizing the fit of models to data (Kase, 2008). These are not problems directly in the science of cognitive science, but they are problems that stop cognitive science from progressing.

The usability of models has been addressed by the three architectures differently, and provide lessons for each other and for other architectures. So I examine them separately here with respect to high-level programming languages.

4.1 Soar

There have been several attempts to provide an interface to the rule level in Soar. The Developmental Soar Interface (DSI) was one of the first (Ritter & Larkin, 1994). Earlier interfaces were modes in the Emacs editor that helped with parentheses balancing. Since then, there have been several console based interfaces that provided commands on menus (Ritter, Jones, & Baxter, 1998) and the ability to create aliases (Nichols & Ritter, 1995) for the command line. The current version of Soar (9) has a pretty nice interface on this level.

More recently, structured editors for rules have been provided as part of the Soar release. These, such as Visual-Soar, are becoming more sophisticated and complex, and start to organize the rules as parts of operators, and provide a datamap of the structures used across operators. These tools help with relatively low-level programming, but are becoming more high-level as time progresses.

There are two more promising approaches that propose higher level languages that compile into Soar. The first high-level language for Soar was TAQL (Yost, 1993). A small study suggested that it allowed users to write Soar models about three times faster. It was left behind when Soar moved to C from Lisp, and was not trusted (perhaps incorrectly) because some of its initial models could not learn when learning was turned on. HLSR is a more recent attempt to create a high level language for Soar and ACT-R (R. M. Jones, Crossman, Lebiere, & Best, 2006). It can create Soar and ACT-R models using a variety of programming constructs. It has been used at Soar Tech.

Herbal is the another attempt to create a high-level language for Soar (Haynes, Cohen, & Ritter, in press). It is implements a version of the Problem Space Computational Model (Newell, 1990; Newell, Yost, Laird, Rosenbloom, & Altmann, 1991) as a language that compiles into Soar and Jess models. It has been used by over 100 people, and our current best estimate is that it allows undergraduate psychology majors to create (small) cognitive models more than 3 times faster than computer science graduate students.

4.2 ACT-R

ACT-R currently has a nice rule-level interface, but it does not have a general high-level language. However, there are several projects on how to make simplified ACT-R models more quickly. These include special purpose languages like ACT-Simple (Salvucci & Lee, 2003), which creates simple models; G2A (St. Amant, Freed, & Ritter, 2005), which creates ACT-R models from GOMS models;

CogTool, a system to build simple models automatically from visual descriptions (John, Prevas, Salvucci, & Koedinger, 2004), and there is a language for menu interaction being developed (Urbas & Leuchter, 2005).

ACT-R has an increasing range of theories that can and are being reused across models. These are not individually or together a general language, but they help build models, and may provide lessons on how to get model reuse, a related concept, or may provide a type of module-based high-level language. The largest one is an architectural component that implements a theory of perceptual motor control mechanisms (formerly, ACT-R/PM, but now just simply included as part of ACT-R) (Byrne, 2001). There is also has a theory of eye-movements called Emma (Salvucci, 2001), a model of arithmetic that we have reused (Lebiere, 1999), and a reusable theory of multitasking (Salvucci, 2005).

4.3 CoJACK

CoJACK was partially created based on a review that argued that it would be easier to create an interface and extension to JACK to make it cognitive than it would be to use Soar (Shakir, 2002). This report had some flaws, for example, a limited sample size of researchers were contacted and the author did not know Soar or JACK. It was, however, very encouraging that usability was considered in this decision, and in the several years that I have worked with CoJACK, users who knew Java did not complain about the usability of CoJACK. There have been and no doubt will be complaints about CoJACK, but it appears fairly usable.

There are probably several reasons that CoJACK appears or is usable (which might be the same thing in some cases). It is probably usable because it provides a set of high level constructs that it inherits from its BDI approach. These constructs are easy to use because they use folk psychology terms that programmers and experts themselves use, which may be why it appears usable. There are questions we are still exploring of whether these are the constructions that people use internally to generate behavior (as compared to describing their and others' behavior), but the translation from BDI constructs to cognitive science constructs, if they are different, might be supported later. Also, a lot of work went into designing and creating a comprehensive interface, broadly defined. This interface defined a lot of entry points to the code, and I believe forced some revisions in the architecture as it was developed. And there is a manual, which is slightly unusual for a such a young architecture. Finally, it could be argued that Java and the constructions are complex enough that at least in our case only elite programmers have used it.

The next step for CoJACK is to use the interface to create larger, more complex models, and then, to validate them. That will be real proof that the architecture is usable.

4.4 Open problems

There is a need to create larger models, that is, with more knowledge in them, and this need will occur in every architecture. A recent review (Ritter et al., 2006) found that there were multiple groups working in this area. All of these architectures are headed towards providing high-level languages. This work will be important for some time to come.

This paper also introduces the term of cognitive science engineering. This refers to the engineering of cognitive science systems. The theories in cognitive science take as their basis a computational metaphor for cognition. This necessarily means software and software system creation. Creating the various aspects of the cognitive theories as software takes time and effort. Such building processes around other sciences get called engineering, and thus, cognitive science engineering. Creating this label may provide a way to describe this type of work, to help practitioners find each other, and provide a way to label commonly occurring problems.

5. Conclusions

This brief review has described several hybrid cognitive architectures. These hybrid architectures discussed in greatest detail, Soar, ACT-R, CoJACK, and PSI, are symbolic architectures that are extended with a sub-symbolic representation. This extension provides them with more accurate timing predictions (particularly including variability in performance), errors (because the symbolic processes become less accurate at applying the knowledge that they have), and the ability to include moderators where basic processes become less accurate.

These projects have both theoretical and applied aspects. They provide new theories about cognition, how it changes over time and tasks, and how people differ. These theories are strong theories in that they predict the time course of information processing, and can be tested with summary as well as sequential data about users. These theories also have multiple applied uses, which can provide an audience for such work, and can be great fun as well.

There are, of course, other research directions. For example, there is interesting work on mapping the cognitive architectures proposed to brain regions. There are at least three important efforts in this area. Anderson (2007) is using ACT-R, Just is using 4CAPS (Just & VarMa, 2007), and the connectionists and neural modelers are using Leabra (O'Reilly & Munakata, 2000) and other architectures (Levine, 2000). And it is increasingly possible to model individual differences. Early work showed that this was possible with a purely symbolic models (e.g., Miwa & Simon, 1993; Siegler, 1988), including symbolic versions of Soar (Recker & Pirolli, 1995; Ritter & Bibby, 2008). Hybrid architectures include parameters that are predicted to vary across people, such as working memory capacity (e.g., Daily, Lovett, & Reder, 2001), so it will be increasingly possible to model individual differences related to those parameters. We have started to find that ACT-R, for example, appears to model only individuals, in that the range of performance on the task by a group is much wider than performance by the model (Kase, Ritter, & Scholles, 2008; Ritter, Schoelles, Klein, & Kase, 2007). Exploring individual differences will be computationally expensive because the model has to be compared not just to the group but also to each individual. This is increasingly possible because of the availability of supercomputer resources (also called high-performance computing, or HPC).

This paper also introduces the term of cognitive science engineering. This refers to the engineering of cognitive science systems. The theories in cognitive science take as their basis a computational metaphor for cognition. This necessarily means software and software system creation. Creating the various aspects of the cognitive theories as software takes time and effort. Such building processes around other sciences get called engineering, and thus, the term cognitive science engineering. Creating this label may provide a way to describe this type of work, to help practitioners find each other, and provide a way to label commonly occurring problems and solutions.

It is worth pointing out that this work noted here does not require much technology for progress. Much of the work can be accomplished with a stopwatch, a way to make audio and perhaps video recordings, and any common computer. However, this work will use technology if it is available, for example, high-performance computing. Thus, it can be taken up in many research settings.

These projects are examples of how computational modeling of cognition can be used to unify psychology and to make increasingly accurate predictions of human behavior. Similar projects are available applying cognitive models to other areas of human behavior. As such, they offer an excellent way to build collaborations across areas of psychology and the social sciences, and to support applications in virtual worlds and to support interface design.

This review has noted several interesting problems in modeling in emotions. These research problems are not unique to the architectures discussed—the same problems are general and will arise in other architectures. So, they will keep us busy for quite a while, and can be attacked by a range of researchers. The research agenda is modest in that we have only just begun to accomplish work in this area. It is not modest at all in that the research agenda is quite large.

References

Anderson, J. R. (1983). *The architecture of cognition*. Cambridge, MA: Harvard University Press.

Anderson, J. R. (1990). *The adaptive character of thought*. Hillsdale, NJ: Erlbaum.

Anderson, J. R. (2007). *How can the human mind exist in the physical universe?* New York, NY: Oxford University Press.

Bach, J. (in press). *Principles of synthetic intelligence: Building blocks for an architecture of motivated cognition*. New York, NY: Oxford University Press.

Belavkin, R., & Ritter, F. E. (2000). Adding a theory of motivation to ACT-R. In *Proceedings of the Seventh Annual ACT-R Workshop*, 133-139. Pittsburgh, PA: Department of Psychology, Carnegie-Mellon University.

Bratman, M. E. (1987). *Intention, plans, and practical reasoning*. Cambridge, MA: Harvard University Press.

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. (2001). ACT-R/PM and menu selection: Applying a cognitive architecture to HCI. *International Journal of Human-Computer Studies, 55*(1), 41-84.

Chong, R. S. (1999). Towards a model of fear in Soar. In *Proceedings of Soar Workshop 19*, 6-9. U. of Michigan Soar Group. Retrieved from www.eecs.umich.edu/~soar/sitemaker/workshop/19/rchong-slides.pdf on 22 November 2006.

Chong, R. S. (2001). Low-level behavioral modeling and the HLA: An EPIC-Soar model of an enroute air-traffic control task. In *Proceedings of the 10th Computer Generated Forces and Behavioral Representation Conference*, 27-35. Division of Continuing Education, University of Central Florida. www.sisostds.org/cgf-br/10th/: Orlando, FL.

Cochran, R. E., Lee, F. J., & Chown, E. (2006). Modeling emotion: Arousal's impact on memory. In *Proceedings of the 28th Annual Conference of the Cognitive Science Society* 1133-1138. Erlbaum: Mahwah, NJ.

Cooper, R. P. (2002). *Modelling high-level cognitive processes*. Mahwah, NJ: Erlbaum.

Daily, L. Z., Lovett, M. C., & Reder, L. M. (2001). Modeling individual differences in working memory performance: A source activation account. *Cognitive Science, 25*(3), 315-355.

Detje, F. (2000). Comparison of the PSI-theory with human behaviour in a complex task. In *Proceedings of the Third International Conference on Cognitive Modelling*, 86-93. Universal Press: Veenendaal, The Netherlands.

Dörner, D. (2000). The simulation of extreme forms of behavior. In *Proceedings of the Third International Conference on Cognitive Modelling*, 94-99. Universal Press: Veenendaal, The Netherlands.

Dörner, D. (2003). The mathematics of emotions. In *Proceedings of the Fifth International Conference on Cognitive Modelling*, 75-80. Universitäts-Verlag Bamberg: Bamberg, Germany.

Evertsz, R., Busetta, P., Pedrotti, M., Ritter, F. E., & Bittner, J. L. (2008). CoJACK—Achieving principled behaviour variation in a moderated cognitive architecture. In *Proceedings of the 17th*

Conference on Behavior Representation in Modeling and Simulation, 80-89. 08-BRIMS-025. U. of Central Florida: Orlando, FL.

Evertsz, R., Ritter, F. E., Russell, S., & Shepherdson, D. (2007). Modeling rules of engagement in computer generated forces. In *Proceedings of the 16th Conference on Behavior Representation in Modeling and Simulation*, 07-BRIMS-021. U. of Central Florida: Norfolk, VA.

Georgeff, M. P., & Ingrand, F. F. (1989). Monitoring and control of spacecraft systems using procedural reasoning. In *Proceedings of the Space Operations Automation and Robotics Workshop*.

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

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: Oxford University Press.

Gunzelmann, G., Gross, J. B., Gluck, K. A., & Dinges, D. F. (in press). Sleep deprivation and sustained attention performance: Integrating mathematical and cognitive modeling. *Cognitive Science*.

Haynes, S. R., Cohen, M. A., & Ritter, F. E. (in press). A design for explaining intelligent agents. *International Journal of Human-Computer Studies*.

Hudlicka, E., & Pfautz, J. (2002). Architecture and representation requirements for modeling effects of behavior moderators. In *Proceedings of the Eleventh Conference on Computer Generated Forces and Behavioral Representation*, 9-20. 02-CGF-085. Division of Continuing Education, University of Central Florida: Orlando, FL.

Hudlicka, E., Zacharias, G., & Psotka, J. (2000). Increasing realism of human agents by modeling individual differences: Methodology, architecture, and testbed. In *Simulating Human Agents, American Association for Artificial Intelligence Fall 2000 Symposium Series*, 53-59. AAAI Press: Menlo Park, CA.

John, B. E., Prevas, K., Salvucci, D. D., & Koedinger, K. (2004). Predictive human performance modeling made easy. In *Proceedings of CHI 2004 (Vienna, Austria, April 2004)*, 455-462. ACM: New York, NY.

Johnson, T. R. (1997). Control in ACT-R and Soar. In *Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society*, 343-348. Erlbaum: Hillsdale, NJ.

Johnson, T. R. (1998). A comparison of ACT-R and Soar. In U. Schmid, J. Krems & F. Wysotzki (Eds.), *Mind modeling - A cognitive science approach to reasoning, learning and discovery* (pp. 17-38). Lengerich, Germany: Pabst Scientific Publishing.

Jones, G. (1996). The architectures of Soar and ACT-R, and how they model human behaviour. *Artificial Intelligence and Simulation of Behaviour Quarterly, 96 (Winter 1996)*, 41-44.

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

Jones, R. M., Crossman, J. A. L., Lebiere, C., & Best, B. J. (2006). An abstract language for cognitive modeling. In *Proceedings of the 7th International Conference on Cognitive Modeling*, 160-165. Erlbaum: Mahwah, NJ.

Jongman, G. M. G. (1998). How to fatigue ACT-R? In *Proceedings of the Second European Conference on Cognitive Modelling*, 52-57. Nottingham University Press: Nottingham.

Just, M. a., & VarMa, S. (2007). The organization of thinking: What functional brain imaging reveals about the neuroarchitecture of complex cognition *Cognitive, Affective, & Behavioral Neuroscience, 7*(3), 153-191.

Kandel, A., & Langholz, G. (1992). *Hybrid architectures for intelligent systems*. Boca Raton, FL: CRC Press.

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

Kase, S. E., Ritter, F. E., & Scholles, M. (2008). From modeler-free individual data fitting to 3-D parametric prediction landscapes: A research expedition. In *Proceedings of the 30th Annual*

Conference of the Cognitive Science Society, 1398-1403. Cognitive Science Society: Austin, TX.

Kennedy, W. G., & Trafton, J. G. (2006). Long-term learning in Soar and ACT-R. In *Proceedings of the Seventh International Conference on Cognitive Modeling*, 162-168. Edizioni Goliardiche: Trieste, Italy.

Kieras, D. E. (1998). A guide to GOMS model usability evaluation using NGOMSL. In M. Helander (Ed.), *Handbook of human-computer interaction* (Vol. 12, pp. 391-438). Amsterdam: Elsevier.

Kieras, D. E., & Hornof, A. J. (2004). Building cognitive models with the EPIC architecture for human cognition and performance. In *Proceedings of ICCM - 2004 - Sixth International Conference on Cognitive Modeling*. Erlbaum. simon.lrdc.pitt.edu/~iccm/proceedings/schedule.htm: Mahwah, NJ.

Kieras, D. E., Wood, S. D., & Meyer, D. E. (1997). Predictive engineering models based on the EPIC architecture for a multimodal high-performance human-computer interaction task. *Transactions on Computer-Human Interaction, 4*(3), 230-275.

Laird, J. E., Rosenbloom, P. S., & Newell, A. (1986). Chunking in Soar: The anatomy of a general learning mechanism. *Machine Learning, 1*(1), 11-46.

Lebiere, C. (1999). The dynamics of cognition: An ACT-R model of cognitive arithmetic. *Kognitionswissenschaft, 8*, 5-19.

Lebiere, C. (2001). Multi-tasking and cognitive workload in an ACT-R model of a simplified air traffic control task. In *Proceedings of the 10th Computer Generated Forces and Behavioral Representation Conference*, 91-98. Division of Continuing Education, University of Central Florida. www.sisostds.org/cgf-br/10th/: Orlando, FL.

Levine, D. S. (2000). *Introduction to neural and cognitive modeling*. Lawrence Erlbaum: Mahwah, NJ.

Lewis, R. L., Polk, T. A., & Laird, J. E. (Eds.). (2007). *Proceedings of the 8th International Conference on Cognitive Modeling*. Oxford, UK: Taylor & Francis/Psychology Press.

Marinier, R. P., & Laird, J. E. (2007). Computational modeling of mood and feeling from emotion. In *Proceedings of the 29th Annual Conference of the Cognitive Science Society*. Cognitive Science Society: Austin, TX.

Miwa, K., & Simon, H. A. (1993). Production system modeling to represent individual differences: Tradeoff between simplicity and accuracy in simulation of behavior. In *Prospects for artificial intelligence: Proceedings of AISB'93*, 158-167. ISO Press: Amsterdam.

Morrison, J. E. (2003). *A review of computer-based human behavior representations and their relation to military simulations* (IDA Paper P-3845): Institute for Defense Analyses.

Newell, A. (1968). On the analysis of human problem solving protocols. In J. C. Gardin & B. Jaulin (Eds.), *Calcul et formalisation dans les sciences de l'homme* (pp. 145-185). Paris: Centre National de la Recherche Scientifique.

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

Newell, A., Yost, G. R., Laird, J. E., Rosenbloom, P. S., & Altmann, E. (1991). Formulating the problem space computational model. In R. F. Rashid (Ed.), *Carnegie Mellon Computer Science: A 25-Year commemorative* (pp. 255-293). Reading, MA: ACM-Press (Addison-Wesley).

Nichols, S., & Ritter, F. E. (1995). A theoretically motivated tool for automatically generating command aliases. In *Proceedings of the CHI'95 Conference on Human Factors in Computer Systems*, 393-400. ACM: New York, NY.

Norling, E., & Ritter, F. E. (2004a). A parameter set to support psychologically plausible variability in agent-based human modelling. In *The Third International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS04)*, 758-765. ACM: New York, NY.

Norling, E., & Ritter, F. E. (2004b). *COJACK Software Specification*: Agent Oriented Software Limited.

Norman, D. A. (2006). *Emotional design: Why we love (or hate) everyday things*. New York, NY: Basic Books.

O'Reilly, R. C., & Munakata, Y. (2000). *Computational explorations in cognitive neuroscience: Understanding the mind by simulating the brain*. Cambridge, MA: MIT Press.

Ohno, T., & Ogasawara, H. (1999). Information acquisition model of highly interactive tasks. In

Proceedings of ICCS/JCSS-99, the Joint Conference of International Conference of Cognitive Scince/Japanese Cognitive Science Society Annual Meeting, 288-293. Japanese Cognitive Science Society: International Convention Center, Waseda University.

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 Academy Press. books.nap.edu/catalog/6173.html.

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

Picard, R. W. (1997). *Affective computing*. Cambridge, MA: MIT Press.

Recker, M. M., & Pirolli, P. (1995). Modeling individual differences in students' learning strategies. *Journal of the Learning Sciences, 4*(1), 1-38.

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. (2003). Soar. In L. Nadel (Ed.), *Encyclopedia of cognitive science* (Vol. 4, pp. 60-65). London: Nature Publishing Group.

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., & Bibby, P. A. (2008). Modeling how, when, and what learning happens in a diagrammatic reasoning task. *Cognitive Science, 32*, 862-892.

Ritter, F. E., Haynes, S. R., Cohen, M., Howes, A., John, B., Best, B., et al. (2006). High-level behavior representation languages revisited. In *Proceedings of ICCM - 2006- Seventh International Conference on Cognitive Modeling*, 404-407. Edizioni Goliardiche: Trieste, Italy.

Ritter, F. E., Jones, R. M., & Baxter, G. D. (1998). Reusable models and graphical interfaces: Realising the potential of a unified theory of cognition. In U. Schmid, J. Krems & F. Wysotzki (Eds.), *Mind modeling—A cognitive science approach to reasoning, learning and discovery* (pp. 83-109). Lengerich, Germany: Pabst Scientific Publishing.

Ritter, F. E., & Larkin, J. H. (1994). Developing process models as summaries of HCI action sequences. *Human-Computer Interaction, 9*(3), 345-383.

Ritter, F. E., & Norling, E. (2006). Including human variability in a cognitive architecture to improve team simulation. In R. Sun (Ed.), *Cognition and multi-agent interaction: From cognitive modeling to social simulation* (pp. 417-427). Cambridge, UK: Cambridge University Press.

Ritter, F. E., Reifers, A. L., Schoelles, M., & Klein, L. C. (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., 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. Oxford, UK: Taylor & Francis/Psychology Press.

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), formerly known as the Crew System Ergonomics Information Analysis Center (CSERIAC).

Salvucci, D. D. (2001). An integrated model of eye movements and visual encoding. *Cognitive Systems Research, 1*(4), 201-220.

Salvucci, D. D. (2005). A multitasking general executive for compound continuous tasks. *Cognitive Science, 29*, 457-492.

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

Salvucci, D. D., & Lee, F. J. (2003). Simple cognitive modeling in a complex cognitive architecture. In *Human Factors in Computing Systems: CHI 2003 Conference Proceedings*, 265-272. ACM: New York, NY.

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

Siegler, R. S. (1988). Strategy choice procedures and the development of multiplication skill. *Journal of Experimental Psychology : General, 117*(3), 258-275.

Silverman, B. G. (2004). Human performance simulation. In J. W. Ness, D. R. Ritzer & V. Tepe (Eds.), *The science and simulation of human performance* (pp. 469-498). Amsterdam: Elsevier. www.seas.upenn.edu/~barryg/Ch9-final.pdf.

Singley, M. K., & Anderson, J. R. (1989). *The transfer of cognitive skill*. Cambridge, MA: Harvard University Press.

St. Amant, R., Freed, A. R., & Ritter, F. E. (2005). Specifying ACT-R models of user interaction with a GOMS language. *Cognitive Systems Research, 6*(1), 71-88.

Sun, R. (2006). The Clarion cognitive architecture: Extending cognitive modeling to social simulation. In R. Sun (Ed.), *Cognition and multi-agent interaction*. New York: Cambridge University Press.

Sun, R., & Bookman, L. A. (Eds.). (1994). *Computational architectures integrating neural and symbolic processes: A perspective on the state of the art* (Lawrence A. ed.). Norwell, MA: Kluwer Academic Publishers.

Sun, R., & Zhang, X. (2006). Accounting for a variety of reasoning data within a cognitive architecture. *Journal of Experimental and Theoretical Artificial Intelligence, 18*(2), 169-191.

Taatgen, N. A., & Lee, F. J. (2003). Production compilation: A simple mechanism to model complex skill acquisition. *Human Factors, 45*(1), 61-76.

Touretzky, D. S. (1986). *A distributed connectionist production system* (technical report No. CMU-CS-86-172): Computer Science Department, Carnegie Mellon University.

Urbas, L., & Leuchter, S. (2005). Model based analysis and design of human-machine dialogues through displays. *KI – Zeitschrift für künstliche Intelligenz [AI-Journal for AI]*, 45-51.

Yost, G. R. (1993). Acquiring knowledge in Soar. *IEEE Expert, 8*(3), 26-34.