QN-ACES: Integrating Queueing Network and ACT-R, CAPS, EPIC, and Soar Architectures for Multitask Cognitive Modeling

Yili Liu
The University of Michigan

Comprehensive and computational models of human performance have both scientific and practical importance to human–machine system design and human-centered computing. This article describes QN-ACES, a cognitive architecture that aims to integrate two complementary classes of cognitive architectures: Queueing network (QN) mathematical architecture and ACT–R, CAPS, EPIC, and Soar (ACES) symbolic architectures. QN-ACES represents the fourth major step along the QN architecture development for theoretical and methodological unification in cognitive and human–computer interaction modeling. The first three steps—QN architecture for response time, QN-RMD (Reflected Multidimensional Diffusions) for response time, response accuracy, and mental architecture, and QN-MHP (Model Human Processor) for mathematical analysis and real time simulation of procedural tasks—are summarized first, followed by a discussion of the rationale, importance and specific research issues of QN-ACES.

1. INTRODUCTION

The increasing complexity of advanced human–machine systems makes it necessary for system designers to consider human capabilities and limitations as early as possible in system design. In order to reduce risks associated with poor task design with appropriate tools and methods for task analysis and function allocation, it is important to develop models of human performance and human-system interaction that are comprehensive, computational, science-driven, and application-relevant.

Models of human performance and human–system interaction should be comprehensive to capture the whole range of concurrent perceptual, cognitive, motor, and communication activities of human-system performance. These models should be computational and computerized to allow quantitative and rigorous simulation and analysis of design alternatives and scenarios. These models should be science driven, with deep roots in and strong connections with cognitive science.

Correspondence should be addressed to Yili Liu, Department of Industrial & Operations Engineering, The University of Michigan, 1205 Beal Avenue, Ann Arbor, MI 48109-2117. Email: yililiu@umich.edu
theories and principles. These models should also be application-relevant, striving to tackle and solve practical design problems, with an engineering philosophy that having an “imperfect” or approximate solution is better than no solution at all.

Along this line of research, for more than a decade, we have been developing a queuing network (QN) based unified theory and computational architecture of human performance and human–system interaction that simultaneously meets the criteria just listed: comprehensive, computational, science driven, and application relevant. Several major steps have been taken along this direction, each producing significant results and generating unique insights on cognitive architecture in general and the role of queuing networks in cognitive architecture in particular.

In this article I first summarize our accomplishments in three of the steps along this line of research and then discuss the next step called QN-ACES as reflected in the title of this article. More specifically, in the first step, a QN theory of reaction time (RT) was developed that integrates the influential architectural RT models as special cases, including the serial discrete-stages, the serial continuous-flow, and the discrete network models (such as the critical path network model). Further, the QN models cover a broader range of mental architectures and can be subjected to well-defined empirical tests. In the second step, the architectural RT models and the sequential information sampling RT/accuracy models are unified through QN-RMD (Reflected Multidimensional Diffusions). Specifically, the “state” of a K-server QN of mental architecture is represented as a reflected diffusion space of K dimensions, in which “reflecting barriers” reveal architectural constraints, while “absorbing barriers” represent accuracy-related response criteria. QN-RMD moves beyond the current one-dimensional random walk/diffusion/accumulator models that have successfully accounted for but are limited to single-stage fast responses. In the third step, QN-MHP (Model Human Processor) was developed to bridge the mathematical and the symbolic models of mental architecture and to support mathematical modeling and real-time generation of task performance and mental workload. QN-MHP expands the three discrete serial stages of perceptual, cognitive, and motor processing in MHP into three continuous-transmission subnetworks of servers, each performing distinct psychological functions specified with a procedural/symbolic language. Multitask performance and workload emerges as the network behavior of multiple streams of information flowing through a network. QN-MHP has been applied to generate and model a variety of tasks including the psychological refractory period, visual search, transcription typing, and driving a vehicle simulator.

This summary of the three steps provides the background for the discussion of the next step of our research—development of QN-ACES. In this article and in this step of our QN research, QN-ACES represents a cognitive architecture that integrates the QN architecture we have developed thus far and the complementary symbolic architectures of ACT–R, CAPS, EPIC, and Soar. I discuss the importance, the rationale, and some of the specific research issues for developing and evaluating QN-ACES. I do not present a comprehensive literature review of the broad field of cognitive and human–computer interaction (HCI) modeling in this article. Numerous reviews and books are listed in the references (e.g., Gluck &
The most crucial theoretical and methodological issues related to QN-ACES are discussed together with QN-ACES.

1. QUEUING NETWORK COGNITIVE ARCHITECTURE: FIRST THREE STEPS IN ITS DEVELOPMENT

Comprehensive models of complex human performance and human-system interaction should be deeply rooted in fundamental and unified theories of cognition (UTC), rather than ad hoc solutions. Based on this belief, in every step of our development of the QN cognitive architecture, we closely examine how the QN architecture relates to existing theories and how it can help bridge or unify isolated models. In every step, we address some of the most fundamental and enduring puzzles in psychology and mental architecture: why is there a delay between stimulus presentation and response initiation (called reaction time—RT) and what is the mental architecture explanation for this delay? Why is there a trade-off between RT and response accuracy and what is the relationship between RT, response accuracy, and mental architecture? What are the mathematical properties of mental architecture and its symbolic processing characteristics and how can mental architecture be represented and analyzed both mathematically and symbolically? In fact, examining these three questions constitute the research focus of the three steps we have undertaken in the QN architecture development.

1.1. QN Architecture of RT: Integrating Architectural and Information Transmission Models

Historically, the first groups of computational models of mental architecture are mathematical models, shown in the table on the left side of Figure 1, so we started our QN architectural development by exploring the relationship between QN and those mathematical models. In fact, the first model of mental architecture is a simple mathematical model called the subtractive method developed by Donders (1868/1969) that dates back even before the official birth of psychology as a scientific discipline in 1879. Donders assumed that psychological processes can be inserted to or deleted from a chain of processes and the mean duration of the inserted or deleted process can be inferred by examining the difference between the mean duration of a task that does not include the process in question and one that does—the method is thus called the subtractive method. The underlying model for the subtractive method assumes non-overlapping durations of serially-arranged mental processes or stages, as shown in the top-left cell of Figure 1. Other mathematical models of mental architecture, all developed in the last 40 years or so, shown on the left half of Figure 1, try to relax the original assumptions of Donders and expand the scope of modeling to cover a broader range of temporal and architectural arrangements that mental processes might assume. For example, the cascade model (McClelland, 1979) shown in the lower-left cell of Figure 1 investigates the possibilities of continuous flowlike serial mental processes that allow temporal overlap in processing, whereas the program-evaluation-and-review technique (PERT) network model (Schweickert, 1978), also
called critical-path network model, examines network arrangements of discrete mental processes. In a PERT network, processes can be arranged as a complex network with strictly serial and strictly parallel structures as special cases. Processes that are not on the same path of the network are allowed to be active at the same time, but those on the same path are assumed to operate in strict sequence. Thus, a PERT network model is also called a discrete network model, as shown in the top-right cell in the table on the left side of Figure 1 (see Liu, 1996, for details).

To cover a broader range of temporal and architectural arrangements that mental processes might assume, I developed QN models for mental architecture and RT (Liu, 1996). The QN mental architecture assumes that there is a close resemblance between the human mind and a QN. The idea of a QN arises naturally when one thinks of a network of service stations (also called servers or nodes), each of which provides a service of some kind to the demanders for service (called customers), either immediately or after a delay. Each server has a waiting space for customers to wait if they cannot immediately receive their requested service, and thus multiple queues may exist simultaneously in the system. The servers are connected by arcs over which customers flow from server to server in the network. Telecommunication systems, computer networks, and road traffic networks are examples of QNs.

The class of QN models, in its most general form, is continuous-transmission-network models, as shown in the lower-right cell in the table on the left half of Figure 1. As discussed in detail in that article, the class of QN models includes the existing models in the other three cells of the left half as special cases and thus is able to unify them in a larger modeling framework. I also reexamined the logic

<table>
<thead>
<tr>
<th>Mathematical Models of RT and Mental Structure Classified in terms of Discrete versus Continuous Information Transmission and Serial versus Network Architecture (from Liu, 1996)</th>
<th>Mathematical Models of RT and Response Accuracy (sequential sampling models)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architectural Arrangement of Mental Processes</td>
<td></td>
</tr>
<tr>
<td>Temporal Transmission</td>
<td>Serial</td>
</tr>
<tr>
<td></td>
<td>Stages</td>
</tr>
<tr>
<td>Discrete</td>
<td>Additive factors</td>
</tr>
<tr>
<td></td>
<td>General Gamma</td>
</tr>
<tr>
<td>Continuous</td>
<td>Cascade</td>
</tr>
<tr>
<td></td>
<td>Queueing series</td>
</tr>
</tbody>
</table>
and conclusions of the previous models. It turns out that many of the conclusions based on the previous models are open to alternative explanations. All the QN models in Liu (1996) were discussed in relation to empirical data. Furthermore, I showed that QN models allow us to cover a broader range of possible mental structures that mental system might assume but had not been modeled by previous models, such as feedback or non-unidirectional information flow, information “overtaking and bypassing,” and process dependencies or nonselective influence of factor effects and can be subjected to well-defined empirical tests (Liu, 1996).

1.2. QN Architecture of RT and Response Accuracy: Bridging the Gap Between Architectural and Sequential Information Sampling Models

The QN architecture and RT modeling published in Liu (1996) demonstrated the power of QN in modeling a greater variety of processing systems than earlier ones and in serving as a larger modeling framework for unifying some isolated models. The success of this modeling work represents the first and the foundational step in our journey of developing a unified computational theory of mental architecture and cognition. This modeling work focuses on the use of RT to infer mental architecture and is able to broaden the scope of thinking about the possible configurations of mental systems and the possible causes for certain RT phenomena. However, this work, as well as all the models in the table on the left side of Figure 1, is silent on the important question of how RT covaries with response accuracy. RT and accuracy are arguably the top two most frequently used performance measures in cognitive psychology research. Revealing their relationships to each other and to the underlying mental architecture is an important challenge to any work on mental architecture and is thus the focus of our second step in QN architecture development.

The importance of examining RT and accuracy together in RT analysis and modeling has been emphasized by many researchers (e.g., Audley, 1960; Pew, 1969; Wickelgren, 1977). A crucial requirement is that they both arise naturally from common processing mechanisms (Ratcliff, 1985). The class of mathematical models that have achieved the greatest success in this regard is the class of sequential information sampling (also called stochastic information accumulation) models, including random-walk models and related diffusion models, and counter or accumulator models, shown on the right half of Figure 1. All sequential sampling models share the notion that the human information processing system accumulates information over time until a preset response criterion is reached and this accumulation process evolves stochastically, either in discrete steps (top cell on the right side of Figure 1) or in a continuous manner (lower cell on the right side of Figure 1). More specifically, “counter models” assume discrete counting increments of evidence information (Laberge, 1962; McGill, 1963). The random walk models assume that in a two-choice response situation, the information accumulation process “randomly walks” in discrete steps between two decision boundaries (also called “absorbing barriers”) based on the value of a cumulative evidence variable, each boundary representing one of the two choices.
The process generally walks to the positive or the negative boundary depending on whether the value of the evidence variable is positive or negative (Link, 1975; Stone, 1960). The continuous versions of the random walk models are called diffusion models, which assume that the corresponding stochastic process drifts continuously toward the positive or the negative boundary, depending on whether the mean rate of information accumulation is positive or negative (Ratcliff, 1985; Ratcliff, Van Zandt, & McKoon, 1999). The term “accumulator models” is used broadly to refer to both discrete and continuous evidence accumulation (Audley & Pike, 1965; Usher & McClelland, 2001).

The two groups of models shown on the two sides of Figure 1 are both often referred to as RT models, but there has been a substantial gap between the two groups. In fact, one finds little cross-referencing between publications of the two groups. A close examination of the two groups of RT models reveals that each group has its own research focus. The first group focuses on using RT to infer the possible temporal and architectural structures of the underlying mental system that transforms stimulus into response. The second group focuses on modeling the relationship between RT and accuracy. The first group can be called models of RT and mental architecture, and the second group can be called models of RT and response accuracy. Each group has made great progress in modeling the aspects of RT it focuses on but has little to say on the issues of concern of the other group. Specifically, the first group has not made great progress in modeling the intrinsic relationship between RT and accuracy, whereas the second group has been relatively silent about the mental architecture in which the sequential samplings (such as random walks or diffusions) occur. In fact, the second group of models has demonstrated its utility mainly in modeling binary and fast responses involving only a single stage of psychological processing.

After the successful development of the QN model for RT to unify the architectural models of RT on the left side of Figure 1, as the second step of our QN modeling work, we have successfully extended the QN architectural model of RT to cover accuracy and established a natural bridge between the QN and the sequential sampling/diffusion models through a modeling approach called QN-RMD (Liu, 2005, 2007). I pointed out that existing one-dimensional sequential sampling models of RT and accuracy shown on the right side of Figure 1 are, in essence, state transition models of a single server or a single processing unit. RMD is needed to represent state transitions of multiple-server complex architectures. More specifically, the “state” of a K-server QN of mental architecture can be represented as a reflected diffusion space of K dimensions, in which “reflecting barriers” represent and reveal architectural constraints, whereas “absorbing barriers” represent accuracy-related response criteria. QN-RMD clarifies the relationship between “architectural” models and “state transition” models and allows the use of RT and accuracy together for revealing mental architecture. This approach moves beyond the current one-dimensional diffusion models that have successfully accounted for but are limited to single-stage fast responses. One-dimensional diffusions can only represent the “state” of a single-server system in stochastic information accumulation not multiserver architectures. This research establishes a larger framework to unify or bridge two currently isolated major groups of RT models and brings the sequential sampling modeling
approach to the multidimensional multi-server cognitive architectural domain (Liu, 2005, 2007).

1.3. QN-MHP: Bridging the Gap Between Mathematical and Symbolic Architectures

All the models shown in Figure 1 are mathematical models of psychological processes, which demonstrate their unique strengths in supporting precise mathematical analysis of the mental architectural or speed-accuracy trade-off issues within the scope of each of the models. However, a major limitation of these mathematical models for engineering design and application is their inability, when used as pure mathematical theories, to generate detailed actions of a person in specific task situations. They do not represent the specific steps a person may undertake or the specific knowledge a person may employ in accomplishing his/her specific goals in real world situations.

In contrast, another group of cognitive architectural models, called symbolic models (shown on the right side of Figure 2), demonstrates their particular strengths in modeling and generating the detailed procedures and actions that a person might choose when interacting with an interface or system. Two publications in 1973 set the stage for this endeavor of developing comprehensive symbolic models of cognitive architecture. In his book chapter titled “You Can’t Play 20 Questions with Nature and Win,” Newell (1973) advocated the development of UTC and made theoretical unification of micromodels and theoretical constructs an immediate and principal goal. Anderson and Bower (1973) published the HAM theory of human memory, initiating an effort to create a theory of cognition that meets the twin goals of precision and complexity. Over 3 decades of creative efforts of numerous researchers have led to the development of several important UTCs or harbingers to UTCs, exemplified by the MHP and the Goals, Operators, Methods, and Selections (GOMS) family of models (Card, Moran, & Newell, 1983; John & Kieras, 1996), ACT-R (Anderson et al., 2004; Anderson & Lebiere, 1998), CAPS (Just & Carpenter, 1992), EPIC (Meyer & Kieras, 1997a, 1997b), and Soar (Laird, Newell, & Rosenbloom, 1987; Newell, 1990; see the right half of Figure 2).

Although symbolic models use mathematics in analyzing some of the specific mechanisms or operations of their models (e.g., spreading activation and the set of fundamental equations in ACT–R, algebraic formulation of the Psychological Refractory Period (PRP) in EPIC), they lack mathematical frameworks for representing their overall “architectures”—that is, representing the interconnected arrangements of all the perceptual, cognitive, and motor components and their interaction patterns in one coherent mathematical structure. These models are symbolic models (Card & Newell, 1990), not mathematical models.

Clearly, the mathematical approach and the symbolic approach are complementary and the challenge is to develop a modeling approach that bridges the gap between the two approaches. After our successful development of the QN architecture for RT, we simultaneously embarked on two steps: development of QN-MHP to bridge the mathematical and the symbolic approaches and development of QN-RMD previously discussed.
A s i l l u s t r a t e d i n F i g u r e 2, M H P i s t h e f a t h e r o f s o m e o f t h e m a j o r s y m b o l i c (also called procedure/prod uction system) models of cognitive architecture. We thus started our efforts of integrating QN with the symbolic architectures (the two sides of Figure 2) by first integrating QN and MHP and its related GOMS method, through adopting and modifying three major components of the MHP/GOMS.

First, we decompose the three discrete processors (perception, cognition, and motor) of the MHP into three continuous subnetworks of QN servers based on extensive reviews of neuroscience and psychological findings and utilize the various MHP parameters such as decay rates and processing cycle times. Second, each QN server is defined with processing logics to perform certain procedural operations. Third, QN-MHP uses an NGOMSL-style method (Kieras, 1988) as the tool for procedural task analysis. The resulting approach is thus called QN-MHP (Figure 3).

The QN cognitive architecture represents the human cognition system as a queueing network based on several similarities between them. First, ample research evidence has shown that major brain areas with certain information processing functions are localized and connected with each other in the brain cortex via neural pathways (see, e.g., Bear, Connors, & Paradiso, 2001), which is highly
similar to a queueing network of servers (information-processing units) that can process entities (to-be-processed pieces of information) traveling through the routes serially or/and in parallel depending on specific network arrangements. Therefore, brain regions with similar functions can be represented as servers and neural pathways connecting them as routes in the queueing network. Second, for

FIGURE 3 The general structure of the queueing network cognitive architecture and approximate mapping of its servers onto the human brain (Liu, Feyen, & Tsimhoni, 2006; Wu & Liu, 2008).
different tasks and learning stages, the to-be-processed information sometimes is
processed by the related brain regions (servers) immediately; sometimes informa-
tion has to be maintained in certain regions, to wait while some other information
is being processed (see, e.g., Rieke, Warland, van Steveninck, & Bialek, 1997). Hence, these to-be-processed pieces of information are represented as entities in
the queueing network and entities are processed in the network through certain
queueing processes, which represent the waiting and servicing processes of the
information entities.

QN-MHP consists of three subnetworks: perceptual, cognitive, and motor sub-
networks as summarized in the following sections (see Liu, Feyen, & Tsimhoni,
2006; Wu & Liu, 2008, for details).

**Perceptual subnetwork.** The perceptual subnetwork includes a visual and
an auditory perceptual subnetwork, each of which is composed of four servers. In
the visual perceptual subnetwork, visual entities enter the network at Server 1,
representing the eye, the lateral geniculate nucleus, the superior colliculus, the
primary visual cortex, and the secondary visual cortex (Bear et al., 2001). Then,
these entities are transmitted in parallel visual pathways—the parvocellular
stream (represented by Server 2) and the magnocellular stream (Server 3) where
the object content features (e.g., color, shape, labeling, etc.) and location features
(e.g., spatial coordinates, speed, etc.) are processed. The distributed parallel area
(represented by Server 4)—including the neuron connections between V3 and V4
as well as V4 and V5, the superior frontal sulcus, and the inferior frontal gyrus—
integrate the information of these features from the two visual pathways and gen-
erate integrated perception of the objects. These brain regions also serve as the
visual sensory memory storage for the visual information. Based on the mecha-
nism of working memory of Baddeley (1992) and the functions of these brain
regions (Ohbayashi, Ohki, & Miyashita, 2003; Smith & Jonides, 1998), the majority
of the graphical visual information is transmitted to the left and right hemisphere
posterior parietal cortex (Server B and Server A) (Bear et al., 2001; Smith &

The auditory perceptual subnetwork also contains four servers: The middle
and inner ear (represented by Server 5) transmit auditory information to the par-
allel auditory pathways, including the neuron pathway from the ventral cochlear
nucleus to the superior olivary complex (represented by Server 7) and the neuron
pathway from the dorsal and ventral cochlear nuclei to the inferior colliculus
(Server 6) where location, pattern and other aspects of the sound are processed
(Bear et al., 2001). The auditory information in the auditory pathways is inte-
grated at the primary auditory cortex and the planum temporale (represented by
Server 8; (Mustovic et al., 2003). These brain regions also serve as the auditory
sensory memory storage place for the auditory information (Mustovic et al.,
2003). Based on the mechanism of working memory of Baddeley (1992) and func-
tions of multimodal areas (neuron pathways between the primary auditory cortex
and the posterior parietal cortex, and the angular and supramarginal gyri), the
auditory information is transmitted to the left-hemisphere posterior parietal cor-
tex (Server B) as well as the right-hemisphere posterior parietal cortex (Server A).
Cognitive subnetwork. The cognitive subnetwork includes a working memory system, a goal execution system, a long-term memory system, and a complex cognitive processing system. Following Baddeley’s working memory model, QN-MHP contains four components in the working memory system: a visuospatial sketchpad (Server A), representing the right-hemisphere posterior parietal cortex; a phonological loop (Server B), standing for the left-hemisphere posterior parietal cortex; a central executor (Server C), representing the dorsolateral prefrontal cortices, the anterior-dorsal prefrontal cortices, the right ventral frontal cortex, and the middle frontal gyrus; and a performance monitor (Server E), standing for the anterior cingulate cortex. The visuospatial sketchpad and the phonological loop store and maintain visuospatial and phonological information in working memory (Smith & Jonides, 1998). The brain regions represented by Server C play a crucial role in suppressing the automatic responses (Burle, Vidal, Tandonnet, & Hasbroucq, 2004; Smith & Jonides, 1998) and categorization of information (Grossman et al., 2002; Shafritz, Kartheiser, & Belger, 2005). The anterior cingulate cortex (Server E) is responsible for performance monitoring and error detection (Smith & Jonides, 1998). The goal execution system (Server G) represents the orbitofrontal region and the amygdala complex which are typically involved in goal initiation and motivation (Rolls, 2000).

The long-term memory system represents two types of long-term memory in the human brain: (a) Declarative (facts and events) and spatial memory (Server H), standing for the medial temporal lobe including the hippocampus and the diencephalons. These brain areas store various kinds of production rules in choice reaction, long-term spatial information, perceptual judgment, decision making, and problem solving. (b) Nondeclarative memory (procedural memory and motor program; Server D), representing the striatal and the cerebellar systems, which store all of the steps in task procedure and motor programs related to motor execution (Bear et al., 2001).

Based on Byrne and Anderson (2001)’s experimental finding that humans cannot perform two arithmetic operations at once, the complex cognitive processing system (Server F) is assumed to perform complex cognitive functions in a serial manner. Such functions include multiple-choice decision, phonological judgment, anticipation of stimuli in simple reaction task, spatial working memory operations, visuomotor choices, and mental calculation (excluding the functions of Server C, e.g., information categorization and suppressing the automatic responses). Correspondingly, Server F represents brain areas responsible for these functions and these areas include the intraparietal sulcus, the superior frontal gyrus, the inferior frontal gyrus, the inferior parietal cortex and the ventrolateral frontal cortex, the intraparietal sulcus, and the superior parietal gyrus (Fletcher & Henson, 2001; Smith & Jonides, 1998).

Motor subnetwork. The motor subnetwork includes five servers corresponding to the major brain areas in retrieval, assembling, and execution of motor commands as well as sensory information feedback. First, Server V represents the premotor cortex in Brodmann Area 6, which plays an important role in sensorimotor and sensory cue detection especially in a single reaction time task (see, e.g.,
Roland, 1993). Second, the basal ganglia (Server W) retrieve motor programs and long-term procedural information from long term procedural memory (Server D; Bear et al., 2001). Third, the supplementary motor area and the pre-SMA (Server Y) have the major function of assembling motor programs and ensuring movement accuracy (Gordon & Soechting, 1995). Fourth, the function of primary motor cortex (Server Z) is to address the spinal and bulbar motorneurons and transmit the neural signals to different body parts as motor actuators (mouth, left and right hand, left and right foot server etc., Roland, 1993). Fifth, S1 (the somosensory cortex, Server X) collects motor information of efference copies from the primary motor cortex (Server Z) and sensory information from body parts and then relays them to the prefrontal cortex (Server C) as well as the SMA (Server Y; Roland, 1993).

QN-MHP allows mathematical analysis and real time simulation of human performance and has been successfully used to generate and model human performance and mental workload in real time, including driver performance (Liu et al., 2006) and driver workload (Wu & Liu, 2007b), transcription typing (Wu & Liu, 2008b), the psychological refractory period (Wu & Liu, 2008b), visual search (Lim & Liu, in press; Lim, Tsimhoni, & Liu, 2008a, 2008b), visual manual tracking performance and mental workload measured by NASA-TLX subjective workload (Wu & Liu, 2007b) and the event-related potential techniques (Wu, Liu, & Walsh, 2008). A simulation software with an easy-to-use graphical interface has been developed to implement the model with which a modeler only needs to select and click menu buttons with no need to learn a programming language (Wu and Liu, 2007a), as shown in Figure 4, which is a snapshot of the graphical modeling interface for QN-MHP.

An example of QN-MHP’s application is shown through a QN-MHP driving model that was developed and interfaced with a DriveSafety driving

**FIGURE 4** A screenshot of the current QN-MHP graphical interface for defining a dual task (Vehicle steering as Task 1 and another task as Task 2, both specified graphically by a modeler, with no need to learn a programming language.
simulator at the UofM via a TCP/IP host, as shown in Figure 5. The QN-MHP driving model steers the simulator in real time while performing a secondary task of map reading (Figure 6). Its performance is comparable to human drivers (Tsimhoni and Liu, 2003a, 2003b). For details, see Liu et al. (2006) and Wu and Liu (2007b). A corresponding multimodal adaptive workload management system has also been developed that estimates driver workload depending on the road condition and driver characteristics and then dynamically controls the rate of messages presented to the drivers. The system was experimentally tested with the University of Michigan Transportation Institute driving simulator (Wu, Tsimhoni, & Liu, 2008).
2. MULTITASK MODELING: SOME UNIQUE FEATURES OF THE QUEUING NETWORK COGNITIVE ARCHITECTURE

The QN architecture demonstrates its unique theoretical position and features in its multitask modeling. First, as the first word of the name of the queueing network architecture indicates, “queuing” is a unique and central theoretical concept in our work as a task coordination mechanism. Queuing (“waiting for service”) at the various servers while task entities (represented as multiple streams of customers) compete for service allows multitask interference and performance patterns to emerge without the need for any interactive executive process (Liu, 1996, 1997; Liu et al., 2006). Like cars and trucks traveling on the same highway, much of the network behavior predictions depends on characteristics of the network architecture and traffic flows. Like ACT–R and EPIC, a modeler still needs to analyze each concurrent task; but unlike ACT–R and EPIC, one does not need to write codes to interleave or interactively control the tasks.

The second unique theoretical position and a corresponding feature of the queueing network architecture compared with other cognitive architectures is its hybrid cognitive network structure with both serial and parallel information processing components in its cognitive subnetwork. Based on Byrne and Anderson’s (2001) finding that humans cannot perform two arithmetic tasks at once, QN-MHP assumes that Server F is serial while other cognitive servers are parallel. As pointed out in Liu et al. (2006),

from the theoretical point of view, although Byrne and Anderson’s experiment raises serious doubts about human ability to perform two complex cognitive tasks at once, it does not logically imply that human cognitive system cannot perform any two or more activities at once at all. . . . It is logically possible that the separate servers in the “cognitive network” work concurrently, but certain servers within this network (such as the server responsible for complex cognitive functions such as mental arithmetic) can only process one ‘rule’ at a time. (Liu et al., 2006, pp. 44–45)

Our PRP modeling work with closed-form mathematical equations offers a strong support to this theoretical position, because the modeling results would not appear and the mathematical equations would break down if we did not assume F to be serial or if we assigned serial processing to any other servers (Wu & Liu, 2008a).

Third, the overall mathematical structure of the queueing network model is also a unique property of the queueing network model. Although ACT–R and EPIC use mathematics in analyzing some of the specific mechanisms or operations of their models (e.g., the algebraic formulation of the PRP in EPIC and the spreading activation mechanism and the set of 20 fundamental equations in ACT–R), they lack mathematical frameworks for representing their overall “architectures”—that is, representing the interconnected arrangements of all the perceptual, cognitive, and motor components and their interaction patterns in one coherent mathematical structure. These models are symbolic models (Card & Newell, 1990), not mathematical models. All their modeling work, therefore, are performed with simulations. In contrast, the queueing network mathematical structure enables
QN-MHP to mathematically quantify the interactions among the servers and derive and predict a wide range of PRP behavior patterns and general trends without the need for simulation. In the PRP modeling work of Wu and Liu (2008a), all of the empirical PRP data are modeled with closed-form mathematical equations, without any reliance on simulation.

Fourth, in addition to an underlying mathematical structure and the computational power to generate/simulate behavior in real time, another unique feature of the QN model is its visual representation of the real-time operation of the mental network architecture. For example, it allows a modeler to visualize the internal information flows inside “the mind” whereas the “human” performs real world simple or complex tasks such as the PRP or a driving task. Subjective and physiological mental workload is mathematically analyzed and visually displayed as QN server and subnetwork utilizations (Liu et al., 2006; Wu & Liu, 2007b, 2008b).

3. QN-ACES FOR MULTITASK MODELING: OBJECTIVES, RATIONALE, AND RESEARCH ISSUES

A major limitation of QN-MHP is its inability to generate or model complex cognition such as language comprehension or problem solving that requires the creation of new rules by the model itself rather than only using rules preprogrammed by a model developer. Fortunately, this is exactly the unique strength of some of the production-system based architectures, and incorporating the complex cognition modeling capabilities of symbolic models in the QN architecture is one of the reasons for developing QN-ACES. Another major reason for research on QN-ACES is to further examine some of the unresolved theoretical issues in cognitive architecture, as discussed in detail next in this section.

On the basis of the research accomplishments and experiences developed in the first three steps previously summarized, developing QN-ACES represents another major step along this line of research on theoretical and methodological unification in cognitive and HCI modeling (Liu, 2006). Research on QN-ACES will help further clarify some important theoretical issues in multitask modeling as well as enrich the cognitive modeling tool base. Therefore, the specific objectives of developing QN-ACES are to establish a bridge between QN and ACES, examine some important research questions described next, and take another major step forward toward theoretical and methodological unification in cognitive modeling. In the recent book reporting and reflecting upon the findings of the Agent-based Modeling and Behavior Representation (AMBR) project, Pew and Gluck (2005) stated that “it is premature to settle. . . . There will be opportunities for cross-fertilization from one approach to another and even the potential for aggregation across architectures” (p. 412). The integration effort of QN-ACES is perfectly consistent with this statement and with other related earlier work, such as ACT-R/PM (adapting EPIC’s Perception and Motor work for ACT–R), and EPIC-Soar and EASE (discussed next).

One question that may arise is why ACES, not others? Why not more? Why not less? For example, 19 architectures are listed in Table 1.1 of the AMBR book.
Queueing Network Cognitive Architecture

(Gluck & Pew, 2005, p. 5). The answer is at least five folds. First, science and integration work can only proceed step by step; we need to start somewhere, but not everywhere; linking with ACES is already a major challenge. Second, all the six architectures included in QN-ACES (QN, QN-MHP, ACT–R, CAPS, EPIC, Soar) have been published in the most prestigious journals in the field (Psychological Review, Artificial Intelligence, ACM Transactions) and have undergone the most rigorous peer review process—an important factor to consider in integration. Third, each of the four architectures in ACES brings something very unique and crucial to QN-ACES, as discussed in detail next, and their codes are all available free of charge. Fourth, some of the other existing architectures are important general platforms for simulation, but not specific cognitive architectures by themselves (e.g., MicroSaint, D-Omar). We are seriously considering implementing QN in those platforms. Fifth, some other architectures are neural network connectionist models, which are very important but also beyond the scope of the current step. A future step of our integration endeavor is, in fact, to establish integral relations with those models but not in the current step.

In the following, I won’t attempt to describe the achievements of ACES. Their success and capabilities is one of the reasons for integrating QN with them in QN-ACES. Nor am I going to describe the similarities and shared views among all the architectures, such as the important role productions and symbolic processing play in cognition, the modular organization of the mind (although the partitions are not exactly the same across architectures), and the importance of 50 msec (as the level of timing analysis, per production cycle).

I would like to emphasize one word all architectures use: the word wait—one process may “wait” until another process completes or reaches a certain stage or state. This word may be mentioned in passing in others, but it plays a crucial and unique role in Queueing Network architecture. “Waiting” is Queuing. In a sense, “waiting” plays the role of the “invisible hand” in Adam Smith’s terminology. Behavior patterns in multitask performance emerges naturally (at least partially) when task entities all desire to minimize their own wait while traversing a network and competing for services with other task entities, just like economic behavior emerges naturally (at least partially) when economic agents all attempt to maximize their own (potentially related or conflicting) interests.

In the following I describe some of the specific research issues and approaches for developing and evaluating QN-ACES and their rationale. The first four sections (3.1–3.4) highlight the first set of bridges to be built between QN and ACES and the specific research questions to be addressed, including QN-ACT–R, which uses QN as a architectural platform to run concurrent ACT–R tasks and provides a unique method to explore some of the important unresolved questions in both ACT–R and cognitive architecture modeling. Another specific activity is to incorporate some of the unique modeling capabilities of ACES into QN-MHP to enhance the QN modeling capabilities, including ACT–R’s long-term memory, CAPS work on working memory, EPIC’s work on perceptual and motor systems, and Soar’s method of automatic subgoaling, chunking, and impasse resolution. Sections 3.5 and 3.6 discuss some of the empirical methods and databases with which QN-ACES and related research issues can be examined.
3.1. ACT–R Inspired (First Set of) Research and Development

As discussed in section 2 of this article, the QN architecture demonstrates its unique theoretical position in multitask modeling. Its focus is on investigating and representing the architecture of the mind as a whole, and the research philosophy is integration and unification rather than discrimination or placement of existing models, unless existing models are silent on related research issues, incapable of addressing them or in a state of debate and controversy among themselves.

As discussed earlier, “queuing as a task coordination mechanism” and “hybrid cognitive subnetwork” are two unique theoretical positions of the QN cognitive architecture. QN-MHP represents our first step in testing these two theoretical positions jointly in modeling procedural tasks and in bridging QN and symbolic models, but it is by no means the only approach to test them. In fact, because MHP and GOMS are not capable of modeling complex cognition, QN-MHP is incapable of modeling complex cognition and can not be used to test the two theoretical positions in complex cognitive task domains.

One way to test the QN theoretical positions in complex cognition is through integration and synthesis of QN and ACT–R modeling methods, which can be called QN-ACT–R, for ease of communication, as an important component of QN-ACES research and development. This QN-ACT–R modeling work can serve many theoretical and application purposes, capitalizing on the strengths and overcoming the limitations of QN and ACT–R alone. Using existing ACT–R findings and products, QN modeling capabilities can be significantly extended without reinventing the wheels (see next). Using QN, several important research questions in ACT–R modeling can be examined from the QN perspective, and let me mention three of them as examples: (a) QN coordination of concurrent ACT-R goals and tasks, (b) concurrency in cognition, and (c) buffer “jamming.” The last two correspond to the two serial bottlenecks in ACT–R (Anderson et al., 2004, p. 1038).

**QN coordination of concurrent ACT–R goals and tasks.** As discussed earlier, in the QN architecture, multitask performance emerges as the behavior of multiple streams of information flowing through a network, with no need to devise task specific procedure to interleave production rules into a serial program or for an executive process to interactively control task processes (Liu et al., 2006). Encouragingly, this QN position has been adopted recently by ACT–R modelers in their threaded cognition work (Salvucci & Taatgen, 2008), in which they maintain a serial procedural memory for task coordination. In fact, their threaded cognition work can be mathematically represented as a special type of QN with a serial bottleneck server. The recent development of threaded cognition and its adoption of the QN position of less reliance on executive processes, plus that fact that ACT–R itself has been moving in the direction of becoming more “network-like” with more modules and less central control (e.g., the replacement of a goal stack with multiple buffers), are all encouraging signs that ACT-R is in fact, step by step, moving closer to the QN architectural positions.

In QN-ACT–R, we can explore the use of the QN multitask modeling mechanism as a new method for running multiple concurrent ACT–R goals and tasks. In
other words, all the rules of the related single tasks will be ACT–R based, but when multiple ACT–R single tasks run concurrently, their behavior and potential conflicts will be modeled with the QN architecture—a general QN architecture with “queuing” and a “hybrid cognitive subnetwork” as task coordination mechanisms, in which the procedural memory is not assumed to be serial or acts as a serial bottleneck in contrast (or in comparison) to the special QN case of the threaded-cognition, which was unable to model some complex multitask data, as reported in Salvucci and Taatgen (2008).

**Concurrency in cognition.** As discussed previously, ACT–R assumes serial cognition, whereas QN-MHP and QN-ACES assume parallel cognition with a potential serial server within cognition. A closer comparison between the two positions also suggests that the difference may be only or partially a matter of the level of granularity. ACT–R’s serial cognition is defined as “A SINGLE production is selected at each cycle to fire” (Anderson et al., 2005, p. 1038). One question to ask is, What defines or is allowed in a single production? If a single production contains two or more conjunctive components (linked with “AND”) in its rule statement, then obviously two or more activities would occur with one firing of a single production. Let me use the following typical and representative ACT–R rule as an example (from ACT-R (2006), ACT–R 6.0 Tutorial Unit One, p. 34; with emphasis added):

**ADD-ONES**

IF the goal is to add a pair of numbers **AND** you are busy waiting for the answer for the ones digit **AND** the sum of the ones digits has been retrieved

THEN **STORE** the sum as the ones answer **AND** **NOTE** that you are busy . . . **AND** **REQUEST** a retrieval to determine if . . .

Clearly this single production rule entails multiple concurrent actions within cognition. Looking at the production as a whole, it is serial (only a single production to fire), but as a network of servers, it is parallel (several QN servers are working simultaneously to accomplish the multiple actions specified by this single production). This concurrency is also revealed in the trace of running an ACT–R model (in the ACT–R Listener Window), particularly if one sets the trace detail to high (:trace-detail set to high). A typical example can be found on pages 5 to 6 of ACT–R 6.0 Tutorial Unit EXPERIMENT, which shows usually three to six activities occurring at any particular 50 msec cycle interval of trace. In a sense, QN-MHP and QN-ACES can be used to display graphically in real time where those multiple activities occur in a QN network for each cycle.

**Buffer “jamming” issue.** Since Version 5.0, ACT–R’s goal stack has been replaced with multiple buffers; in this sense, ACT–R has become more distributed and network-like than earlier versions. “One of the visual buffers, associated with the dorsal ‘where’ path of the visual system, keeps track of locations, while the other, associated with the ventral ‘what’ system, keeps track of visual objects and their identity” (Anderson et al., 2004, p. 1038). These two ACT-R buffers perform
the same functions as Servers 2 and 3 in Figure 2 (Feyen & Liu, 2001; Liu et al., 2006). A major difference between QN and ACT–R and an open research question is the capacity of the buffers/servers and whether queueing is allowed. ACT–R regards buffers as the second serial bottleneck—at most one item can be held in a buffer (Anderson et al., 2004, p. 1038). In ACT-R modeling, this serial buffer assumption requires the modeler to “query the state in every production that makes a request that could potentially jam a module” (ACT–R Tutorial Unit 2, p. 8). Potential causes for jamming the various types of buffers (manual, vocal, perceptual, retrieval style, and goal style) are discussed in detail in ACT-R Tutorial Unit 7.

Just like the serial cognition issue that has been at the center of debate in cognitive psychology and HCI modeling for a long time, this serial buffer assumption needs to be examined as well, from both theoretical and methodological standpoints. What are the consequences (theoretically and methodologically) of allowing buffer capacity to be greater than 1 and/or queueing (waiting to enter a busy module, rather than jamming it) as in QN? This is a question that can be explored through QN-ACES simulations, after importing and integrating related ACT-R concepts into QN-ACES.

In parallel with the three QN-ACT–R research issues previously illustrated, an ACT–R-inspired QN-MHP model enhancement can be explored by adopting the ACT–R’s declarative memory system. In QN-MHP, long-term declarative memory is coded and represented as entries in Excel data arrays preprogrammed by QN-MHP modeler and accessed through Server H with MHP timing parameters. This simple representation is adequate for approximating certain procedural task situations but definitely has many limitations. It would be interesting to explore the effects of replacing the long-term declarative memory of QN-MHP with the declarative memory mechanisms of ACT–R, which has an attribute-value memory structure and a suite of well-developed and validated subsymbolic mechanisms that determine the availability of symbolic memory elements and the time needed to retrieve them (Anderson & Lebiere, 1998; Anderson et al., 2004). Long-term declarative memory will continue to be accessed through Server H, but the availability of items and the time needed to retrieve them will be determined by the ACT–R mechanism. As the first step, ACT–R’s base-level learning mechanism specified with the activation equation can be implemented.

3.2. CAPS Inspired (First Set of) Research and Development

The CAPS line of architectures (CAPS, 3CAPS, now 4CAPS; all are called CAPS in this report for ease of description) is a hybrid of a production system and an activation-based connectionist system. CAPS reflects the theory of capacity constrained comprehension, which states that cognitive capacity constrains comprehension and it constrains comprehension more for some people than for others (Just & Carpenter, 1992; Just & Varma, 2006). A central thesis of the theory is that the nature of a person’s language comprehension depends on his or her working memory capacity. Capacity is defined in this theory as the maximum amount of activation available in working memory to support either the storage or the
computation function of the working memory. This theory has been applied successfully to model a variety of language comprehension phenomena, particularly systematic individual differences which indicate that individuals vary in the capacity they have for meeting the computational and storage demands of language processing. According to this theory, both the time course and the content of language processing depend on working memory capacity: When task demands are high, processing will slow down (because of an increase in the number of cycles required to bring an element to threshold) and some partial results may be forgotten. CAPS and 3CAPS successfully account for behavioral measures of high-level cognition and for individual differences. 4CAPS extends to neuroimaging measures and new populations.

CAPS has the view that thinking is a network phenomenon—a view shared by our queueing network approach. One component of CAPS is a parser called Capacity Constrained Reader for processing successive words of a text during reading. As part of our effort of developing QN-ACES, we have been examining closely the CAPS architecture and its software implementation, accessible through their website, particularly for our specification of the QN-ACES working memory servers, adaption of Capacity Constrained Reader to enable QN-ACES to process language information, and exploration of methods to account for individual differences in cognition. The data and CAPS modeling work on the five capacity-related issues discussed in detail in Just and Carpenter (1992) is an excellent platform for testing QN-ACES after it incorporates the related CAPS concepts and methods. It would also be interesting to model text reading and comprehension while driving a simulator, as discussed in section 3.6.

3.3. EPIC Inspired (First Set of) Research and Development

EPIC and QN share the same view that cognition is parallel. The current theoretical and methodological difference is whether behavior coordination requires a central executive to perform detailed interactive control or this coordination/control is local at the servers and distributed in a network (with “waiting” playing the invisible hand, see previously). Our modeling work of a wide variety of procedural tasks demonstrates that the QN position is a viable alternative approach to EPICs, as previously discussed and in our numerous related publications. However, an open question is whether the QN theory and method can hold for complex cognition. This research question can be examined with QN-ACES, by evaluating the applicability of the QN to complex cognition tasks to see whether the need for central interactive control would arise.

Among all the cognitive architectures, EPIC pioneered the efforts in integrating cognition with perception and action, as reflected in the developments of ACT–R/PM (now ACT–R 6.0) and Soar-EPIC and EASE, all of which reimplemented or adapted many aspects of EPIC’s perceptual and motor systems in their developments.

Kieras and Meyer (2005) presented some fascinating findings in their recent EPIC modeling work on the perceptual and motor systems that challenge some of the conventional wisdoms. For example, they showed that the motor feature
programming effects in eye movements and aimed manual movements are most likely a result of translations due to poor Stimulus-Response compatibility, which is actually a Hick’s Law effect. Guided eye movements and aimed manual movements have high Stimulus-Response compatibility, and therefore there is no justification in keeping motor feature programming for these tasks (in fact, it slows down the motor processor and causes poor data fitting)—suggesting an architectural modification. Kieras and Meyer pointed out that it is still not clear whether this finding holds for other manual movements such as keypress. They also proposed a hypothesis about the primacy of spatiality in perceptual and motor relations with important architectural implications: perceptual-motor system is organized around an integrated spatial system that transforms visual coordinates to motor movements subcognitively and rapidly.

The development of QN-ACES will pay close attention to these EPIC developments and adopt them when they are ready (many are still being carefully evaluated by EPIC modelers). On these perceptual and motor issues, QN appears to be in full agreement with EPIC and will incorporate their findings into QN-ACES. The primary goal of QN-ACES is integration, “synthesizing what we know” (Newell, 1990). Debates occur only when disagreements arise.

### 3.4. Soar Inspired (First Set of) Research and Development

QN-MHP has been successfully applied to model a wide variety of procedural tasks as discussed previously and in our publications. Its ability to perform real-world tasks is both enabled and limited by its sole use of a procedural GOMS-style method to define the processing functions of all the servers. Specifically, QN-MHP can use general, task independent procedural rules provided by the QN-MHP developer and additional task specific procedural rules provided by a modeler to model a specific task domain. However, QN-MHP cannot generate or create new rules by itself, which is a crucial characteristic of intelligent behavior or complex cognition.

In this regard, we find the related Soar mechanisms particularly appealing. A Soar model can learn or create new rules by itself (called chunks) in addition to using those written by the modeler, and the learning process that creates the chunks are called chunking (Laird et al., 1987; Lehman, Laird, & Rosenbloom, 2006). In Soar, behavior is viewed and modeled as movement through a problem space (Newell & Simon, 1972). More specifically, a particular behavior is movement along a path in a problem space from an initial state (initial description of the situation) to a desired goal state or states via the application of operators. At any time, only one state exists, which is called the current state. The application of an operator transforms the current state by changing some of its features and values. The decision cycle is the processing component that selects the next operator to apply to generate behavior based on the content in the long term and working memories. Processing is in “impasse” when a selection cannot be made between two or more operators. Soar automatically creates a subgoal or substate to resolve the impasse, which represents the information relevant to resolving the impasse. Soar automatically forms a new rule whenever results are generated from an
impasse. This chunking process is deductive learning from prior knowledge. The new rule is composed from a “then” part containing the deduction and an “if” part containing the knowledge that contributed to the deduction. An impasse is the Soar’s way of signaling a lack of knowledge and indicating an opportunity for learning. Soar has defined a full set of impasses that are fixed and domain independent (Lehman et al., 2006). It is important to note that there are some important distinctions between operators in Soar and rules in regular rule-based systems. In regular rule-based systems, individual rules are the operators. In Soar, the precondition and the action components of an operator are implemented as separate rules. This separation allows Soar agent to recognize and select options, requiring fewer rules than regular rule-based systems, and potentially empowering Soar with greater flexibility in dealing with uncertainty (Soar Technologies, 2002).

We have started to examine the implementation details of these Soar mechanisms. We would like to reuse/adopt/adapt as much of the existing method and code as possible, and will modify them when necessary. Our current tentative approach is to let QN-ACES access and implement related functions through three servers: Servers E (“performance monitor” for recognizing impasse), G (goal initiation), and D (long-term procedural memory).

This work will not only give QN-ACES the “rule creation” problem solving capability, but also offer important insight into several research questions, both in general and in the context of Soar in particular. An important open research question is the level of cognitive parallelism and how to represent and model it. Soar only allows one operator at a time (per decision cycle) and is serial at the operator level. Thus, in general, Soar would not allow “move-hand” and “calculate” to occur simultaneously as two operators per decision cycle. Soar does allow the two actions to occur simultaneously if and only if the two operators are prepackaged into one operator. In other words, Soar supports limited parallelism by allowing prepackaged operators in which concurrent activities are prepackaged together into a single operator. In contrast to Soar’s dynamism, flexibility, and power of modeling uncertainty as shown in its subgoaling and the separation of precondition and action (previously mentioned), my impression of this predefined and prepackaged operator approach is it is quite “static” and is a wide open question to address, both theoretically and methodologically. QN-ACES offers one avenue to examine this question by using the QN parallel cognition while adapting Soar’s chunking mechanisms (or its central ideas).

Another point to make, from the QN standpoint, is that an individual production rule in Soar often involves a number of psychological activities simultaneously, although they are represented and coded in one rule by a modeler. For example, the following are rule 1 (r1) and rule 6 (r6) from Lehman et al. (2006, p. 16), which are representative and typical of Soar rules (with emphasis added):

(r1) IF I am the pitcher, AND the other team is at bat, AND I perceive that I am at the mound
THEN suggest a goal to get the batter out via pitching

(r6) IF the throw-curve-ball operator has been selected
THEN send throw-curve to the motor system AND add pitch thrown to the state.
From the QN perspective, each rule involves multiple concurrent activities performed by distinct regions of the brain/mind or several servers of the QN-MHP/ACES. Specifically, (r1) requires multiple judgments or perception in the condition part; (r6) requires two cognitive activities in the action part. In this regard, QN explicitly specifies which servers perform these activities and visualizes them in simulation (Figure 4). Recall, in QN, all activities must occur at a particular psychological server; no action is left unspecified with regard to which server is responsible for doing it.

Another research question is the pros and cons of using a single problem space to represent a multitask situation, which is “solving multiple problems concurrently,” in a sense. As mentioned above, in Soar, behavior is movement through a problem space. Using a single problem space is a powerful approach to model single task situations (e.g., finding “Burger King” on an electronic map). However, it may have its limitations for multitask modeling. For example, representing “driving a car to a destination” and “finding a Burger King on an in-vehicle map” in one problem space can be cumbersome, because the need to “find Burger King” may come and go, is secondary to driving, has no intrinsic relationship to driving although it may interfere with driving when carried out together. Further, a third or fourth “unrelated” activity may occur at any moment (e.g., a fellow passenger suddenly starts a conversation or asks a question). Representing all these dynamic concurrent activities in one problem space naturally is certainly a challenge. The QN approach of multitask modeling does not require putting all these activities in one problem space. Comparing the pros and cons of each approach and finding ways to integrate the two approaches is part of this research on QN-ACES.

3.5. Testing QN-ACES with AMBR Data and Comparison With the Four AMBR Approaches

The first set of research and development work previously discussed can be tested and evaluated with many sources of existing data and related modeling work, one of which is the AMBR data and the findings of the four AMBR modeling teams. The AMBR project was sponsored by the U.S. Air Force Research Lab substantially and the Defense Modeling and Simulation Office and the Office of Naval Research in part. For a detailed description of the objectives, rationale, process, and results of the AMBR project, see Gluck and Pew (2005). I do not summarize them here, but to mention that AMBR had two tasks used for model evaluation: an air traffic control task and a category learning task. Both tasks were selected on the basis that they rely on multiple human systems—visual sensation and perception, memory, cognition, and action—and the interaction of such systems (e.g., eye–hand coordination), and both are multitask situations, representative of Defense missions. Among the four teams (or approaches) that participated in the AMBR project, EASE is the most relevant to the current project, for the following reasons: The ACT–R team used ACT–R 4.0 which was based on the goal stack structure (no longer exists in current ACT–R) and did not have the buffer systems of current ACT–R. Further, ACT–R team did not use the P/M
component (Lebiere, 2005). However, we closely examine their code, ideas, and research approach, to absorb/reuse/compare as much as there are relevant.

EASE was an extension of EPIC-Soar to include ACT–R’s base-level learning, incorporating both symbolic and subsymbolic representation and mechanisms (Chong & Wray, 2005). There are many components or related work in EASE that can be reused or modified for use (or to gain insight) in QN-ACES for modeling the two AMBR tasks, including EASE implementation of ACT–R’s base-level learning and the corresponding activation equation, its implementation of EPIC retinal zones, and its set of Soar operators. However, there are also major theoretical and methodological differences between the two approaches in multitask and workload modeling.

First, EASE, using the cognitive mechanisms of Soar, selects one operator at a time, in contrast to the parallel cognitive system of QN-ACES (see section 3.4). A comparison between the two approaches will offer insights on this issue, considering we are using the same AMBR tasks and a potentially large overlapping set of other assumptions and codes.

Second, EASE models subjective workload with workload values preassigned by EASE modeler for each kind of work. For example, “accept incoming blip” was assigned a workload value of 3 (Table 7.3, in Chong & Wray, 2005, p. 256). As previously discussed, workload in QN-MHP and QN-ACES represent dynamic changes in server utilization and congestion depending on a task situation at any instant and they are not preassigned values. We have successfully applied this method to model workload (subjective and physiological) in performing a variety of procedural tasks, and it is important to see whether this approach can be extended to AMBR complex cognition tasks and to compare the findings with those of the four AMBR teams.

Chong and Wray (2005) stated that “one lesson learned while developing the model was the importance of tools that make observable the model’s covert and overt behavior and its internal state” (p. 261). It is worth noting that this is exactly what QN-MHP and QN-ACES do—performance and workload are visualized in real time as entity travels and congestions in QN network (see Figure 4).

3.6. Evaluating QN-ACES With Driving Tasks Involving Language Comprehension, Paying Close Attention to Individual Differences

Drivers (and pilots) often need to perform simple procedural (e.g., push a familiar button) and complex cognitive tasks (e.g., language comprehension and communication, decision making, or recall facts) while performing a continuous steering (or flying) task at the same time, which involve many components itself (e.g., perceiving the road, steering, foot pedal action, etc.). In addition to the research and development efforts previously described, we plan to evaluate QN-ACES in its ability to model concurrent driving and language comprehension, which provides another link with CAPS work on capacity constrained language comprehension and the related ACT-R and Soar work on knowledge/strategy and language; further, it extends the language comprehension inquiry to the multitask domain. This work is also a natural and significant extension of our many years of driver modeling work into the complex cognition domain while driving.
A series of driving experiments can be conducted with the simulator at the University of Michigan Transportation Research Institute, in which human participants will be asked to perform a driving task while reading (and/or listening) text materials, comprehending them, and making appropriate responses. The same Reading Span task of Daneman and Carpenter (1980) can be used as a measure of participants’ working memory capacity for language. Some of the five specific issues addressed in Just and Carpenter (1992) can be examined in this new context. For example, we can use navigation information as text materials, involving “subject relative” or “object relative” sentences (e.g., “the car that just passed you . . .” vs. “the car that you just passed . . .”) and/or different levels of ambiguity. Participant performance can be analyzed and modeled to see whether the extensive set of findings of Just and Carpenter and their individual capacity differences account of language comprehension can be extended to this important multi-task driving domain. The modeling work can also be compared with ACT–R and Soar work on language comprehension, which put more emphasis on production and strategies, rather than on capacity. This comparison will offer insight into the relative role of capacity and strategy in multitasking and language comprehension.

In conclusion, on the basis of the accomplishments achieved and lessons learned in the three steps of our QN architectural development summarized in this article, QN-ACES represents another major step along the direction of theoretical and methodological unification in cognitive architecture development and HCI modeling. Research on QN-ACES will help enhance the knowledge and tool base in cognitive modeling and provide further insights on some important theoretical issues in cognitive architecture.

REFERENCES


