Are Cognitive Architectures Mature Enough to Evaluate Notification Systems?

Douglas G. Turnbull, C. M. Chewar, and D. Scott McCrickard
Center for Human-Computer Interaction and Department of Computer Science
Virginia Polytechnic Institute and State University (Virginia Tech)
Blacksburg, VA 24061-0106
dturnbul@vt.edu; {cchewar, mccricks}@cs.vt.edu

Abstract

This study investigates the use of cognitive architectures for usability and software engineering of notification system user interfaces. These interfaces, which are special in that they are used in a divided-attention situation, require careful study of multi-tasking support of the information design. Cognitive architectures appear promising for this type of research, since they attempt to model human information processing characteristics related to system events. Three versions of cognitive architectures are compared according to criteria that would support research of notification systems. Data is gathered from empirical observation and is used to draw conclusions about suitability for modeling notification interfaces (and thus reducing the expense of user testing). The findings show that no system is ideal, but a combination of features from each of the three systems may be an ideal future solution.

Keywords: human-computer interaction, ACT-R, Soar, EPIC, dual-task, information display, empirical study

1. Introduction

Although there have been many challenges, usability testing continues to dominate interface evaluation. The typical setup calls for gathering data and feedback from participants as they interact with interfaces in a controlled computing environment. However, testing sufficient numbers of the appropriate kinds of participants bottlenecks the software development process. Because of this, challenges, work-arounds, or modifications to pure usability testing are not rare [10, 14]. A recently maturing sector of cognitive science, *cognitive architectures*, could fulfill the role of usurper once-and-for-all. Cognitive architectures such as ACT-R, Soar, and EPIC, much like computer architectures, specify the mechanisms and constraints of human information processing, including theories of memory, attention, perception, and action. Implementations contain perceptual-motor systems designed specifically for interaction with graphical user interfaces, potentially giving interface evaluators cheap, instantaneous methods of extracting volumes of data about usability in a completely managed environment. Modeling the human side of human-computer interaction could partially or completely eliminate the need for human participants in usability evaluation.

A specific set of tools, notification systems, provides an interesting challenge for the "cognitive architecture methodology." *Notification systems* are a special type of human-computer interface attempting to deliver current, important information to the user in an efficient and effective manner [8, 9]. Examples include stock tickers, instant messengers, and email tool-bar notifications. Notification system designers build systems with a user's complete task environment in mind. For example, a designer could argue that users of a notification system would feel annoyed, if while immersed in an important task, they are notified with a "pop up" window. On the other hand if the notification is very important, and the user is not occupied perhaps a pop-up window would be appropriate. Many methods alerting users of information, including pop-up windows, sound, secondary displays and real-world interfaces are deployed as appropriate based on the expected or specified task-environment of the user.

Task environments encase an enormous variety of tasks. Tasks can be described as "primary," the main focus of a user's attention, or "secondary," tasks that demand occasional or no attention. A distinction also exists originating with Norman between complex tasks, described with a deep and wide decision tree, and shallow tasks, described with a narrower and shallower tree [11]. Notification systems exist as complex, simple, primary, or secondary tasks, and everything in between. A well designed notification system allows itself to become secondary while still alerting the user of needed information in a manner conducive with its complexity and secondary status. Therefore, for cognitive architectures of be effective, user behavior cannot simply progress linearly with a single task, but must be prepared for sudden or subtle shifts of cognitive resources to a set of tasks as simple as closing a "pop-up" and as complex as playing an engaging chess match.

2003 International Conference on Software Engineering Research and Practice (SERP 2003), Las Vegas NV, June 2003.

As the study of Notification Systems becomes an increasingly important topic within HCI, it is important to assess the suitability of current cognitive architectures. It is hoped that such results can help steer the development of these architectures so that they can positively impact notification system research. This study proceeds along two paths. First we inquire about the practicality of applying cognitive architectures to any Human-Computer Interaction domain. Architectures must be easy to install and learn. A brief survey of HCI literature is done to examine precedents for using cognitive architectures.

2. Research Questions

Three questions provide a test to establish whether task interaction is properly modeled by the architectures. These questions can be thought to probe the cognitive architecture's ability to handle the spectrum of notification system assertiveness.

First, the most assertive: *Does the architecture properly handle forcible interruptions of one task by another?* An example of this phenomenon is seen when a pop-up window appears in foveal vision, forcibly diverting attention to itself.

Second: *How much attention must a cognitive architecture apply to a peripheral stimulus to perceive and choose to react to it?* A secondary task that gently asserts itself peripherally gives users a choice to ignore or attend to the secondary task. An email tool-bar notification allows users to choose whether to divert attention to their email client, or continue with their ongoing task.

Third: *Can the architecture comprehend and learn information occurring pre-attentively?* Ambient displays and many real-world interfaces maintain a constant peripheral stream of information. The goal is to maintain a level of comprehension over a period of time that leaves, at the least, a "feel" for ongoing events on in long-term memory. Architectures must allow memory to be pre-attentively altered at a perceptual level while gradually establishing links between stimulus and meaning on a higher, long-term level of understanding.

Next, we discuss the methodology used to asses the ACT-R, Soar and EPIC systems with respect to these questions.

3. Methodology

These questions, combined with the aforementioned practical concerns of installation, learnability, and HCI literature presence correspond to six criteria that are considered. In considering each criteria, we assign acceptability points on a five point scale: five being completely acceptable and one being completely unacceptable. Using this scale is clearly subjective, it should be considered as both our educated opinions as well as extensive user feedback. The real substance of this study lies not in the ratings but in the experienced chronicled here—giving both architecture designers and HCI practitioners guidance on the practical and theoretical capabilities of cognitive architectures.

The basic methods for approaching each criteria were the same. Installations were generally given a two-hour time limit. However, if we felt progress could still be made after two hours, installation continued until success or until all avenues had been exhausted. We also judged the learnability of a system by what could be learned within a two hour period by an advanced HCI student with a strong computer science background. Resources obtained from the architecture's websites[1] formed the focus of the learning sources. Reviewing the HCI literature revolved around analyzing the results from the ACM® Web Portal using the keywords <cognitive architecture> AND HCI (e.g. ACT-R and HCI). Beneficial results either evaluate an interface or extend HCI theory with a cognitive architecture. To answer the aforementioned concerns about modeling complex task interaction, documentation by the system's authors referred to as "primary literature" is queried first, with secondary sources being called upon when the primary literature fails to provide answers.

4. Observations and Results

This section presents the details related to the empirical observations achieved using each cognitive architecture. Most discussion and conclusions are saved for the next section.

4.1. Cognitive Architectures in HCI Literature

HCI practitioners use ACT-R and Soar similarly. Both architectures are used to explore how users reason through interfaces. Typically both are used together to confirm one-another. Rieman et al implement ACT-R and Soar models to support consistency in interface design [13]. Peck and John account for 90% of user browsing behavior in an "on-line help browser" [12]. These results, however, were scant. Out of a combined twenty-one results for ACT-R and twenty-nine for Soar (with much overlap) only four involved the use of ACT-R and Soar for theory extension. Many of the other results were workshops of studies promoting the use of cognitive architectures in HCI. The results, while not prolific, point to a small precedent of using ACT-R and Soar to make theoretical points.

[1] Architecture websites: ACT-R: http://act-r.psy.cmu.edu/,; EPIC: http://www.eecs.umich.edu/~kieras/epic.html, Soar: http://ai.eecs.umich.edu/soar/)

Kieras and Meyer dominate EPIC's literature. Each of their papers develops the capabilities of EPIC further—always with the disclaimer that EPIC remains a research system [6, 7]. Extending HCI theory is no the goal of these works. Hornof and Kieras use EPIC to show how users anticipate the location of items in a menu [4]. The authors themselves admit that menus have been heavily studied in the field, hence they merely demonstrate EPIC's ability to function within an established framework. Other studies include an attempt to model the multi-modal behavior of telephone operators. Out of the twenty-five results, none of the studies involve expansion of HCI theory. The results are more scant than the ACT-R and Soar results.

4.2. System Installation and Learnability

Installation ease varied drastically from architecture to architecture. ACT-R came packaged as a "zip" file containing a win32 installation executable. On execution, an easy to follow "wizard" interface guided installation. An ACT-R environment was up and running in no less than twenty minutes after beginning the download. Installing Soar, however was much more trouble. We exhausted the two hours attempting to install Soar. After going down several blind alleys and being forced to back-track, we discovered that Soar's installation depends heavily on the installed Tcl/Tk version (a prerequisite for Soar). After upgrading to Tcl/Tk 8.4, installation was successful. EPIC was only partially installable. EPIC comes packaged as LISP source code. The authors have yet to document an installation procedure and do not provide any method for testing installation.

Similarly, ACT-R and Soar provided extensive tutorials on their websites, while EPIC had none whatsoever. ACT-R's installation includes a trainer environment and example modules which correspond to examples in one of nine tutorial units. The first unit prints to approximately twenty pages, and took the entire two hours to complete. The tutorials target an audience in psychology, which can be a hindrance for computing professionals. For example, the first tutorial spent considerable space explaining a concept akin to "scope." The tutorial instructs based on an earlier version of the environment, meaning some instructions are faulty. Overall, ACT-R's tutorials come in easy to swallow packages which fruitfully combine theory with practice.

Soar's tutorials come in four parts, each prints to about sixty to eighty pages. Two hours was spent to the first third of the first tutorial. Each tutorial focuses on a specific problem. In the first tutorial the student must devise simple strategies for a "pac-man®" style game called "Eaters." To work with Eaters examples, the tutorial requires an additional piece of software called Visual Soar. The experience, however, was very fun, engaging and easy.

4.3. Examining Architecture Characteristics

ACT-R, EPIC, and Soar share several cognitive structures. Cognitive architectures divide elements of human memory between *declarative memory*, descriptive memory about things (what a bike is) and *procedural memory*, methods for doing things (how to ride a bike). While the names assigned to units of declarative memory change per architecture, elements of procedural memory are almost always referred to as *production rules* or as abbreviated *productions*. Production rules take their name from their syntax which is of the form: If some declarative memory precondition is met, then alter declarative memory. Cognitive architectures execute by repeatedly selecting production rule(s) whose preconditions are met then executing the selected rule(s).

To develop models of cognitive phenomena, a modeler specifies the production rules, the structures of declarative memory, and declarative memory's initial state. Models are tested against empirical data on the phenomena and refined. Recall the three criteria: (1) *Does the architecture properly handle forcible interruptions of one task by another?* (2) *How much attention must a cognitive architecture apply to a peripheral stimulus to perceive and choose to react to it? and* (3) *Can the architecture comprehend and learn information occurring pre-attentively?* Understanding the specific constraints of each architecture allows a closer examination of its ability to capture user behavior within the task environment.

4.3.1. ACT-R. In addition to production rules and *chunks*, ACT-R's unit of declarative memory, ACT-R implements *goals* – chunks that encapsulate an end-state of declarative memory. Since goals are chunks, production rules may create *subgoals* to establish a prerequisite state of declarative memory. For example, if ACT-R aims to obtain a college degree, a natural subgoal might be to pass freshman calculus. ACT-R maintains goal-subgoal relationships using a *goal stack*. That is, ACT-R pushes new subgoals on top of their parent goals much like a computer program's run-time stack pushes called procedures on top. ACT-R stands out by enforcing the serial execution of production rules. Productions are strategically selected based on past experience using that production rule with the current goal [1].

Does the architecture properly handle forcible interruptions of one task by another? ACT-R's goal stack setup obviates concern about whether a secondary, unrelated goal can interrupt the ongoing goal-stack. Anderson and Lebiere address this problem. A production rule could create a subgoal or change the current goal to reflect interruptions of one task by another. The example they give involves escaping a fire:

IF the goal is to do any task

AND one hears "FIRE!

Then escape the fire.

However, this solution comes with a warning:

> Cognitive Psychology has tended not to be in the business of creating such emergency interrupts and studying the cognition that results. Therefore, we cannot say that ACT-R's attention mechanism is the right mechanism for modeling such interrupt handling because there is no data with which to assess it. All we can say is that there is no inherit contradiction between such interrupt handling and ACT-R goal structures [1].

Nevertheless, Gray and Altman present an alternate argument. Gray and Altman discuss cognitive strategies of memory management during task switches and develop an ACT-R model that strategically coordinates the activation of task-specific memory [2]. This study, unfortunately, stands alone in the literature. Moreover, Gray and Altman's study sticks to two related and simple tasks. More research is needed before it can be said whether or not ACT-R can handle abrupt and forced task-switching—an essential characteristic for investigating usability questions of notification systems.

How much attention must a cognitive architecture apply to a peripheral stimulus to perceive and choose to react to it? In order to respond to a stimulus pre-attentively, ACT-R must be able to select a production rule based on knowledge about the periphery pre-attentively. Production rules only test chunks from declarative memory. Therefore any stimulus must arrive in declarative memory for it to be testable. To get to declarative memory, attention must be directed to objects in visual or auditory memory via commands in a production rule's execution side. For example, an approaching fire would prompt the mind to react, yet ACT-R would completely ignore the fire unless attention was focused on it [1].

In addition, it is highly unlikely that ACT-R would select production rules unrelated to the current goal. In ACT-R, any production rule has an associated expected gain relative to the current goal. The probability a production rule will be selected is high when its expected gain .is high ACT-R performs a cost-benefit analysis based on the production rules past progress toward the goal [6]. In short, there is a small probability that an unrelated production, such as the "FIRE!" example above, would get chosen when other productions remain. ACT-R stays preoccupied with a good book instead of escaping a fire.

ACT-R completely ignores a secondary task prompting reaction if the task is unrelated to the current goal. Unfortunately this also severely limits this cognitive architecture's ability to represent many notification systems.

Can the architecture comprehend and learn information occurring pre-attentively? ACT-R creates a distinction between chunks received about the world— "the characters '3 + 4 = ?' "—and chunks resulting from goal completion— "the result of adding 3 and 4 was 7." As the same productions process the former type into the latter type, the result becomes reinforced. Hence, when "3 + 4 = ?" are seen, ACT-R begins to respond with a result of "7" quicker on each successful addition. In notification systems, expert users comprehend visualizations similarly. Immediate mappings between stimulus and meaning grow with time. ACT-R provides extensive tools for increasing knowledge via automatic encoding and understanding of stimulus.

ACT-R fails, however, to support any pre-attentive comprehension. Over time the knowledge gained by a complex display might become automatic. However, a production rule must still be fired to recognize the previously solved problem. Productions are selected to support the current goal with secondary, unrelated goals or tasks remaining outside of consideration.

Therefore, while ACT-R fosters an interesting environment with which to observe how users attentively understand visualizations, it fails to support any parallelism whatsoever for pre-attentive comprehension. Unfortunately, notification systems that aim for pre-attentive comprehension are designed within completely separate constraints than those demanding complete attention, voiding any benefits ACT-R might bring.

4.3.2. EPIC.

EPIC represents cognition, perception, and action using several interconnected processors executing in parallel. Each processor corresponds to a component of cognition, perception or action. For instance, the "cogp" processor selects and executes production rules and interacts with working memory, similarly the "auditory" processor "listens" to audio input with write-only access to memory. EPIC's parallelism runs deep; all production rules whose preconditions match are executed.

Does the architecture properly handle forcible interruptions of one task by another? EPIC targets human factors and engineering psychology problems. Kieras's EPIC site states:

> The most important issue that we are studying with EPIC is the nature of human multiple-task performance: these are situations in which a person is executing more than one task simultaneously, such as tuning a radio while driving a car, or making tactical decisions while tracking a specific target in a military fighter aircraft [7].

EPIC's management of attention show this as well. EPIC represents each task as a set of production rules with a governing goal. EPIC can be built using an executive-control mechanism, working by shifting the focus of attention from one task's governing goal to another (synonymous to an operating system's scheduler). Using this framework, Kieras and Meyer modeled the results of many classic multi-tasking experiments. Despite EPIC's success in this realm, Kieras and Meyer always remind their audience that EPIC remains a research system not ready for mass use. In addition, little has been published since 1999 further chronicling the system [6].

How much attention must a cognitive architecture apply to a peripheral stimulus to perceive and choose to react to it? EPIC executes all production rules whose prerequisites are met, allowing a decision to be made without diverting from other decision-making resources. The multiple actions are capable of reacting with one response modality and not another. EPIC could easily swat a fly with its left hand and continue to work ardently on a paper. Attention, therefore, is extremely parallel and very capable of handling reactive tasks [7].

In EPIC, each input channel (vision, auditory, tactile) has its own processor. EPIC undergoes great lengths to ensure that visual events outside of the fovea are processed—allowing for reaction to events in the periphery. The visual field is divided into several zones. As an object moves closer to the fovea, more visual properties (location, color, etc) become available. Noticed changes in properties are reported to a visual perception processor that maintains data structures within working memory. This data remains available to EPIC's cognitive processor. There is only one drawback, however, there has not been enough detailed research on which visual properties should be available at what foveal distance [7].

In any case, EPIC provides a great window for exploring these issues. Given an appropriate model of vision, the reactive decision-making can be modeled while not distracting from a primary task's parallel cognition. It remains to be seen to what extent modeled peripheral reactions interfere impacts primary task cognition. In this sense, EPIC seems too parallel, but promising for notification systems research nevertheless.

Can the architecture comprehend and learn information occurring pre-attentively? Amongst the EPIC literature listed on EPIC's site, nothing is available describing EPIC's theories of learning. Indeed, EPIC has avoided intense single-task problem-solving cognition, concentrating on multitasking simpler tasks. With EPIC therefore, it seems perception-meaning mappings are "hard-wired."

However, EPIC attempts to provide adequate perceptual systems. As mentioned above, EPIC keeps tabs on all elements in vision, paying attention to what properties should be knowable at each level of perception. Therefore, testing a notification system for peripheral comprehension might prove more fruitful with EPIC.

The final piece of the puzzle lies in the massively parallel production rule selection and execution. Knowledge gain can occur without any reallocation of attention from the primary task. Productions to understand the display and productions to continue performing the primary task will fire simultaneously without interrupting one-another. This by itself seems unrealistic. Added with the concept of "hard-wiring" the display to EPIC and pre-attentive comprehension seems very unrealistically modeled. Seemingly removing both the potential for primary task disruption and the integral learning process does not allow for a fruitful exploration of comprehension. At best, "hard-wired" comprehension can model the most expert behavior, but only if the comprehension disrupts the primary-task appropriately.

4.3.3. Soar. Soar differentiates itself from ACT-R and EPIC by not allowing the right-side of production rules to alter declarative memory. Instead, productions propose operators which in turn act upon declarative memory. At any iteration in Soar's execution cycle, selected production rules propose a set of operators. Soar analyzes the proposed operators, considering intersecting proposals (two productions propose the same operator) and past experience. Unlike ACT-R, "goals" and "task" do not correspond to explicit structures within the architecture and may be represented multiple ways [3].

Does the architecture properly handle forcible interruptions of one task by another? Soar avoids formalizing goals and tasks. This leaves Soar open to a variety of approaches to model sudden interruptions. Unfortunately, nothing within the Soar primary literature attacks this problem. The secondary literature also yields very little results. Searching the "PsychInfo" database on OVID for "Soar + cognitive" returned 11 results, none of which were related to task switching, interruptions, or multitasking. The same search at IEEE® Explore produced four results, having little to do with task-switching or interruptions. Research is required to examine Soar's capabilities with abrupt interruptions, which would allow better assessment of the usefulness for notification systems research [3].

How much attention must a cognitive architecture apply to a peripheral stimulus to perceive and choose to react to it? After production rules propose a collection of operators, one must be selected and executed. Soar's choice involves choosing amongst proposed operators. Goals and tasks are secondary constructs in Soar. Therefore, it can be said that Soar's choice occurs without attention toward a goal or task. Operator selection occurs based upon a holistic analysis, rather than goal-specific, therefore allowing operators from separate tasks to enter into consideration. Soar's cognitive functions could fairly leverage the decision between very different operators, such as choosing between the operators: "check one's email" on peripheral perception or "continue to work," when the tradeoff [3].

A potential benefit and drawback to Soar is its lack of a perceptual-motor system. Modelers are expected to provide their own "input" and "output" functions. Input functions work on their own to provide data to working memory, instead of passively needing to be queried like ACT-R. Customized input functions might provide appropriate pre-attentive feature analysis [3].

Can the architecture comprehend and learn information occurring pre-attentively? Many times, Soar considers two operators, which appear equally favorable. On such an impasse, Soar performs complex levels of consideration before selecting an operator. Soar remembers the results of the scenario. Mapping perception onto action (pre-attentively or otherwise) is managed through a different mechanism. Production rules in Soar can perform "state elaboration" before proposing operators. The knowledge that "red" means, "my team scored!", for example, would be acquired outside of Soar's actions, exemplified in executing operators. The downside, as mentioned, is that modelers are expected to provide their own perceptual-motor systems. Both methods of learning provide a rich cognitive exploration of pre-attentive information [3].

Table 1

Rating	***ACT-R***	***Soar***	***EPIC***
Installation	5	2	1
Learnability	4	5	1
HCI Lit.	2	2	1
Question (1)	3	3	4
Question (2)	1	4	4
Question (3)	2	4	2
Total	17	20	13

Overall ratings, reflecting that no architecture is currently fully capable for modeling notification system user behavior. Each has its own specific flaws, and none has an overwhelming advantage.

5. Discussion and Conclusions

Cognitive architectures strive to capture the universal limitations and capabilities of human cognition, perception, and action. Modelers take architectures into their domains of study to refine a universal theory of cognition. To model notification systems, a broader view of cognition, not limited to a single task, but rather concerned with an individual's complete environment is needed. Architectures should seek crucial bottlenecks in perception, cognition, and action while elaborating on human beings' documented ability to multi-task [8]. EPIC sets out to address these concerns, but remains poorly documented, difficult to learn and unusable. Soar provides an avenue for notification systems issues, is easy to learn and use, yet requires modelers specify complete perceptual-motor models – no easy task. ACT-R is also easy to learn, use, and install but implements attention strictly serially. There is almost no chance that ACT-R would move away from a current task.

Each architecture has growing to do before applying it to multitasking–demands required in the study of notification systems. It appears, also, that each architecture concentrates on separate types of tasks. ACT-R and Soar, as exemplified in the HCI survey, concentrate on the complex cognition of problem-solving. Linear progress toward completing a complex task, with a deep, narrow decision tree, mostly ignores the possibility that several of these tasks could be interacting concurrently. EPIC, however, has concentrated on multitasking simpler tasks (shallow, broad decision trees). A user's environment undoubtedly overflows with every type of task, simple and complex. To truly be complete, architectures must begin to examine each other in detail, and learn. Until then, it is difficult to imagine testing notification systems with shoddy multi-tasking models, a poorly documented system, or a non-existent perceptual-motor system. However, if positive attributes from each were taken together, then automated models and evaluations of concurrent user interfaces might seem far less of an unattained ambition.

References

1. Anderson, L. *The Atomic Components of Thought* (Mahwah NJ, 1998), Erlbaum.
2. Altman, E and Gray, W. "An Integrated Model of Set Shifting and Maintanence." In *Proceedings of the Third International Conference on Cognitive Modeling* (Veenendaal, The Netherlands, 2000), 17-24.
3. Bates et. al., *The Soar User's Manual Version 8.2* (Online, 1998) [http://ai.eecs.umich.edu/soar/docs/manuals/soar8manual.pdf].
4. Hornof, A. and Kieras, D. E. "Cognitive Modeling Demonstrates How People Use Anticipated Location Knowledge of Menu Items." In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (1999) 410 – 417.
5. Kieras, D. E., Wood, S. D., and Meyer, D. E. "Predictive Engineering Models Based on the EPIC Architecture for a Multimodal High-Performance Human-Computer Interaction Task." *ACM Transactions on Computer-Human Interaction 4,* 3, (September 1997). 230-275.

6. Kieras, D. E. and Meyer, D. E. “Predicting Performance in Dual-task Tracking and Decision Making with EPIC Computational Models.” In *Proceedings of the First International Symposium on Command and Control Research and Technology* (Washington D.C. 1995), National Defense University, 314-325.
7. Kieras, D. E. and Meyer, D. E. *The EPIC architecture: principles of operation* (Online, 1998) ftp://www.eecs.umich.edu/people/kieras/EPIC/EPICArch.pdf.
8. McCrickard, D. S. and Chewar, C. M. “Attuning Notification Design to User Goals and Attention Costs.” *Communications of the ACM 46*, 3 (March 2003), 67-72.
9. McCrickard D. S., Czerwinski, M., and Bartram, L. “Introduction: Design and Evaluation of Notification User Interfaces.” *International Journal of Human-Computer Studies 8*, 5 (May 2003), 509-514.
10. Nielsen, J. “Why GUI Panic is Good Panic.” *Interactions 1*, 2 (1994), 55-58.
11. Norman, D. *The Psychology of Everyday Things* (1988).
12. Peck, V.A. and John, B. E. “Browser-Soar: A Computational Model of a Highly Interactive Task.” In *ACM CHI`92 Conference on Human Factors in Computing Systems,* (1992), 165-172.
13. Rieman, J. Lewis, C., Young, R.M., and Polson P.G. “Why is a Raven Like a Writing Desk? Lessons in Interface Consistency and Analogical Reasoning from Two Cognitive Architectures.” In *ACM CHI`94 Conference on Human Factors in Computing Systems,* (1994) 434 – 444.
14. “The State of the Art in Automating Usability Evaluation of User Interfaces.” *ACM Computing Surveys* 33, 4 (December, 2001), 470-516.