Reflections

Mathematical Models of Mental Structure Classified in terms of Discrete versus Continuous Transmission and Serial versus Network Architecture

from Liu [1996] "Queueing network modeling of elementary mental processes," Psychological Review, 103(1), pp. 116-136.

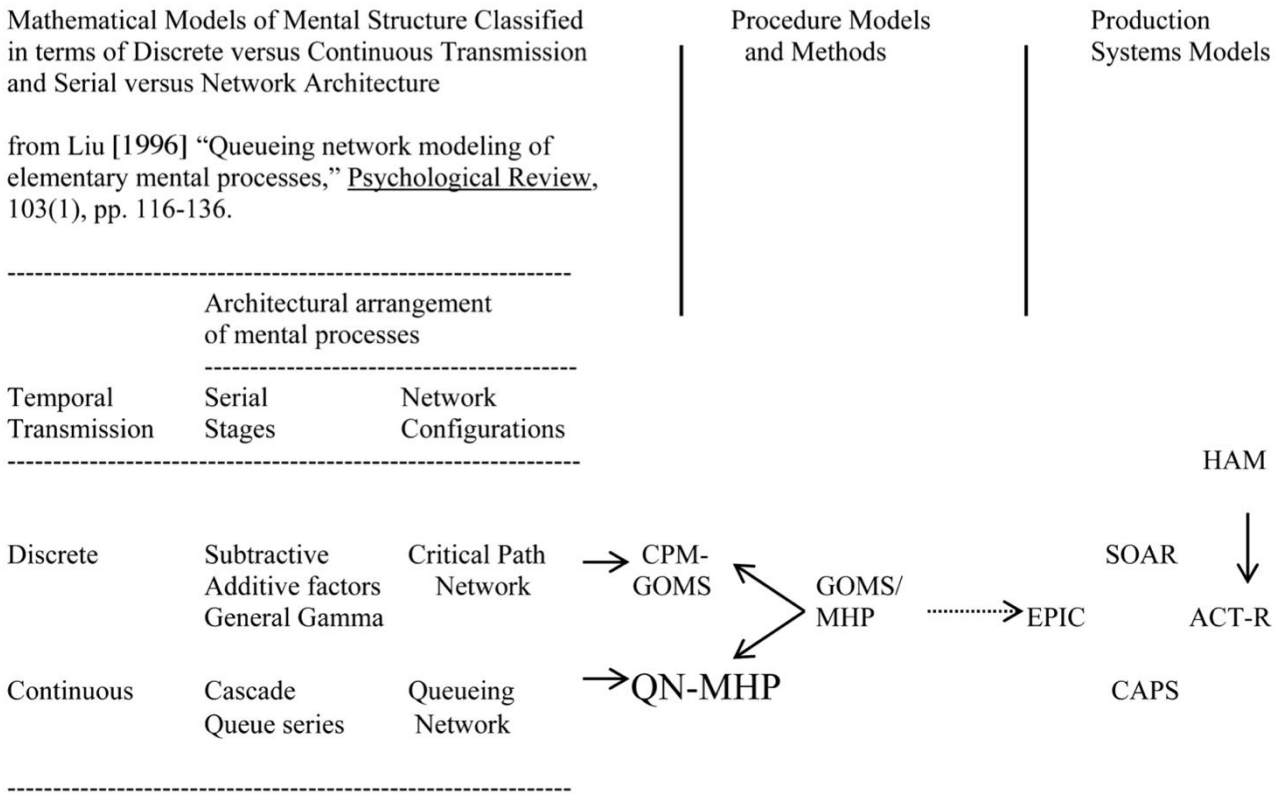


Fig. 1. Mathematical models of mental structure and procedure/production system models of cognitive architecture.

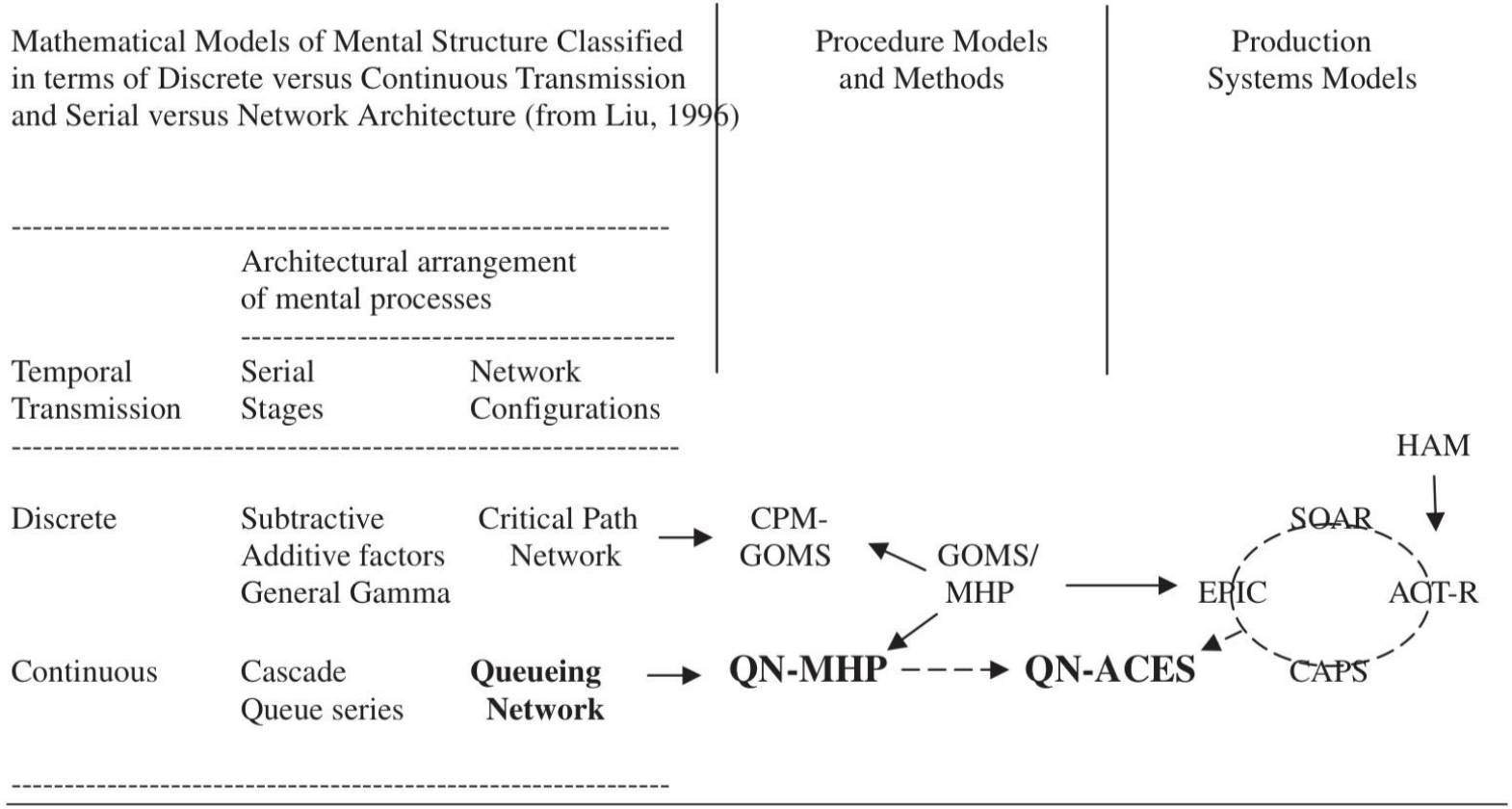


FIGURE 2 Mathematical and symbolic models of mental architecture (Liu, 2006; Liu et al, 2006) showing the relationship between QN, QN-MHP, QN-ACES and a sample of influential cognitive architectures. *Note*: By integrating the complementary schools of mathematical (left half of the figure) and symbolic (right half) models, QN-MHP supports both precise mathematical analysis and real-time generation of behavior, thus capitalizing on the strength and overcoming the weaknesses of either mathematical or symbolic modeling.

Queuing Network Cognitive Architecture

1. A hybrid network:

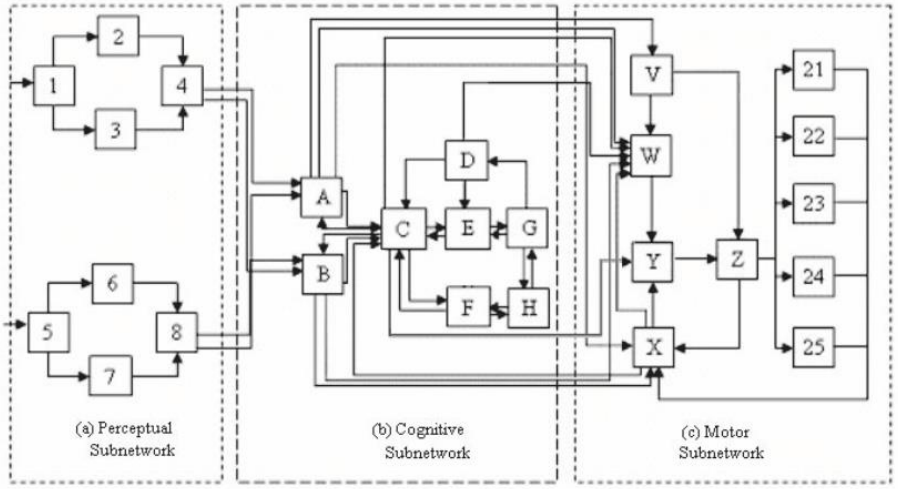
   (biologically-inspired, theory- and evidence- based)

   with serial and parallel components)

2. Analysis of both *queuing* and *processing*

   (queuing as a task coordination mechanism)

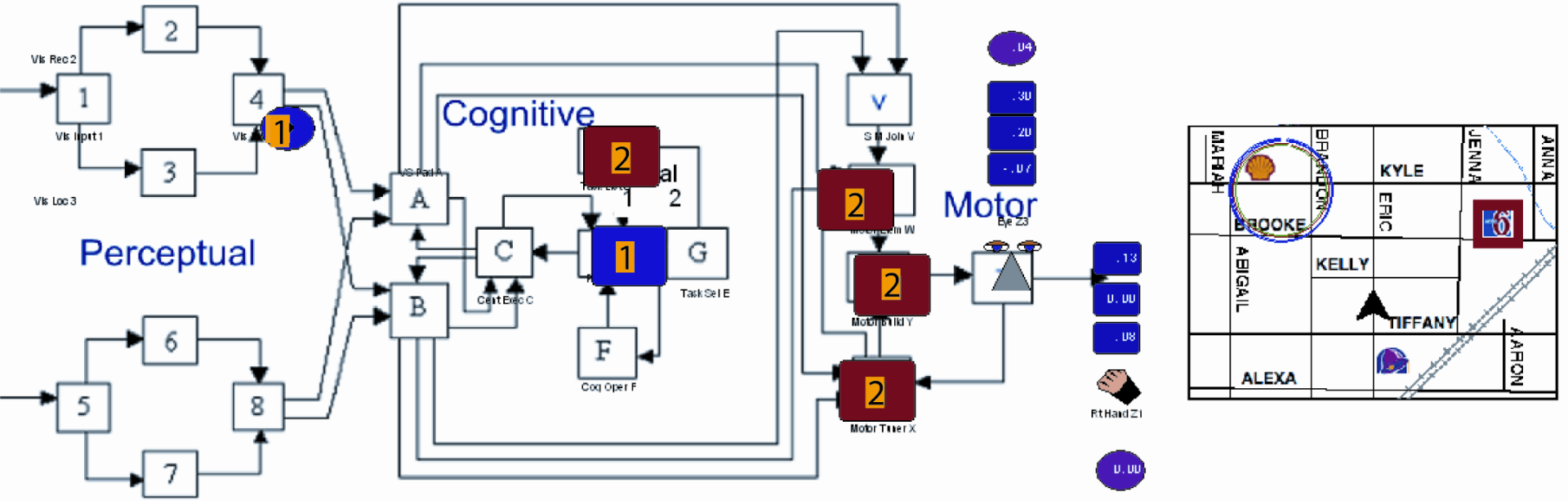
3. Real-time generation and mathematical analysis of behavior mechanisms beyond phenomena



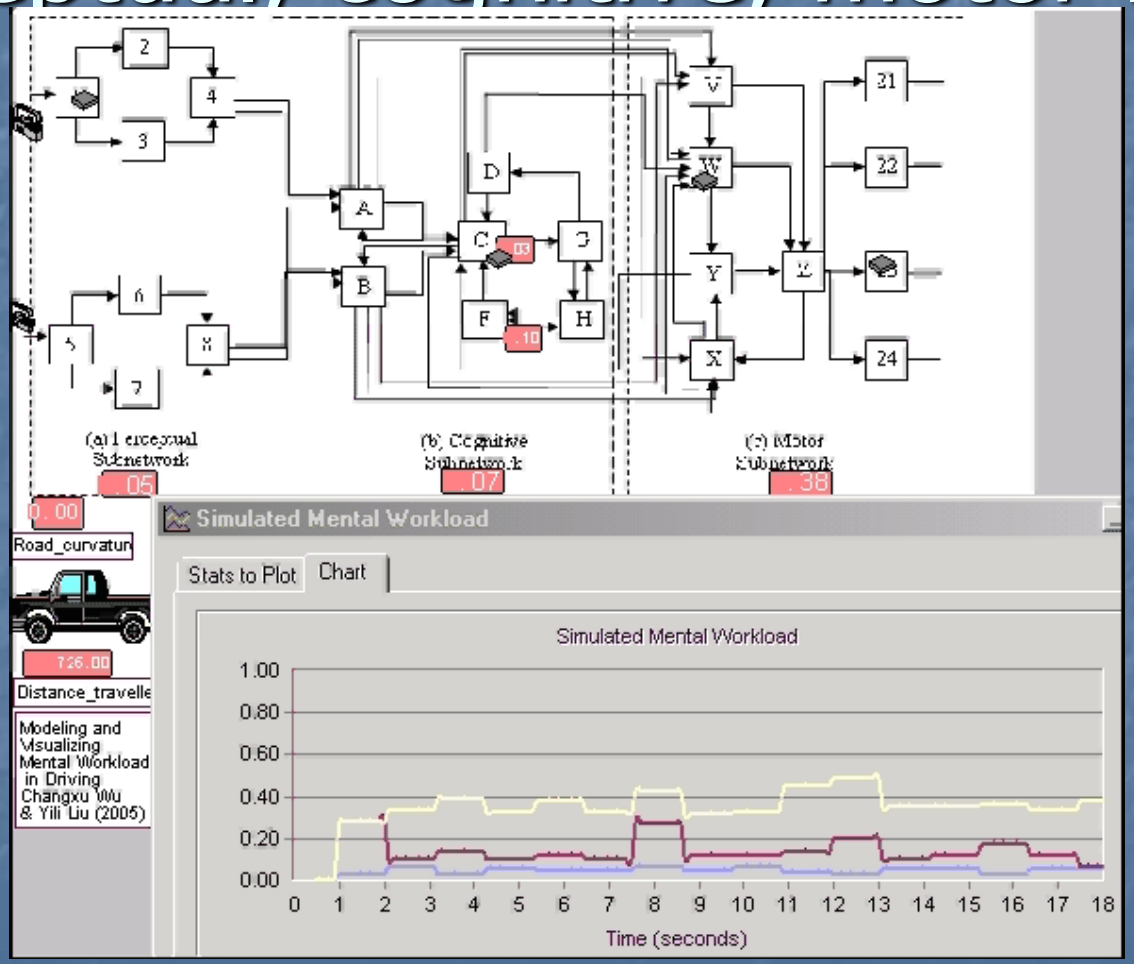
Perceptual Subnetwork	Cognitive Subnetwork	Motor Subnetwork
1. Common visual processing	A. Visuospatial sketchpad	V. Sensorimotor integration
2. Visual recognition	B. Phonological loop	W. Motor program retrieval
3. Visual location	C. Central executive	X. Feedback information collection
4. Visual recognition and location integration	D. Long-term procedural memory	Y. Motor program assembling and error detecting
5. Common auditory processing	E. Performance monitor	Z. Sending information to body parts
6. Auditory recognition	F. Complex cognitive function	21-25: Body parts: eye, mouth, left hand, right hand, foot
7. Auditory location	G. Goal initiation	
8. Auditory recognition and location integration	H. Long-term declarative & spatial memory	

Figure 3. The general structure of the queuing network-model human processor (further developed from Liu, Feyen, & Tsimhoni, 2006; Wu & Liu, 2007a).

Fig. 10. Screenshot of QN-MHP in action. A visual entity is about to be processed by server A as two concurrent tasks are being processed in the cognitive subnetwork. The eye is looking at the map but a steering action is still underway.

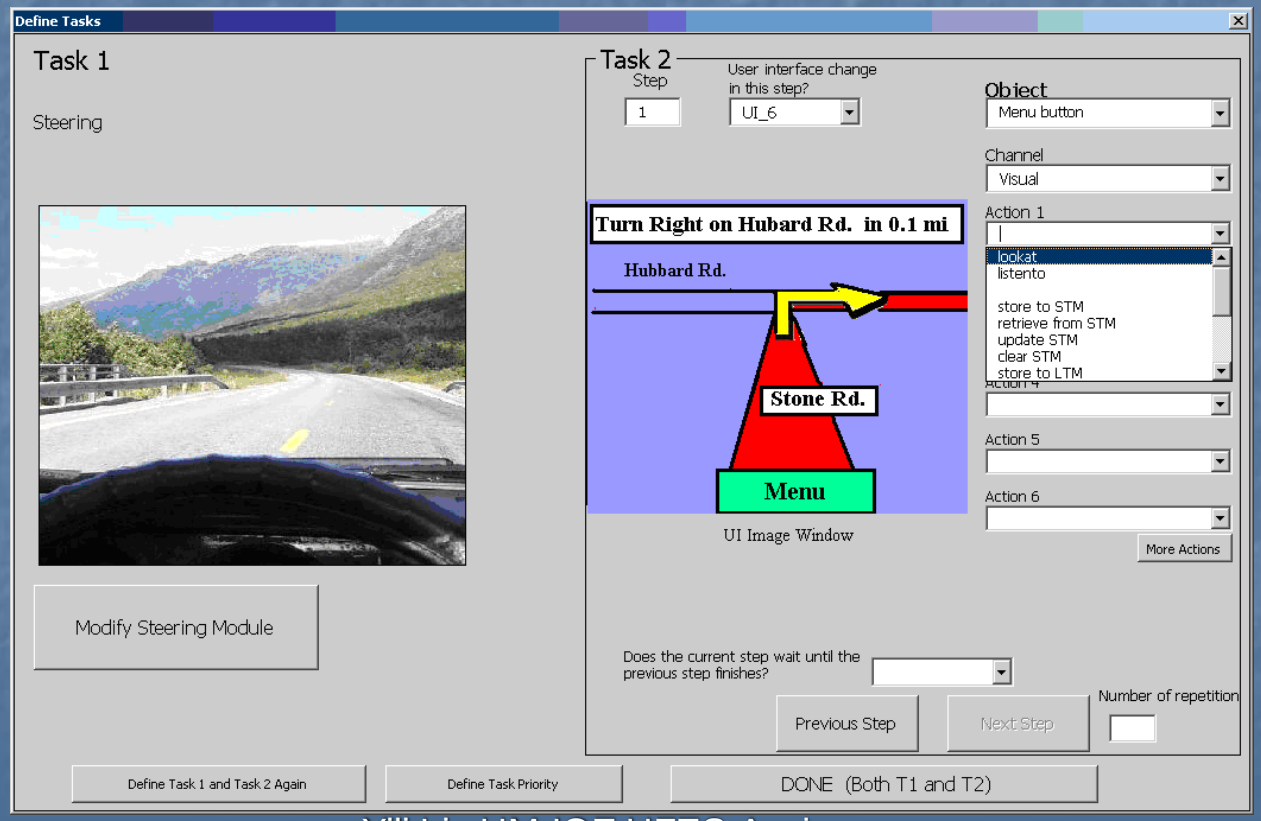


Visualizing Mental Workload (perceptual, cognitive, motor loads)



A screenshot of the current QN-MHP graphical interface for defining a dual task (vehicle steering as Task 1 and a secondary task as Task 2, both specified graphically by a modeler)

in Wu and Liu, 2007, *Ergonomics in Design*, Winter 2007 issue



Mathematical Models of Mental Structure Classified in terms of Discrete versus Continuous Transmission and Serial versus Network Architecture

(from Liu, 1996, "Queueing network modeling of elementary mental processes," Psychological Review, 103(1), pp. 116-136.

Temporal Transmission	Architectural arrangement of mental processes: Serial Stages	Network Configurations
Discrete	Subtractive Additive factors General Gamma	Critical Path Network
Continuous	Cascade Queue series	Queueing Network

Procedure Models and Methods

Production Systems

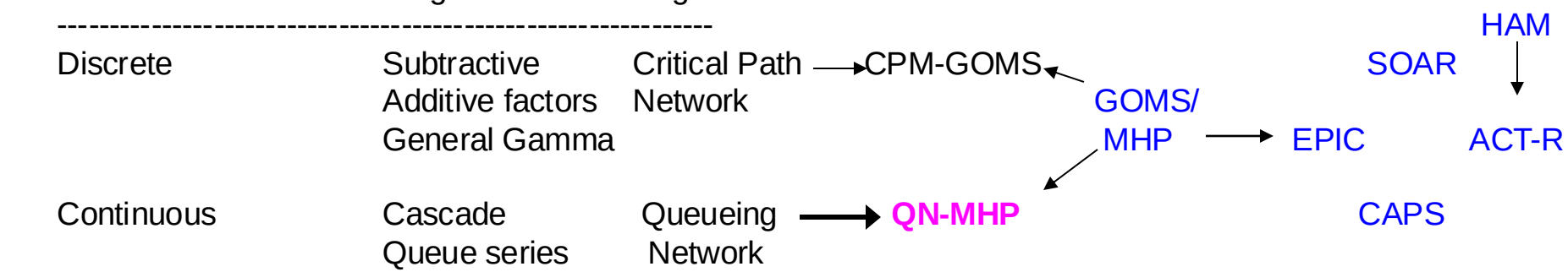


Figure 1: **Mathematical** (left side) and **Symbolic** (right side) models of mental architecture

Liu, Feyen, & Tsimhoni (2006) "QN-MHP: A computational architecture for multitask performance," ACM Transactions on Computer-Human Interaction, vol 13, pp. 37-70.

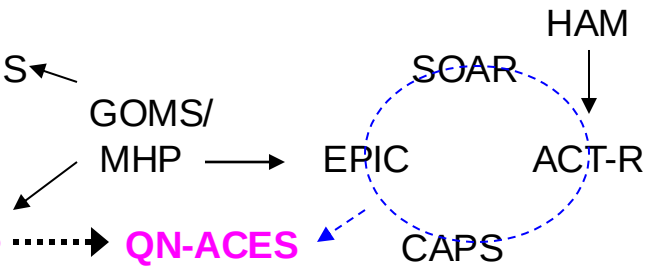
Mathematical Models of Mental Structure Classified in terms of Discrete versus Continuous Transmission and Serial versus Network Architecture

(from Liu, 1996, "Queueing network modeling of elementary mental processes," Psychological Review, 103(1), pp. 116-136.

Temporal Transmission	Architectural arrangement of mental processes: Serial Stages	Network Configurations
Discrete	Subtractive Additive factors General Gamma	Critical Path Network
Continuous	Cascade Queue series	Queueing Network

Procedure Models and Methods

Production Systems



QUEUEING NETWORK MODELING OF VISUAL SEARCH

by

Ji Hyoun Lim

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Industrial and Operations Engineering)
in The University of Michigan
2007

TABLE OF CONTENTS

LIST OF FIGURES.....vi
LIST OF TABLES.....ix
ABSTRACT.....x

CHAPTER 1
VISUAL SEARCH MODELS AND QUEUEING NETWORK ARCHITECTURE

1.1. Motivation.....1
1.2. Existing Computational Models of Visual Search.....2
1.2.1. Computational models specialized for visual search.....2
1.2.2. Visual search models based on unified cognitive architectures.....4
1.3. The QN-MHP: Computational Cognitive Model of Queueing Network.....7
1.4. Research Objective.....8

CHAPTER 2
CONVERSION OF VISUAL STIMULUS:
FROM NATURAL SCENES TO DATA ARRAYS

2.1. Introduction.....11
2.2. Computational Framework for Vision.....13
2.2.1. Basic vision and visual stimulus.....13
2.2.2. Existing computational frameworks of the early vision.....15
2.3. Representations of Visual Stimulus for Queueing Network Model.....17
2.4. Implementation of a Conversion System.....20
2.4.1. Size of a visual stimulus array.....20
2.4.2. Features for the early vision.....21
2.4.3. Data array for a moving visual stimulus.....23
2.5. Conclusion.....24

CHAPTER 3
QUEUING NETWORK MODELING OF VISUAL SEARCH AND MENU SELECTION

3. 1. Introduction.....25
3. 2. Existing Experimental and Modeling Efforts on Menu Search 27
3. 3. Queueing Network Modeling of Menu Search.....36
3. 3. 1. Queueing Network – Model Human Processor (QN-MHP).....37
3. 3. 2. Modeling menu search task with QN-MHP.....39
3. 4. Simulation Results and Discussions.....50
3. 4. 1. Eye overshooting.....51
3. 4. 2. Varying length in saccade.....54
3. 4. 3. Target position effect.....59
3. 5. Conclusion.....61

CHAPTER 4
REINFORCEMENT LEARNING IN EYE MOVEMENTS: MODELING THE INFLUENCES OF CYCLIC TOP-DOWN AND BOTTOM-UP PROCESSES

4. 1. Introduction.....64
4. 2. Relationship between Attention and Saccades.....66
4. 3. Computational Models of Reinforcement Learning.....69
4. 4. Eye Movements and Reinforcement Learning.....74
4. 5. Effect of Top-Down Processes on Eye Movements – Yarbus' Picture Viewing Task.....80
4. 6. Effect of Bottom-Up Processes on Eye Movements – Findlay's Visual Search Task.....85
4. 7. Discussion and Conclusion.....91

CHAPTER 5
EXPERIMENTAL STUDY:
ANALYSIS OF GLANCE BEHAVIOR IN PEDESTRIAN DETECTION
USING A NIGHT VISION ENHANCEMENT SYSTEM WHILE DRIVING

5.1. Introduction.....94

5.2. Study 1: Relationship between the Intensity of Clutter and the Subjective Rating of Clutter.....98

5.2.1. Method.....99

5.2.2. Clutter metrics for the visual stimulus.....100

5.2.3. Results.....103

5.2.4. Conclusion and discussion.....108

5.3. Study 2: Pedestrian Detection while Driving.....111

5.3.1. Tasks and stimulus.....111

5.3.2. Glance analysis.....113

5.3.3. Results.....116

5.3.4. Conclusion and discussion.....121

5.4. Summary.....124

CHAPTER 6
EFFECT OF A CONCURRENT TASK ON VISUAL SEARCH:
STUDY WITH COMPUTATIONAL MODEL OF
VISUAL SEARCH AND TRACKING DUAL TASK

6.1. Introduction.....126

6.2. Existing Studies on Dual Task.....128

6.3. Queueing Network Model for Dual Task of Pedestrian Detection and Driving.....132

6.3.1. Pedestrian detection model.....132

6.3.2. Simple driving model.....141

6.3.3. Managing two tasks in the QN-MHP.....144

6.4. Simulation Results and Discussion……146

6.4.1. Results from reinforcement learning: different strategies from different visual stimulus……..147

6.4.2. Single task: pedestrian detection……..150

6.4.3. Dual task: pedestrian detection and simple driving……...151

6.5. Conclusion and Discussion……158

CHAPTER 7
CONCLUSIONS AND DISCUSSIONS

7.1. General Summary……...161

7.2. Contributions of This Work……...164

7.3. Limitations of This Work and Future Research……..169

REFERENCES……172

Computational Modeling and Experimental Research on Touchscreen Gestures, Audio/Speech Interaction, and Driving

by

Heejin Jeong

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Industrial and Operations Engineering)
in the University of Michigan
2018

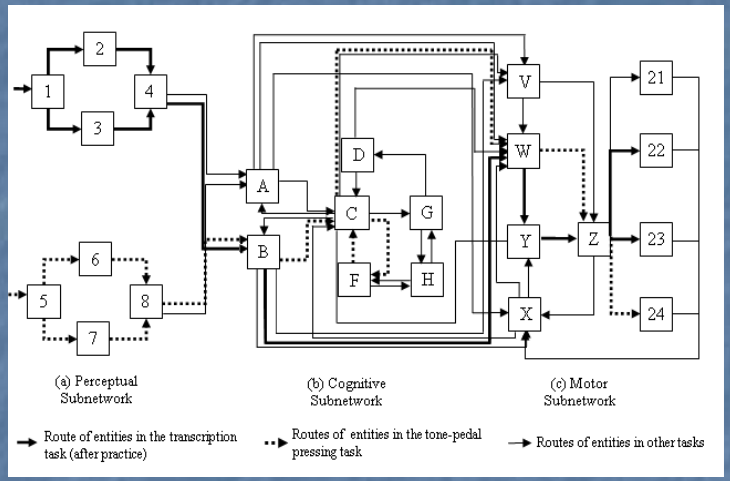
Queuing Network Modeling of Human Multitask Performance and its Application to Usability Testing of In-Vehicle Infotainment Systems

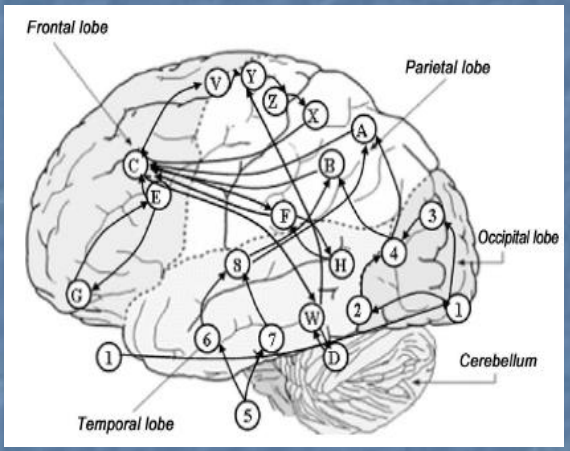
by

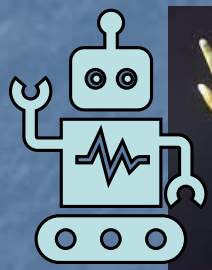
Ruijia Feng

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Industrial and Operations Engineering)
in the University of Michigan
2015

Queueing Network Modeling of Cognitive Architecture and Human-Machine Systems







Queueing Network of Mental Architecture

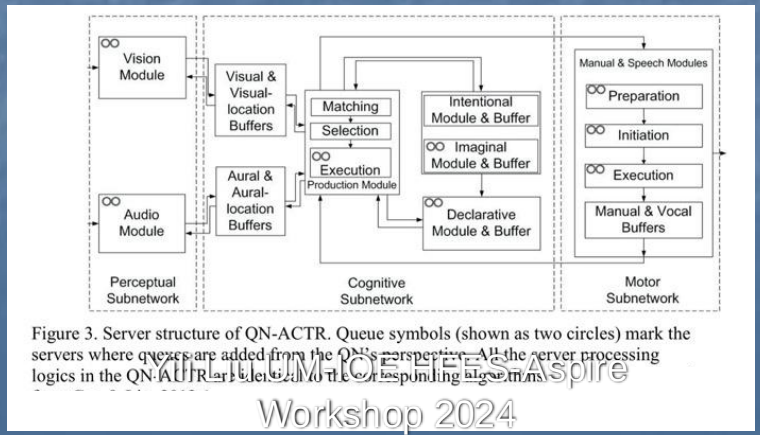
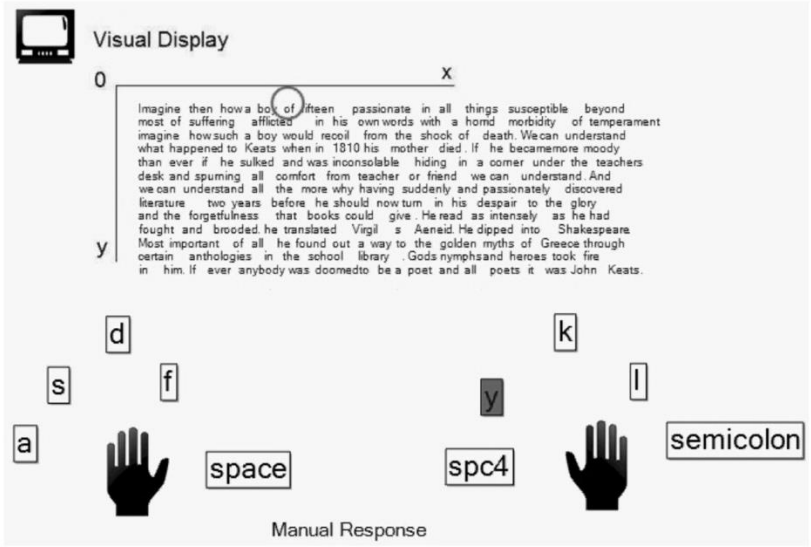


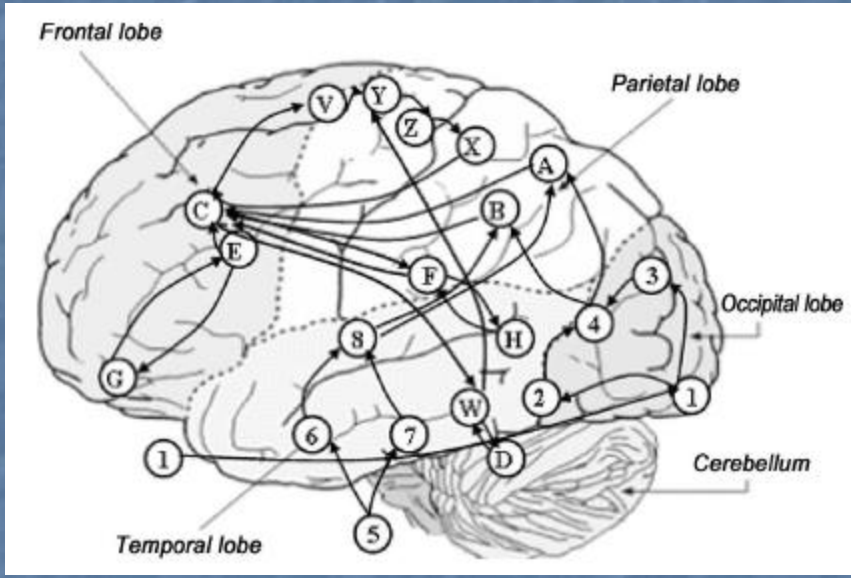
Figure 3. Server structure of QN-ACTR. Queue symbols (shown as two circles) mark the servers where queues are added from the QN's perspective. All the server processing logics in the QN-ACTR are identical to the corresponding algorithms

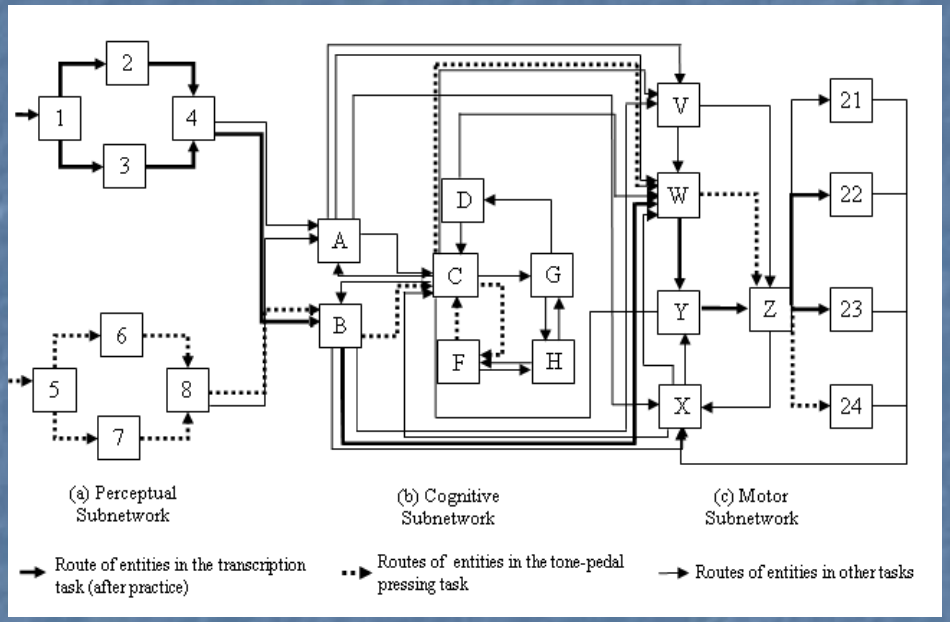
QN-ACTR can show the simulated typing behaviour in real time, as illustrated in the screenshot of Figure 3. Modelling results include text output traces for module activities, typing performance such as finger movement and eye movement, and reading comprehension performance such as reading speed and comprehension accuracy. The absolute percentage error (*APE*) and the coefficient of determination (R^2) were computed between QN-ACTR's results and the human results. These results are summarised in Table 5. The modelling results were similar to the human results from the experiments. It is particularly important to note that the modelling results captured the phenomena involving reading comprehension (i.e., typing is slower than reading; typing skill and comprehension are independent) that is difficult to model by the QN architecture alone and the phenomena involving concurrent tasks and skilled typing (e.g., a concurrent task does not affect typing performance) that is difficult to model by ACT-R alone. To examine whether an ACT-R model without queues can produce similar results, we repeated the QN-ACTR simulation without the QN mechanisms that were introduced previously in Section 4.1 Method. In this case, the reduced version of QN-ACTR became the same as ACT-R, as demonstrated previously in Section 3 Model Verification. This simulation produced a much longer typing interkey time of 500 ms. In comparison, the result from the simulation with the QN mechanisms was just 182 ms, much closer to the human result of 177 ms. Removing the QN mechanisms did not change choice reaction time (still 495 ms).

Figure 3 The visualisation of the task interaction in QN-ACTR



Notes: The visual display section shows the texts on the screen and the location of visual attention (represented by a circle). The manual response section shows that the index finger of the right hand is pressing key 'y' while other fingers are resting at the home locations.





Human Brain

Queueing Network of Mental Architecture

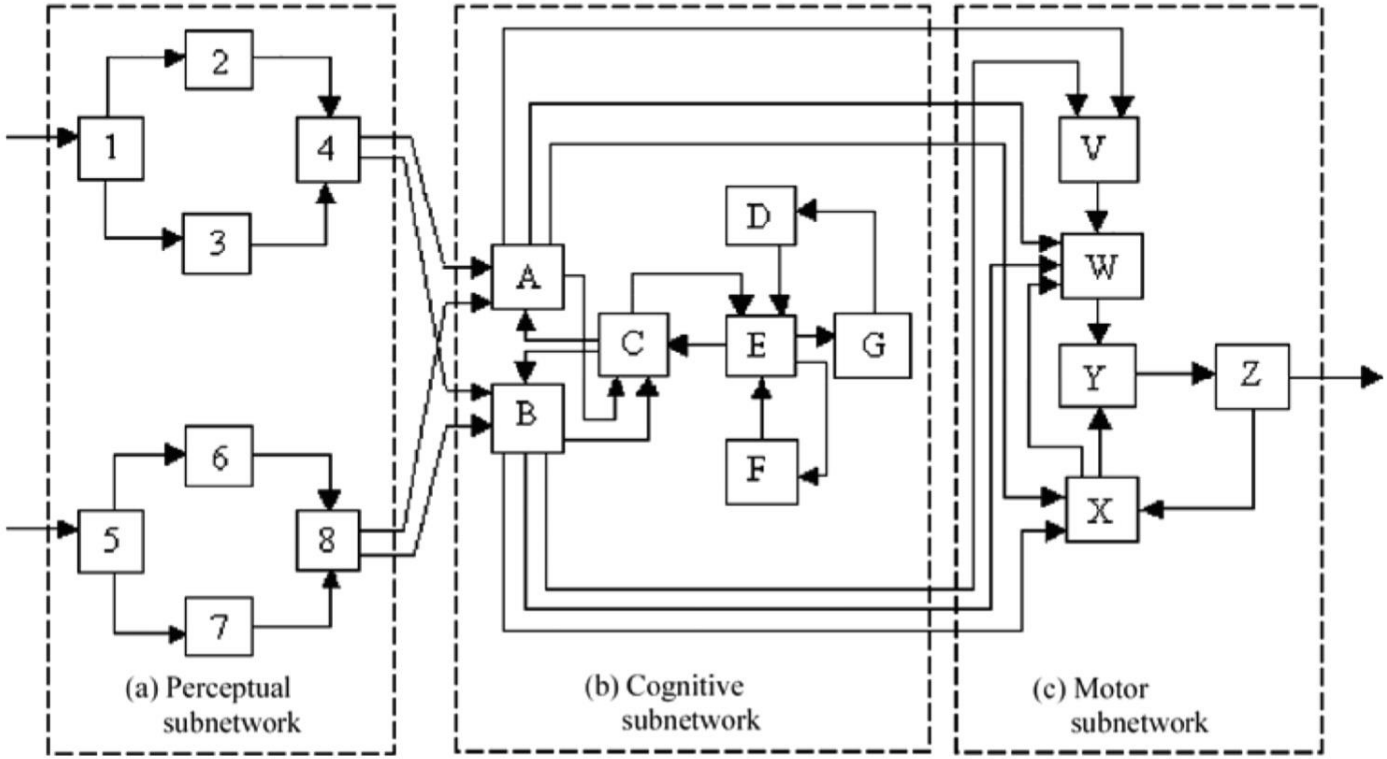
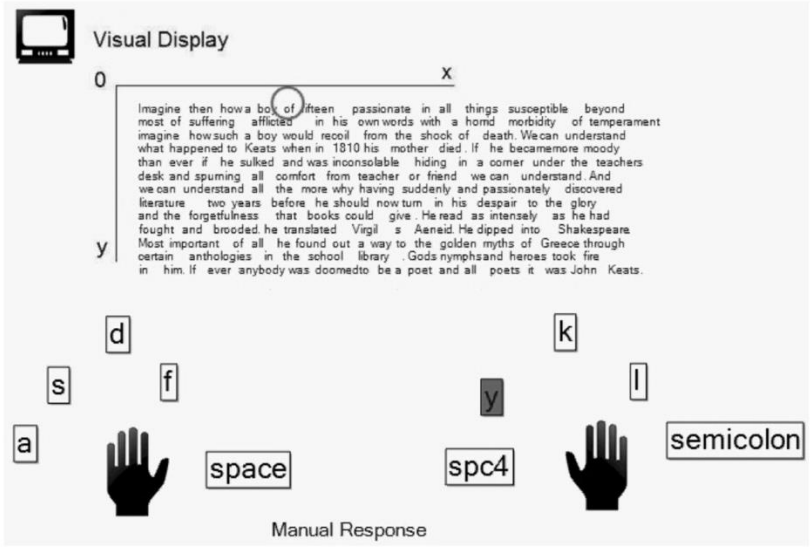


Fig. 2. The Queuing Network-Model Human Processor (QN-MHP) [from Feyen and Liu 2001a]. (a) Perceptual: 1 = common visual processing; 2 = visual recognition; 3 = visual location; 4 = location and recognition integrator; 5 = sound localization; 6 = linguistic processing; 7 = processing of other sounds; 8 = linguistic and other sounds integrator. (b) Cognitive: A = visuospatial sketchpad; B = phonological loop; C = central executor; D = goal procedures; E = performance monitoring; F = high-level cognitive operations; G = goal selection. (c) Motor: V = sensorimotor integration; W = motor element storage; X = movement tuning; Y = motor programming; Z=actuators.
Note: Although not shown in this figure, environmental and device servers receive output customers from server Z and supply input customers to server 1 and server 5.

QN-ACTR can show the simulated typing behaviour in real time, as illustrated in the screenshot of Figure 3. Modelling results include text output traces for module activities, typing performance such as finger movement and eye movement, and reading comprehension performance such as reading speed and comprehension accuracy. The absolute percentage error (*APE*) and the coefficient of determination (R^2) were computed between QN-ACTR's results and the human results. These results are summarised in Table 5. The modelling results were similar to the human results from the experiments. It is particularly important to note that the modelling results captured the phenomena involving reading comprehension (i.e., typing is slower than reading; typing skill and comprehension are independent) that is difficult to model by the QN architecture alone and the phenomena involving concurrent tasks and skilled typing (e.g., a concurrent task does not affect typing performance) that is difficult to model by ACT-R alone. To examine whether an ACT-R model without queues can produce similar results, we repeated the QN-ACTR simulation without the QN mechanisms that were introduced previously in Section 4.1 Method. In this case, the reduced version of QN-ACTR became the same as ACT-R, as demonstrated previously in Section 3 Model Verification. This simulation produced a much longer typing interkey time of 500 ms. In comparison, the result from the simulation with the QN mechanisms was just 182 ms, much closer to the human result of 177 ms. Removing the QN mechanisms did not change choice reaction time (still 495 ms).

Figure 3 The visualisation of the task interaction in QN-ACTR



Notes: The visual display section shows the texts on the screen and the location of visual attention (represented by a circle). The manual response section shows that the index finger of the right hand is pressing key 'y' while other fingers are resting at the home locations.

Appendix B

The shooting and arithmetic dual-task model

As a generic dual-task model, two sets of task-specific knowledge were defined independently. The shooting part was first modelled after a shooting single-task in an enemy-only scenario (Scribner et al., 2007). Production rules were defined to model the shooting-only task as the process of visual searching, manual aiming, and pulling the trigger, as shown in Table B1.

Table B1 English descriptions of the production rules in the shooting and arithmetic model.

Rule name	English description
Shooting task	
Find-target	IF a goal is *shooting* with state *start*, THEN look for the visual-location of an unattended target, and change the goal state to *looking*.
Attend-target	IF a goal is *shooting* with state *looking*, a visual-location is found, and visual module is free, THEN move visual attention to the visual-location, and change the goal state to *attending*.
Aim-target	IF a goal is *shooting* with state *attending*, a target is visually attended, and manual module is free, THEN move the iron sight towards the target, and change the goal state to *aiming*.
Shoot-enemy	IF a goal is *shooting* with state *aiming*, the visual target is *brown*, and manual module is free, THEN pull the trigger, and change the goal state to *shooting*.
Dont-shoot-friendly	IF a goal is *shooting* with state *aiming*, the visual target is *olive*, and manual module is free, THEN change the goal state to *shooting*.
Home	IF a goal is *shooting* with state *shooting*, and manual module is free, THEN move the iron sight back to the initial *home location*, and change the goal state back to *start*.
Shoot	IF a goal is *shooting* with state *aiming*, and manual module is free, THEN pull the trigger, and change the goal state to *shooting*.
Addition task	
Hear-sound	IF a goal is *addition*, aural-location has an external sound, and aural module is free, THEN attend to the sound.
Hear-digit-1	IF a goal is *addition* with argument-1, argument-2, and sum all empty, a *number* sound is attended, and declarative module is free, THEN retrieve a number chunk with value *number*.
Encode-digit-1	IF a goal is *addition* with argument-1, argument-2, and sum all empty, and a number chunk is retrieved, THEN set argument-1's value as the number chunk.
Hear-digit-2	IF a goal is *addition* with argument-2 and sum empty, but argument-1 filled, a *number* sound is attended, and declarative module is free, THEN retrieve a number chunk with value *number*.
Encode-digit-2	IF a goal is *addition* with argument-2 and sum empty, but argument-1 filled, and a number chunk is retrieved, THEN set argument-2's value as the number chunk.
Initialize-addition	IF a goal is *addition* with argument-1 and argument-2 filled, but sum empty, THEN retrieve a count-order chunk with argument-1's value as the first, set the goal's sum as argument-1's value, count as *zero*, state as *increment-sum*.
Terminate-addition	IF a goal is *addition* with argument-2 and count equal, and sum is filled, and declarative module is free, THEN retrieve the sum's value chunk, and reset the goal's argument-1, argument-2, and count to empty.
Speak-result	IF a goal is *addition* with argument-1, 2 and count empty, but sum is filled, and a number is retrieved with a value, THEN speak the value, and remove the old *addition* goal and create a new one.
Increment-sum	IF a goal is *addition* with sum filled, count not equal to argument-2, state is *increment-sum*, and a count-order is retrieved, THEN set the goal's sum as the second number in the count-order, change state to *increment-count*, and retrieve a count-order whose first number equals to the count value of the goal.
Increment-count	IF a goal is *addition* with sum and count filled, state is *increment-count*, and a count-order is retrieved, THEN set the goal's count as the second number in the count-order, change state to *increment-sum*, and retrieve a count-order whose first number equals to the sum value of the goal.
Restart-from-error	IF a goal is *addition* with argument-1 filled, and the state of declarative module is *error*, THEN remove the old *addition* goal and create a new one.

Computers & Industrial Engineering 109 (2017) 313–324

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

A computer-aided usability testing tool for in-vehicle infotainment systems

Fred Feng [a,*], Yili Liu [b], Yifan Chen [c]

[a] University of Michigan Transportation Research Institute, Ann Arbor, MI, USA
[b] University of Michigan, Department of Industrial and Operations Engineering, Ann Arbor, MI, USA
[c] Ford Motor Company, Dearborn, MI, USA

ARTICLE INFO

Article history:
Received 28 December 2016
Received in revised form 15 May 2017
Accepted 17 May 2017
Available online 19 May 2017

Keywords:
Computer-aided engineering
Human factors
Usability testing
Human performance modeling
In-vehicle infotainment system
Queuing network-model human processor

ABSTRACT

This paper describes the development of a computer-aided engineering (CAE) software toolkit for designers of in-vehicle infotainment systems to predict and benchmark the system usability, such as task completion time, eye glance behaviors, and mental workload. A digital driver model was developed based on the task-independent cognitive architecture of QN-MHP (Queuing Network-Model Human Processor). At the front end of the software a graphical user interface (GUI) was developed that allows designers to create digital mockups of the designs and simulate drivers performing secondary tasks while steering a vehicle. To validate the software outputs, an experiment using human drivers was conducted on a fix-based driving simulator with a radio-tuning task as a test case. Three typical in-vehicle infotainment systems that have the function of radio tuning were investigated (a touch screen, physical buttons, and a knob). The results show that the software was able to generate task completion time, total eyes-off-road time, and mental workload estimates that were similar to the empirical data. The software toolkit has the potential to be a supplemental tool for designers to explore a larger design space and address usability issues at the early design stages with lower cost in time and manpower.

1. Introduction

The last decade has witnessed a steady trend of modern electronic technologies, such as touch screens, digital instrument clusters, and head-up displays, becoming common features in many vehicles (Becker, Hanna, & Wagner, 2014). These technologies enable the vehicle cockpit to include an increasing number and variety of infotainment features, such as browsing music, making phone calls, and finding the nearest gas station. Although these functions are designed to enhance the driving experience, they may suffer from usability problems such as driver distraction with extended eyes-off-road operations, information overload due to cluttered displays and system complexity (Becker et al., 2014; Lee, 2007).

Driver distraction has been regarded as one of the leading causes of road accidents. According to the National Highway Traffic Safety Administration (NHTSA), distracted driving accounted for 3477 fatalities (10% of overall fatalities) and an estimated additional 391,000 injuries in the United States in 2015 (National Center for Statistics, 2017). In addition, these numbers are likely under-reported due to the difficulties in identifying driver distraction during accident investigation. A naturalistic driving study shows distraction of secondary tasks (i.e., those tasks not necessary to driving) account for 23% of all crashes and near-crashes (Klauer, Dingus, Neale, Sudweeks, & Ramsey, 2006). Empirical studies have shown that distraction degrades many aspects of driving performance such as drivers' steering behavior and lane keeping performance (Pavlidis et al., 2016; Peng, Boyle, & Hallmark, 2013), braking behavior (Harbluk, Noy, & Eizenman, 2002), and response to hazard events (Horrey & Wickens, 2004). Thus it is critical to test and benchmark the usability of the infotainment systems during the design process to minimize driver distraction and ensure that safety is not compromised.

Standards and guidelines have been proposed to evaluate a variety of secondary tasks and the designs of in-vehicle systems. The Society of Automotive Engineering (SAE) Standard J2364 proposes that the maximum time for drivers to complete navigation-related tasks involving visual displays and manual controls should be less than 15 s (referred to as the 15-Second Rule) (Green, 1999). NHTSA published a guideline for in-vehicle electronic devices with recommendations that tasks should be completed with a driver's glances away from the road of less than 2 s

* Corresponding author at: University of Michigan Transportation Research Institute, 2901 Baxter Road, Ann Arbor, MI 48109, USA.
E-mail address: fredfeng@umich.edu (F. Feng).

http://dx.doi.org/10.1016/j.cie.2017.05.019
0360-8352/

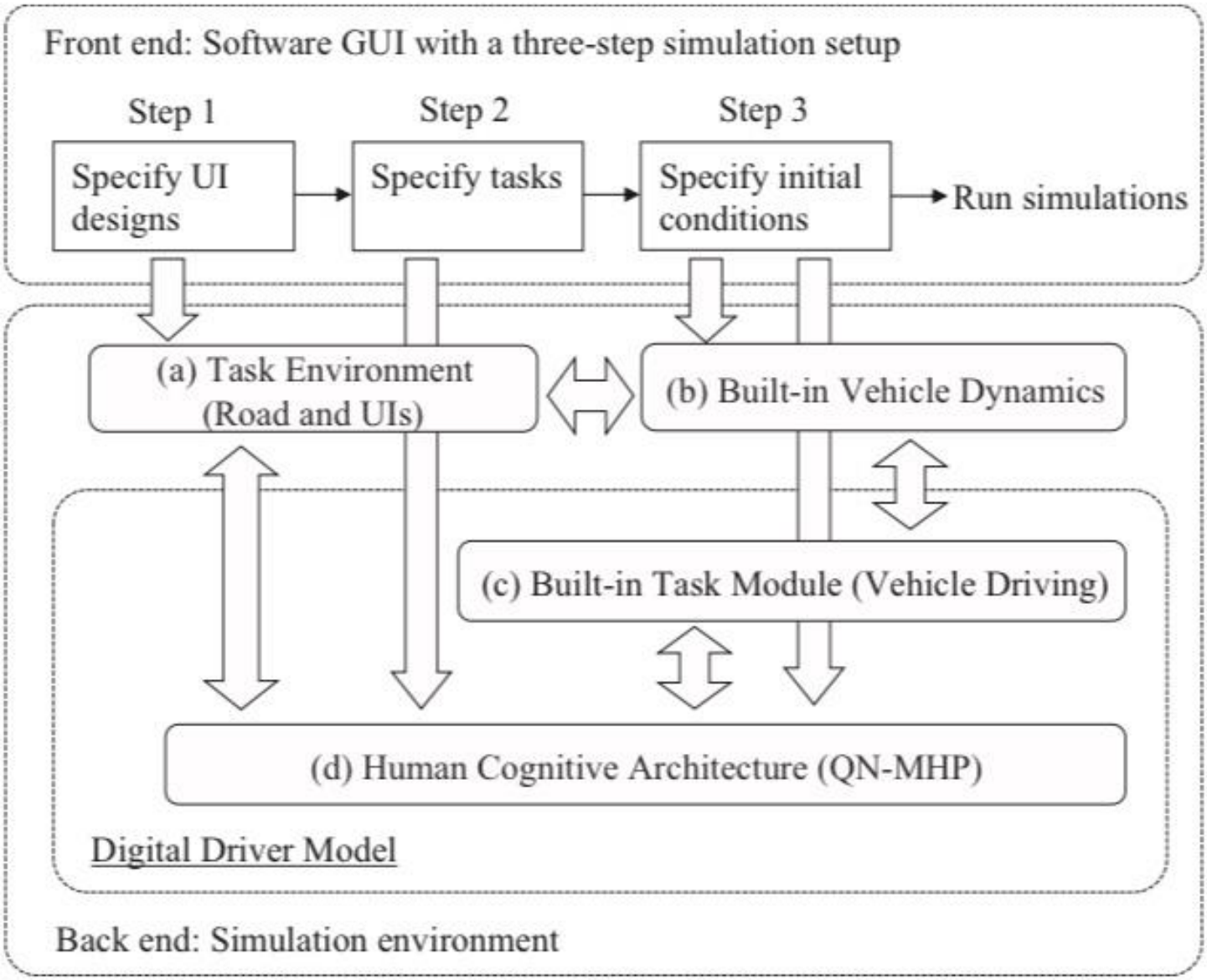


Fig. 1. Software structure.

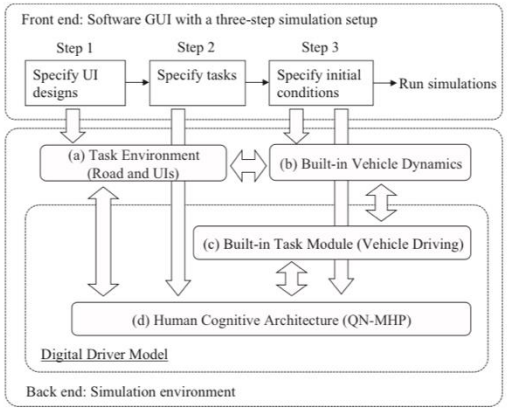


Fig. 1. Software structure.

to output the task performance, drivers' eye glance behavior, and mental workload, based on the status of the queuing network. The method on estimating driver mental workload is based on the utilizations of the related QN-MHP servers. The details of the method could be found in Wu and Liu (2007). Fig. 2 is a screenshot of the MATLAB/Simulink implementation of the digital driver model.

2.1.2. Design prototyping

A prototyping tool is needed to create design mockups that the driver model could interact with. The interactions include the driver model both perceiving information from the digital mockup (e.g., visual information from a display) and generating motor actions upon the digital mockup (e.g., pressing a button). A prototyping tool was developed using MATLAB GUIDE (GUI Develop-

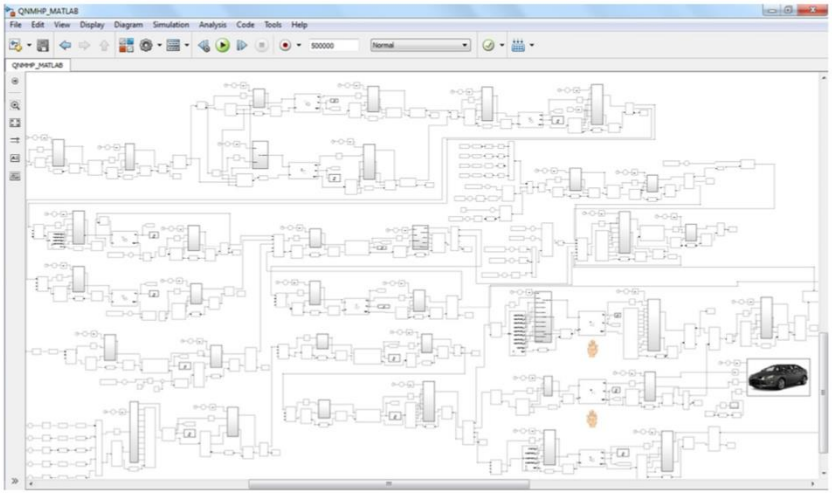


Fig. 2. A screenshot of the QN-MHP model implementation in MATLAB/Simulink.

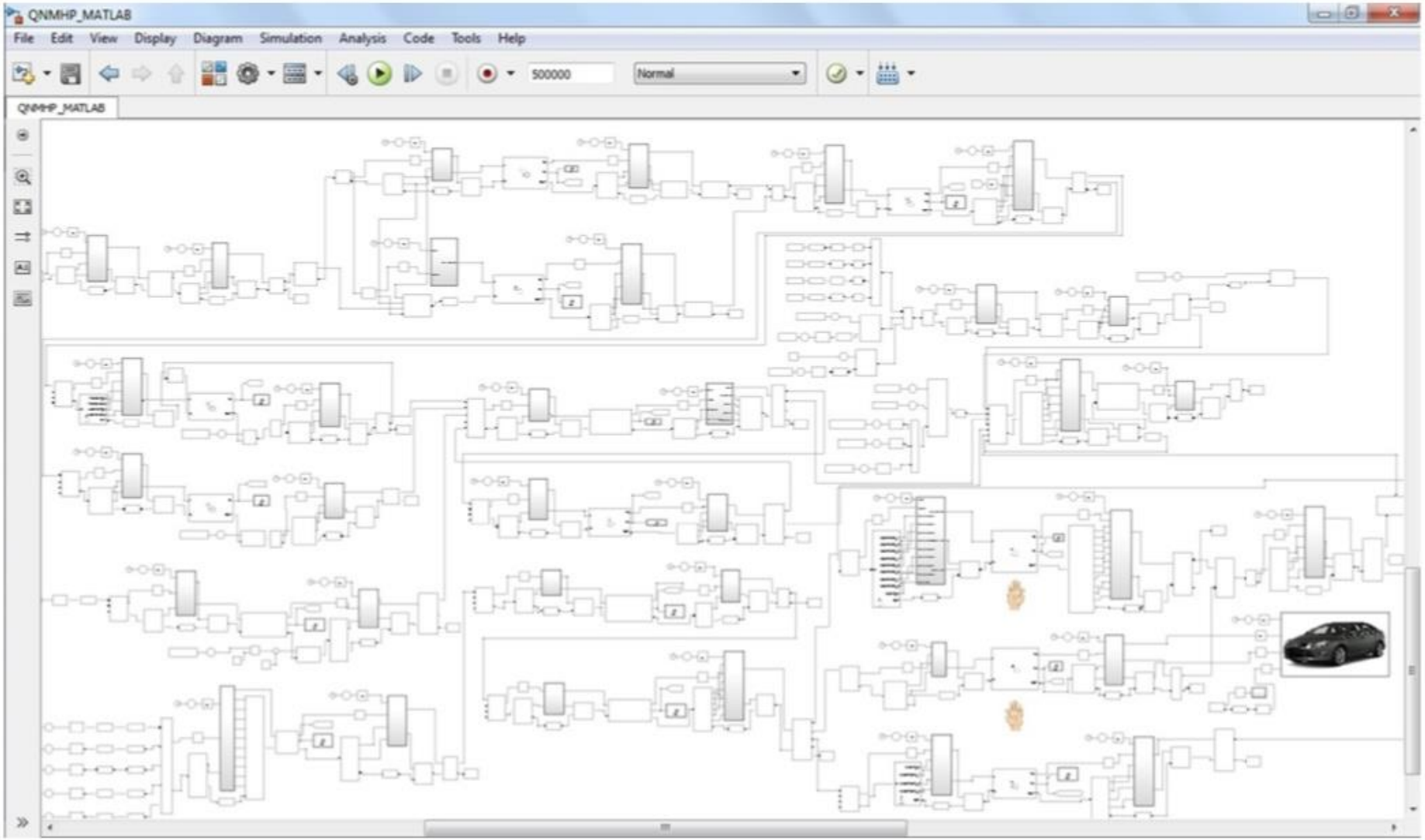


Fig. 2. A screenshot of the QN-MHP model implementation in MATLAB/Simulink.

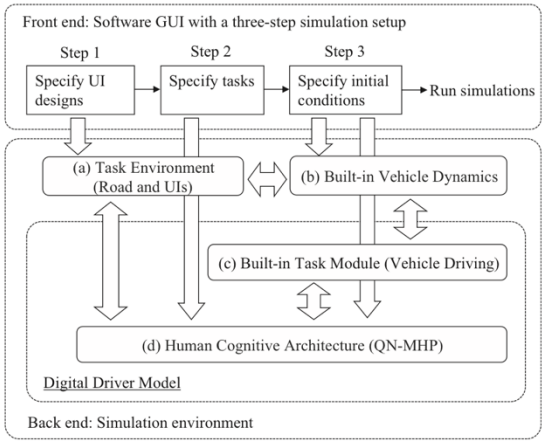


Fig. 1. Software structure.

to output the task performance, drivers' eye glance behavior, and mental workload, based on the status of the queuing network. The method on estimating driver mental workload is based on the utilizations of the related QN-MHP servers. The details of the method could be found in Wu and Liu (2007). Fig. 2 is a screenshot of the MATLAB/Simulink implementation of the digital driver model.

2.1.2. Design prototyping

A prototyping tool is needed to create design mockups that the driver model could interact with. The interactions include the driver model both perceiving information from the digital mockup (e.g., visual information from a display) and generating motor actions upon the digital mockup (e.g., pressing a button). A prototyping tool was developed using MATLAB GUIDE (GUI Develop-

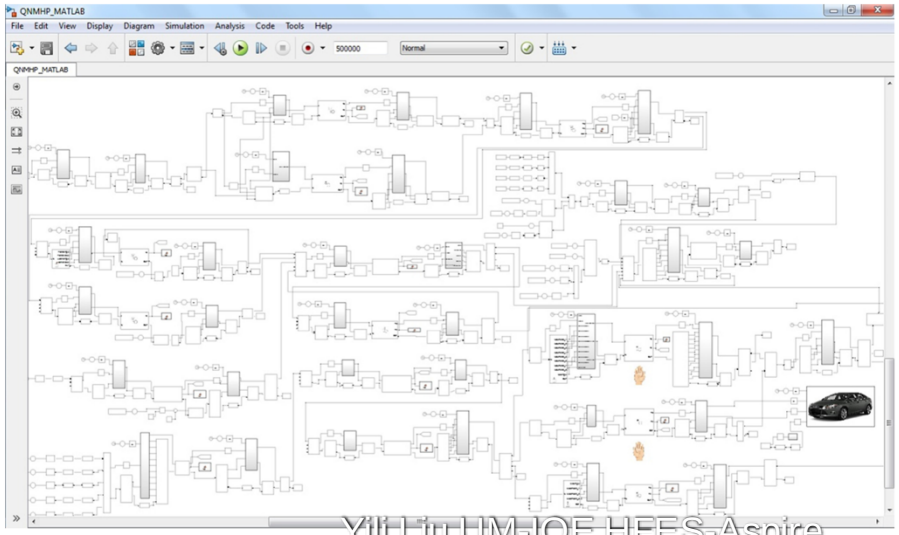


Fig. 2. A screenshot of the QN-MHP model implementation in MATLAB/Simulink.

Computers & Industrial Engineering 109 (2017) 313–324

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

A computer-aided usability testing tool for in-vehicle infotainment systems

Fred Feng [a,*], Yili Liu [b], Yifan Chen [c]

[a] University of Michigan Transportation Research Institute, Ann Arbor, MI, USA
[b] University of Michigan, Department of Industrial and Operations Engineering, Ann Arbor, MI, USA
[c] Ford Motor Company, Dearborn, MI, USA

ARTICLE INFO

Article history:
Received 28 December 2016
Received in revised form 15 May 2017
Accepted 17 May 2017
Available online 19 May 2017

Keywords:
Computer-aided engineering
Human factors
Usability testing
Human performance modeling
In-vehicle infotainment system
Queuing network-model human processor

ABSTRACT

This paper describes the development of a computer-aided engineering (CAE) software toolkit for designers of in-vehicle infotainment systems to predict and benchmark the system usability, such as task completion time, eye glance behaviors, and mental workload. A digital driver model was developed based on the task-independent cognitive architecture of QN-MHP (Queuing Network-Model Human Processor). At the front end of the software a graphical user interface (GUI) was developed that allows designers to create digital mockups of the designs and simulate drivers performing secondary tasks while steering a vehicle. To validate the software outputs, an experiment using human drivers was conducted on a fix-based driving simulator with a radio-tuning task as a test case. Three typical in-vehicle infotainment systems that have the function of radio tuning were investigated (a touch screen, physical buttons, and a knob). The results show that the software was able to generate task completion time, total eyes-off-road time, and mental workload estimates that were similar to the empirical data. The software toolkit has the potential to be a supplemental tool for designers to explore a larger design space and address usability issues at the early design stages with lower cost in time and manpower.

1. Introduction

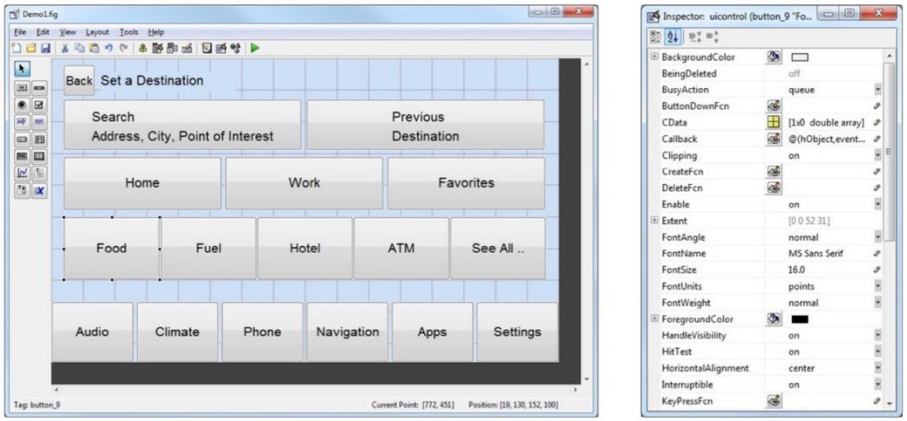
The last decade has witnessed a steady trend of modern electronic technologies, such as touch screens, digital instrument clusters, and head-up displays, becoming common features in many vehicles (Becker, Hanna, & Wagner, 2014). These technologies enable the vehicle cockpit to include an increasing number and variety of infotainment features, such as browsing music, making phone calls, and finding the nearest gas station. Although these functions are designed to enhance the driving experience, they may suffer from usability problems such as driver distraction with extended eyes-off-road operations, information overload due to cluttered displays and system complexity (Becker et al., 2014; Lee, 2007).

Driver distraction has been regarded as one of the leading causes of road accidents. According to the National Highway Traffic Safety Administration (NHTSA), distracted driving accounted for 3477 fatalities (10% of overall fatalities) and an estimated additional 391,000 injuries in the United States in 2015 (National Center for Statistics, 2017). In addition, these numbers are likely under-reported due to the difficulties in identifying driver distraction during accident investigation. A naturalistic driving study shows distraction of secondary tasks (i.e., those tasks not necessary to driving) account for 23% of all crashes and near-crashes (Klauer, Dingus, Neale, Sudweeks, & Ramsey, 2006). Empirical studies have shown that distraction degrades many aspects of driving performance such as drivers' steering behavior and lane keeping performance (Pavlidis et al., 2016; Peng, Boyle, & Hallmark, 2013), braking behavior (Harbluk, Noy, & Eizenman, 2002), and response to hazard events (Horrey & Wickens, 2004). Thus it is critical to test and benchmark the usability of the infotainment systems during the design process to minimize driver distraction and ensure that safety is not compromised.

Standards and guidelines have been proposed to evaluate a variety of secondary tasks and the designs of in-vehicle systems. The Society of Automotive Engineering (SAE) Standard J2364 proposes that the maximum time for drivers to complete navigation-related tasks involving visual displays and manual controls should be less than 15 s (referred to as the 15-Second Rule) (Green, 1999). NHTSA published a guideline for in-vehicle electronic devices with recommendations that tasks should be completed with a driver's glances away from the road of less than 2 s

* Corresponding author at: University of Michigan Transportation Research Institute, 2901 Baxter Road, Ann Arbor, MI 48109, USA.
E-mail address: fredfeng@umich.edu (F. Feng).

http://dx.doi.org/10.1016/j.cie.2017.05.019
0360-8352/



(a) Main screen (b) Object property inspector

Fig. 3. Prototyping UI designs using MATLAB GUIDE.

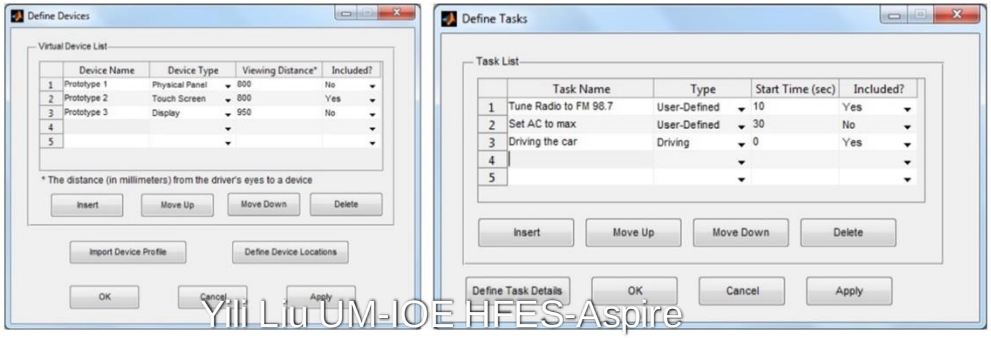
ment Environment). MATLAB GUIDE allows creating interface designs with the support of common GUI objects including text, push buttons, etc. These GUI objects could be created graphically with drag-and-drop and edited using the object property inspector (see Fig. 3). A GUI object's behavior upon user actions (e.g., a button being clicked) could be specified in its callback functions. MATLAB GUIDE supports WYSIWYG (What You See Is What You Get) and allows users to check the look and behavior of the current design at any time during prototyping. Once a design mockup is created, it can be saved as a stand-alone file that can be shared among designers and imported into the simulation software for evaluation.

Fig. 4. Main window of the software simulation.

2.1.3. Simulation setup

Once the digital mockups of the designs are created, the software users could setup the simulation using the software GUI. At the front end the GUI allows the software users to setup and run the simulation in three steps. Fig. 4 shows the main window of the software GUI.

Step 1: The setup starts with adding virtual devices (i.e., digital mockups created in Section 2.1.2) to the task environment. A virtual device could be added by filling out a single row in the virtual device table (see Fig. 5a) with a device name/type (currently it



(a) Step 1: define virtual devices (b) Step 2 :define tasks

Fig. 5. Define UI designs and tasks.

GOAL: Tune the radio to FM 98.7 using the direct tune on the physical panel

Method to accomplish goal of tuning the radio to FM 98.7 using the direct tune

TC 1: Look at <text feature> on <physical panel> at location <190, 46>

TC 2: Store <feature value> to short-term memory

TC 3: Retrieve <feature value> from short-term memory

TC 4: Compare <feature value> to <"Power">

If match, return *result* = 1, else return *result* = 0

TC 5: If *result* = 1, go to TC 6, else go to TC 999 // 999 is a dummy TC

TC 6: Reach <location> with <right hand> <with visual guidance>

TC 7: Click with <right hand> <index finger>

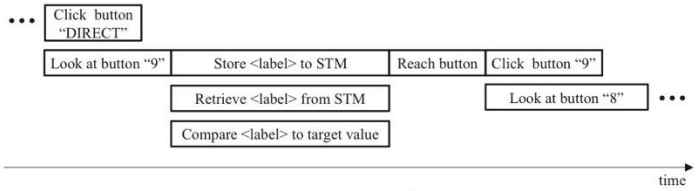
TC 8: Look at <text feature> on <physical panel> at location <224, 88>

... : ...

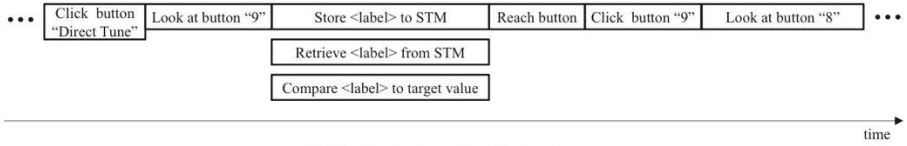
TC N-1: Click with <right hand> <index finger> // click the last button ("7")

TC N: Return with goal accomplished

Fig. 11. NGOMSL-style task analysis of tuning the radio to FM 98.7 using the physical buttons.



(a) Radio-tuning using the physical buttons



(b) Radio-tuning using the touch screen

Fig. 12. Sequential dependency of the task components in the parked condition.

To complete these tasks while driving, time-sharing and task-switching strategies are needed to shift the visual attention between the road scene and the secondary task devices. The secondary task needs to be partitioned so that the driver would operate the vehicle in a safer manner by constantly checking the road for any potential unsafe situations. For the radio-tuning task using physical buttons, it is assumed that the driver could use the tactile feedback to confirm the successful click of a button, and thus could start looking back at the road as soon as his/her hand reaches a button. However, when using the touch screen, the driver can only start looking at the road after the clicking is completed. Fig. 13 illustrates the sequential dependency of the task components when using the physical buttons and touch screen while driving. It can be observed that the task using the physical buttons can be performed in a more parallel manner with the driving task (top figure) compared with the task using a touch screen (bottom figure).

3. Results

3.1. Task completion time

The time it took for the participants to complete the radio-tuning task in each experiment trial was extracted from the video captured by the camera facing the center console area. Normality tests show that the task completion time did not follow a normal distribution in all the groups ($p < 0.001$). Since the normality assumption of the paired t-test is violated, Wilcoxon signed-rank test (a non-parametric equivalent to the t-test) was conducted to

GOAL: Tune the radio to FM 98.7 using the direct tune on the physical panel

Method to accomplish goal of tuning the radio to FM 98.7 using the direct tune

TC 1: Look at <text feature> on <physical panel> at location <190, 46>

TC 2: Store <feature value> to short-term memory

TC 3: Retrieve <feature value> from short-term memory

TC 4: Compare <feature value> to <"Power">

If match, return *result* = 1, else return *result* = 0

TC 5: If *result* = 1, go to TC 6, else go to TC 999 // 999 is a dummy TC

TC 6: Reach <location> with <right hand> <with visual guidance>

TC 7: Click with <right hand> <index finger>

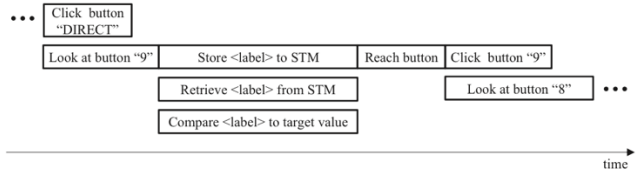
TC 8: Look at <text feature> on <physical panel> at location <224, 88>

... : ...

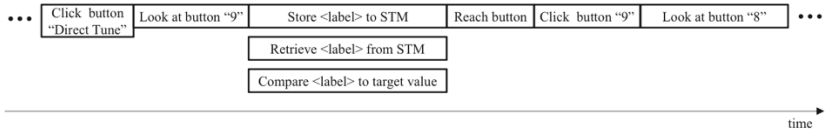
TC N-1: Click with <right hand> <index finger> // click the last button ("7")

TC N: Return with goal accomplished

Fig. 11. NGOMSL-style task analysis of tuning the radio to FM 98.7 using the physical buttons.



(a) Radio-tuning using the physical buttons



(b) Radio-tuning using the touch screen

Fig. 12. Sequential dependency of the task components in the parked condition.

To complete these tasks while driving, time-sharing and task-switching strategies are needed to shift the visual attention between the road scene and the secondary task devices. The secondary task needs to be partitioned so that the driver would operate the vehicle in a safer manner by constantly checking the road for any potential unsafe situations. For the radio-tuning task using physical buttons, it is assumed that the driver could use the tactile feedback to confirm the successful click of a button, and thus could start looking back at the road as soon as his/her hand reaches a button. However, when using the touch screen, the driver can only start looking at the road after the clicking is completed. Fig. 13 illustrates the sequential dependency of the task components when using the physical buttons and touch screen while driving. It can be observed that the task using the physical buttons can be performed in a more versatile manner with the driving task (top figure) compared with the task using a touch screen (bottom figure).

3. Results

3.1. Task completion time

The time it took for the participants to complete the radio-tuning task in each experiment trial was extracted from the video captured by the camera facing the center console area. Normality tests show that the task completion time did not follow a normal distribution in all test groups ($p < 0.001$). Since the normality assumption of the paired *t*-test is violated, Wilcoxon signed-rank test (a non-parametric equivalent to the *t*-test) was conducted to

INTERNATIONAL JOURNAL OF HUMAN–COMPUTER INTERACTION
2019, VOL. 35, NO. 6, 510–520
https://doi.org/10.1080/10447318.2018.1466858

Computational Modeling of Touchscreen Drag Gestures Using a Cognitive Architecture and Motion Tracking

Heejin Jeong and Yili Liu

Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI, USA

ABSTRACT

This article presents a computational model that predicts finger-drag gesture performance on touchscreen devices, by integrating the queueing network (QN) cognitive architecture and motion tracking. Specifically, the QN-based model was developed to predict two execution times: the finger movement time of drag-gesture (i.e., only the motion time of the finger touched and dragged on the surface of touchscreen) and the comprehensive process time of drag-gesture (i.e., the entire process time to complete the finger-drag task, including visual attention shift, memory storage and retrieval, and hand-finger movements). To develop predictive models for the finger movement time of drag-gesture, 11 participants' motion data were collected and a regression analysis with parameters of hand-finger anthropometric data and eight angular directions was conducted. Human subject data from our previous study (Jeong & Liu, 2017a) were used to evaluate the QN-based model, generating similar outputs (R^2 was more than 80% and root-mean square was less than 300 msec) for both execution times.

1. Introduction

As touchscreen technologies have rapidly developed over the last few decades, diverse interfaces that require a wide range of touchscreen gestures have become popular (Bhalla & Bhalla, 2010; Saffer, 2008). Although the definition varies with the research domain, touchscreen gestures can be generally categorized into two types, depending on the number of fingers used: single-touch and multi-touch gestures. Typical examples of a single-touch gesture include tap, swipe, and drag, whereas pinch and spread are examples of a multi-touch gesture.

Since the use of touchscreen gestures has become more prevalent in our daily lives, humans are likely to have more chances to interact with systems or environments using the touchscreen gestures. For instance, they perform touchscreen gestures while walking or driving to navigate to the destination. Thus, it is necessary to investigate human behaviors for touchscreen gesture tasks in the comprehensive process, using human brain and body segments, not just an index finger movement itself.

Many experimental studies have investigated finger gesture performance on touchscreens (e.g., Asakawa, Dennerlein, & Jindrich, 2017; Jeong & Liu, 2017a; Jorritsma, Prins, & Van Ooijen, 2015; Kim & Jo, 2015; Parhi, Karlson, & Bederson, 2006; Sasangohar, MacKenzie, & Scott, 2009). In addition to conducting experiments, several studies have developed prediction models using the empirical data obtained through their experiments (e.g., Epps, 1986; Bi, Li & Zhai, 2013; Ljubic, Glavinic, & Kukec, 2015). However, most of these models have several limitations. First, they relied on Fitts' law or its extension, treating movement time as a function of only the distance to and size of the target (Fitts, 1954). In other words, these previous models do not consider individual differences that affect finger gesture performance, such as human's anthropometry. Furthermore, the previous models focused on just finger movements, not considering human perceptual and cognitive processes; thus, these models cannot fully represent the details of comprehensive activities in human–computer interaction, such as visual attention shift, and memory storage and retrieval.

According to the literature, the movement time of an object (e.g., a fingertip or a mouse cursor) differs depending on which direction the object is heading to. Jagacinski and Monk (1985) compared the movement times when both joystick and head-mounted sights were used in two-dimension directions. In their study, horizontal and vertical movement times were slightly shorter than diagonal movement time, but finger movements on touchscreens were not investigated. In addition to movement direction, anthropometry is also regarded as a factor that affects movement time. Bergstrom-Lehtovirta and Oulasvirta (2014) found that hand size was one of the factors in predicting the functional area of the thumb on a touchscreen surface. Scott and Conzola (1997) examined the effect of finger size on touchscreen keying performance. They found that finger size had a significant effect on keying speed and duplication errors (i.e., a button on the touchscreen was

CONTACT Heejin Jeong heejin@umich.edu
Color versions of one or more of the figures in the article can be found online at www.tandfonline.com/hihc.

pushed twice by accident, even though a user intended to push it only once.). Their findings also revealed that smaller fingers produced significantly faster keying performance but more duplication errors.

In this study, we used a cognitive architecture, the queueing network-model human processor (QN-MHP), to develop a computational cognitive model to predict comprehensive finger-drag gesture performance on touchscreens, for the purpose of helping interface designers to evaluate usability in the early design process (Biswas, Robinson, & Langdon, 2012; Ocak & Cagiltay, 2017). The QN-MHP is a cognitive architecture that integrates the mathematical framework of queueing theory with the human processor model (Liu, Feyen, & Tsimhoni, 2006) to enable computational modeling of human mind and behavior. Over the past years, it has been extensively applied to a variety of domains including in-vehicle map reading (Liu et al., 2006) and transcription typing (Cao, Ho, & He, 2017; Wu & Liu, 2008). However, there was no QN-MHP model to predict performance of comprehensive single-touch drag gestures, one of the most frequently used touchscreen gestures. In this article, we present a QN-based computational model that predicts finger-drag gesture performance on touchscreen devices, through a model development process described in detail next.

Figure 1 shows the conceptual structure for integration of motion tracking and the QN-MHP architecture, to model single-touch drag gestures in the comprehensive process (i.e., the entire process to complete the finger-drag task, including visual attention shift, memory storage and retrieval, and hand-finger movements). We first needed predictive models in eight different angular directions (i.e., 0°, 45°, 90°, 135°, 180°, 225°, 270°, and 315°) to predict how long the movement of the finger-drag performed on the touchscreen devices takes (i.e., the finger movement time of drag-gesture). To develop the predictive models, we collected the motion data of finger-drag movement in eight different angular directions, using a motion tracking system. Then we used the time frames of the motion data as a measurement of the finger-

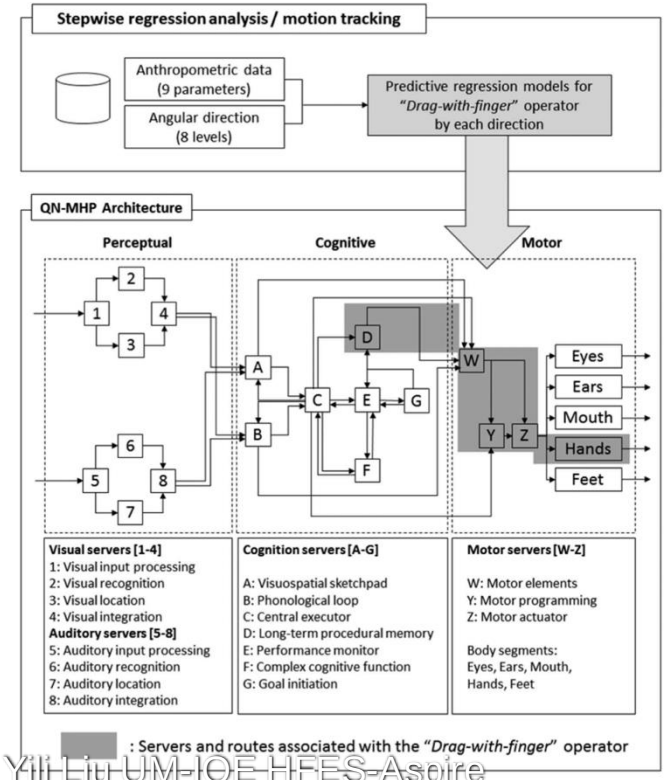


Figure 1. Conceptual structure for integration of motion tracking and the QN-MHP architecture.

Table 1. NGOMSL-style task description of comprehensive finger-drag gesture task.

Goal: Make a finger-drag gesture on a touchscreen device
TC 1: *Look-at* <target type> on <device id> at location <x, y>
TC 2: *Store* <target value> to short-term-memory
TC 3: *Retrieve* <target value> from short-term-memory
TC 4: *Compare* <target value> to <expected value>
If match, go to TC 5, else return to TC 1.
TC 5: *Decide*
TC 6: *Look-at* <target type> on <device id> at location <x, y>
TC 7: *Store* <target value> to short-term-memory
TC 8: *Retrieve* <target value> from short-term-memory
TC 9: *Determine-hand-movement*
TC 10: *Reach-with-hand*
TC 11: *Look-at* <target type> on <device id> at location <x, y>
TC 12: *Store* <target value> to short-term-memory
TC 13: *Retrieve* <target value> from short-term-memory
TC 14: *Determine-finger-movement*
TC 15: ***Drag-with-finger*** with <anthropometric data> in <direction>
TC 16: Return with goal accomplished

whether the information perceived equals to the information expected.

2.1.2. Hand reach to the target [TCs 6–10]

In the second subtask, a perception-cognition-motor process using a hand is performed. Specifically, it includes perceiving and remembering the information of visual signal's location to reach. Using the information, it is determined whether to move the hand. In the motor subnetwork, the hand servers make the hand be reached to the target, based on the estimation of how far/long the hand reaches the target.

2.1.3. Finger-drag on the target [TCs 11–15]

In this subtask, similar to the second subtask, a perception-cognition-motor process is performed, but using a finger, not a hand. After receiving and remembering the visual information to drag, the model determines whether to perform the finger-drag using a finger. The hand servers make the finger be dragged on the target, based on the estimation of how far/long the finger-drags on the target.

2.2. Development of operators in the QN-MHP architecture

2.2.1. Visual perception operator

Look-at. The purpose of this operator is to allow a human model to look at a specific location. Three parameters are used to set the specific target location: type of target (e.g., text or color), device id, and a target's two-dimensional coordinates on the device. Once the "*Look-at*" operator is activated at Server D (a long-term procedural memory server), it verifies whether an eye-saccade motor action is needed, which is used to make the current visual attention be on the target location. If the saccade action is needed, because the current visual attention is not on the target location, it triggers a saccade motor action at Server W (a motor-elements server). Then Server W triggers the Eyes server so that a saccade can be executed at the Eyes server. The saccade execution time is determined by a visual angle (i.e., the angle from the current location of visual attention to the target location) and an angular velocity (2,700 deg/sec) (Viviani & Stucchi, 1992). After the saccade is completed at the Eyes server, an entity (or visual stimulus) of target enters into Server 1 (a visual input server). Then the entity enters Servers 2 (Visual recognition) and 3 (Visual location) in parallel, and thus the human model can recognize the target and its location. Through Server 4 (a visual integration server), the entity is transformed into cognitive subnetwork including Servers A–G. For the finger-drag task in this study, the geometric center of a virtual touchscreen device was set as the location of the target to look at. On the target, a green circle was used as a visual stimulus in this model. Since the initial visual attention was set on the device, the saccade motor action was not required.

2.2.2. Memory and cognition operators

Store to short-term-memory. Using this operator, the model stores the target information to the short-term-memory through Server A or B, depending on the type of the target information. Server A stores visual and spatial information, whereas Server B does phonological information. Once this operator is activated at Server D, the entity enters Server A or B. For the finger-drag task in this study, the entity went to Server B (a phonological-loop server), because a text was used as a visual stimulus, not color or other types.

Retrieve from short-term-memory. This operator allows the model to retrieve target information from the short-term-memory and makes the retrieved value be available for other operators. Similar to the "*Store to short-term-memory*" operator, once this operator is activated at Server D, the entity enters Server A or B. Then, the entity would enter Server C (a central-executor server), if operators for cognitive central executions (e.g., "*Compare*" or "*Compute*") follow after this "*Retrieve from short-term-memory*". For the finger-drag task in this study, Server B was used to retrieve the text information.

Compare. The model compares the target and expected values, using this operator. The type of information can be either text or color. If the values are identical, the result of 1 is returned (going to the next TC), otherwise, 0 is returned (going back to the TC of visual perception). Once this operator is activated at Server D, the entity enters Server C to (1) either only conduct the comparison if the information type is color; (2) or further route the incoming cognitive entity to Server F (a cognitive-complex function server) if the information type is text. In this finger-drag task, Server C was used, because a green circle was used as a visual stimulus.

Determine-hand-movement/determine-finger-movement. These operators allow the model to determine whether to move the hand and finger. Once these operators are activated at Server D, the entity enters Server C and the model determines whether or not to move their hand and finger. Then, Server C would route the entity to Server W (a motor-elements server), if operators for motor executions (e.g., "*Reach-with-hand*" or "*Click-with-finger*") follow after these "*Determine-hand-movement*" or "*Determine-finger-movement*" operators.

2.2.3. Motor operators

Reach-with-hand. This operator initiates a reaching action using the model's hand servers. Once this operator is activated at Server D, a motor entity is created in Server W with the motor type of "*Reach-with-hand*". This motor entity is then processed in Servers W, Y, Z, and right-hand or left-hand servers. The hand servers make the hand be reached to the target, based on the estimation of how far/long the hand reaches the target. The general Fitts' law equation is used to determine the reaching execution time. According to Shannon formulation (MacKenzie & Buxton, 1992), the movement time *MT* is:

$$MT = a + b \times \log_2\left(\frac{A}{W} + 1\right) \quad (1)$$

where *a* and *b* are empirical regression coefficients, varying in the environment (e.g., people and devices). *W* refers to the target's effective size, whereas *A* refers to the distance to the target. According to Wobbrock, Cutrell, Harada, and MacKenzie (2008), the effective size of *W* (*W'*) is:

$$W' = \min(Height,\ Width) \times \frac{Z_{Fitts}}{W} \quad (2)$$

and 4% of error rate is assumed in Fitts' law. Since the MacKenzie and Buxton (1992)'s original model included a mouse clicking time, which corresponds to 260 msec, according to ACT-R 6.0 model, this amount of time was subtracted. The 70 msec of motor preparation time was also deducted to determine the execution time of the "*Reach-with-hand*" operator. For the finger-drag task in this study, we assumed that the model uses a right hand to reach the target. Also, we assumed the reaching distance is 300 mm, which closely resembles the actual distance found from the pilot test. *a* = 230 (msec), *b* = 166 (msec/bit) was used from MacKenzie and Buxton (1992). Each participant's index finger breadth (i.e., IB × IB) was set as the target's height and width in this study.

Drag-with-finger. This operator initiates a dragging action using the model's hand servers. Once this operator is activated at Server D, a motor entity is created in Server W with the motor type of "*Drag-with-finger*". Similar to the "*Reach-with-hand*" operator, this motor entity is then processed in Servers W, Y, Z, and Right-hand or Left-hand servers. The hand servers make the finger be dragged on the target, based on the estimation of how far/long the finger-drags on the target. Note that this motor operator is directly associated with only TC 15, not any other TCs. In the current study, the dragging execution time (i.e., the "*Drag-with-finger*" operator's execution time) was determined by the regression models derived from hand-finger-related anthropometric data and eight different angular directions (i.e., the outcomes from Section 3.5). The regression equations for the finger movement time of drag-gesture are shown in Table 2.

Table 2. Predictive regression equations for the finger movement time of drag-gesture.

Angular direction (degree)	Predictive regression equations (milliseconds)	Adjusted R^2	RMSE	M ± *SD* (milliseconds)
0	1431–2610 *TB* + 3111 *IB*	.70	.23	1127 ± 422
45	2552–2643 *TB* + 649 *STL*	.59	.40	1246 ± 621
90	62–2358 *TB* + 3662 *IB*	.52	.34	1177 ± 490
135	3375–3191 *TB* + 697 *STL*	.55	.50	1252 ± 759
180	1208–2153 *TB* + 2615 *IB*	.68	.20	1030 ± 361
225	2963–2547 *TB* + 531 *STL*	.57	.39	1114 ± 590
270	1171–2618 *TB* + 2349 *IB* + 107 *FS*	.71	.23	1147 ± 435
315	-–1749–4508 *TB* + 4289 *IB* + 768 *IL*	.56	.64	1573 ± 973

The details of this operator's development process are described in Section 3.

2.2.4. Procedural flow operators

Decide. It specifies the procedural sequences of the steps in a list of task. Once this operator is activated at Server D, it makes the task steps specified in its parameters depending on the result of the previous step.

Goal-accomplished. This operator indicates the completion of a task. Once this operator is activated at Server D, it verifies whether there is any other pending task, and switches to that task if any task remains. If no other pending task is remaining, this operator allows the model to terminate the simulation.

3. Motion-tracking data collection for "*drag-with-finger*" operator development

To develop a computational model of touchscreen drag gestures, the "*Drag-with-finger*" operator was developed in this study because this operator did not exist in the previous QN-models, while other operators (e.g., "*Look-at*", "*Reach-with-hand*", and "*Compare*") were adopted from previous QN-modeling studies (Feng, Liu, & Chen, 2017; Jeong & Liu, 2017b).

As highlighted in Figure 1, this new motor operator is associated with not only most of the motor servers (i.e., W–Z and hand servers) in the QN-MHP, but also a cognitive server, Server D (a procedural long-term memory server that stores the whole task procedure/components, thus all the operators are associated with this server).

3.1. Participants

Eleven participants (6 males and 5 females) ranging in age between 20 and 30 years (*M* = 25.5, *SD* = 2.9) were recruited from the campus at the University of Michigan. All

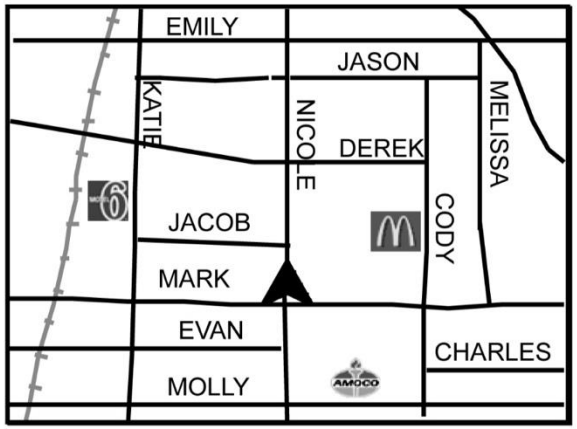


Fig. 3. Sample map for the in-vehicle task.

with the driving simulator; the testing methods and results are described in the following section.

5.1 In-vehicle Map Reading Task

In-vehicle displays are typically used to present spatial or complex information that is difficult to convey with auditory messages. As an example, navigation systems commonly use in-vehicle displays to show the driver a map, directions to a destination, and selection menus. Even though it is safer and faster to use these systems while parked, many drivers glance at the map while driving because they can manage the timing of glances away from the road without detrimental effects to their driving.

The primary difference between performing a map reading task while parked or at the office and performing it while driving is the time pressure that is imposed by the driving task, which dictates the times the driver looks at the in-vehicle display and at the road. A further difference is that the consequences of making a mistake while driving could be fatal. It is for these reasons that the human computer interaction aspects of in-vehicle systems are interesting and need to be studied.

The in-vehicle task that was chosen for modeling was a map viewing task, as illustrated in Figure 3. A driving simulator experiment was conducted by Tsimhoni and Green [2001] to examine the effects of map viewing on driver performance. The driver's task was to look at a map and search for a street name on which an icon was located. The icon was referred to by a category to which it belonged. Three categories of icons (hotel, fast food restaurant, and gas-station) commonly found on real maps were used in the experiment, in which 120 distinct maps were used and no map was presented to a subject more than once. Each map consisted of 12 streets (six horizontal and six vertical), one river, and one railroad track, which is typical for U.S. maps [George-Maletta et al. 1998]. The street names were taken from a list of 100 most popular baby

Table I. GOMS-Style Task Description of In-Vehicle Map Reading

```
GOAL: Do medium duration task

Method for GOAL: Do medium duration task
        Step 1. Find a <category>
        Step 2. Find the street name
        Step 3. Cease //task completed

Method for GOAL: find <category> //example: find restaurant
        Step 1. Accomplish goal: Is icon 1 a <category>
        Step 2. Decide: If location of <category> in memory then, Move to step 7
        Step 3. Accomplish goal: Is icon 2 a <category>
        Step 4. Decide: If location of <category> in memory then, Move to step 7

        Step 5. Accomplish goal: Is icon 3 a <category>
        Step 6. Decide: If location of <category> in memory then, Move to step 7
        Step 7. Return with goal accomplished

Method for GOAL: Is icon <number> a <category>
        Step 1. Watch for icon <number> on the map
        Step 2. Compare icon with <category> icons in long-term memory
        Step 3. Decide: If Match, (icon represents a known <category>) Then retain location
        Step 4. Return with goal accomplished

Method for GOAL: find street name
        Step 1. Watch for the street name next to location of icon
        Step 2. Read the street name
        Step 3. Retain the street name
        Step 4. Return with goal accomplished
```

names (www.babycenter.com/babyname/names98.html) updated to 1998. The maps were 15.9 cm (6.25 inch) diagonal (4:3 aspect ratio), approximately the size of contemporary in-vehicle displays.

To model this task with the QN-MHP, a GOMS-style description of the map task was devised. The GOMS-style analysis appears in Table I. The coordinates and orientation of all icons and street names on the map, and the category of the requested icon were loaded into the environment arrays representing the task situation. As an approximation to the order by which icons were searched, a left-right and top-down order was simulated. Icons that appeared on the top left of the display were fixated first and icons on the bottom right were fixated last. It should be noted that the focus of this model was not on the details of the in-vehicle task, but rather on testing the feasibility of the concurrent processing in QN-MHP.

5.2 The QN-MHP Model of Vehicle Steering

The input, output, and processing logic of the QN-MHP vehicle steering model are summarized in Table II. They are defined and developed on the basis of several major concepts and findings in the driver performance literature.

The inputs to the steering model consist of vehicle heading, vehicle lateral position, and road curvature. These inputs are made available to the simulated driver through focal and ambient vision, which occur concurrently. Perception for steering is based on two concepts: the role of focal and ambient visual systems in [illegible], and the [illegible] of near as tonic to tomy. The role

Table II. Description of the Steering Model
(References to QN-MHP servers correspond to Figure 2)

Inputs	
Vehicle heading	Vehicle heading relative to the road is retrieved as input to server 1 when fixating on a far point down the road, 4 s in front of the driver.
Lateral position	Lateral position relative to the center of the lane is retrieved as input to server 1 when one of three near points (about 1 s in front of the driver, one at the lower center and two near each of the lane markers) is in the peripheral vision of the driver.
Road curvature	On curves and approaches to curves, road curvature is retrieved as input to server 1 when the eye is fixated on a far point about the point of tangency of the curve
Outputs	
Hand position	The model outputs coordinates of hand position as the hand moves the steering wheel. (To interface with the driving simulator, hand position is converted to corresponding steering angle by an intermediate module.)
Eye position	Coordinates of the eye fixation point in space are output whenever the eye (server Z) is moved (by server Y).
Processing Logic	The main goal of maintaining the lane consists of subgoals for detecting the orientation parameters of the vehicle, selecting a steering strategy, and steering the vehicle, correspondingly.
Detecting orientation	A 'watch for' cognitive command at server D directs the model's visual attention (Servers A and C issue commands to, and wait for, proper types of entities from servers 1-4) to a far point when about to retrieve heading and curvature, and to a near point when about to retrieve lateral position. The eye is not moved if the near point information is accessible from peripheral vision. Otherwise, the eye is moved to that point using a saccade.
Selecting a steering strategy	Steering actions are selected based on the orientation of the vehicle within a look-ahead time (a parameter currently defined as 1 s) as calculated in server F using the following logic: If the vehicle's orientation within the look-ahead time is close to the center of the lane (± 0.1 m), no action is taken. Otherwise, if it is within the lane boundaries, a normal steering action is initiated, or if it is outside the lane boundaries, an imminent steering action is initiated.
Steering action	A new steering angle is calculated at server F and executed by servers V, W, X, and Y as a function of the orientation at look-ahead time and the selected steering strategy (normal or imminent). Normal actions are characterized by a steering movement to the new steering angle, and then back to a neutral position. Imminent actions are characterized by a larger magnitude of steering angle and a shorter interval, until the steering wheel is returned to its neutral position.

of focal and ambient systems in driving was first identified by Mourant and Rockwell [1970] and Leibowitz and Owens [1977] and further studied and confirmed in subsequent studies [Owens and Tyrell 1999; Summala 1998]. Visual guidance for steering was found to primarily depend on ambient vision rather than focal vision. Drivers can steer a vehicle with information perceived through peripheral vision. Visual recognition abilities, such as those needed for detecting a yield sign or other hazards, were found to depend on focal vision.

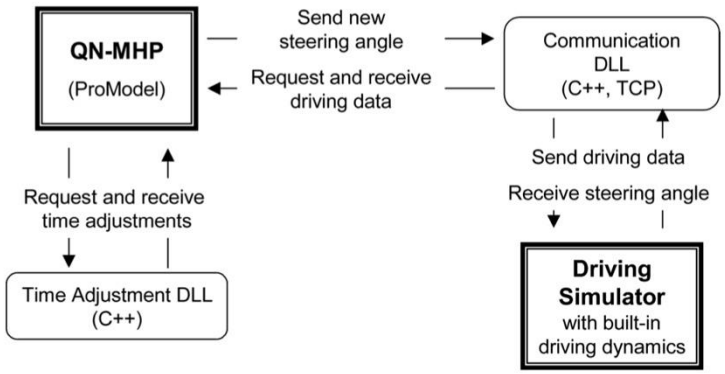


Fig. 4. Block diagram of the interface between the QN-MHP and the driving simulator.

QN-MHP driving model was interfaced with the DriveSafety Research Simulator, a high fidelity driving simulation system used for driving research and training. It utilizes a driving dynamics model that can be adjusted to simulate a variety of vehicle types. It keeps track of numerous state-variables and can output them to external devices. For communication with external devices, the driving simulator uses TCP/IP protocol. Although normally operated via a steering wheel and pedals installed in a simulated car, the driving simulator can also be controlled externally by digital inputs.

Communication between the QN-MHP driving model and the driving simulator (Figure 4) was implemented via a TCP/IP host, linked to the ProModel program (as a DLL). The QN-MHP driving model sent and received data as function calls, directly to the DLL. The driving simulator sent and received data by a TCP/IP client that communicated with the host.

Whenever the QN-MHP model made a glance to a specific position in the road scene and information was assumed to be available to it, corresponding information was retrieved from the communication thread. Whenever a hand or eye movement was made by the model, the intermediate or final steering wheel position and the area of fixation were output to the driving simulator via the communication thread. The driving simulator retrieved steering angle and eye position continuously to keep the virtual steering wheel at the desired position and show the area of fixation overlaid on the road scene.

QN-MHP was successful in steering the driving simulator. Figure 5 shows the physical layout. The simulated vehicle remained within the lane boundaries of straight sections and curves of varying radii. Transfer of information between the software modules of the system was smooth, and timing delays were short. Quantitative validation of the steering model is reported later in the empirical validation section of this article.

5.4 Dual Tasks—Modeling

The steering task and the lane-changing task were implemented as two independent goals. The steering goal was performed continuously as long

Fig. 5. QN-MHP steering the driving simulator in real-time. A short movie clip can be seen on the website: http://www-personal.engin.umich.edu/~yililiu/cogmodel.html.

as the vehicle was in motion. The in-vehicle task was initiated at intervals of 30 s and stopped after completion. Both goals were processed simultaneously except when there was a conflict at the server level. The most notable conflict for these two goals was over visual resources. The conflict occurred when each goal instructed to move the eye to a different location. When such conflicts occurred, priority was given to one of the goals based on the driving situation and the involvement in the in-vehicle task. More specifically, a satisficing rule based on time to line crossing (TLC) was generated such that the eyes were not diverted away from the road unless TLC was above a certain threshold (4 s). TLC represents the time available for a driver until the moment at which any part of the vehicle reaches one of the lane boundaries [Godthelp 1984]. The calculation used in the model was a simplified version of TLC calculation based on the lateral position and its derivatives, which has been shown to be an acceptable predictor of lane departures due to inattention [van Winsum et al. 2000]. When the eyes were already directed at the in-vehicle display, TLC was estimated by the steering task, and the eyes glanced back to the road when TLC reached the threshold.

6. EMPIRICAL VALIDATION OF THE QN-MHP DRIVING MODEL

6.1 In-vehicle Task Performance

In-vehicle task performance was validated for the baseline condition in which the subject performed the task without driving and the model had only the

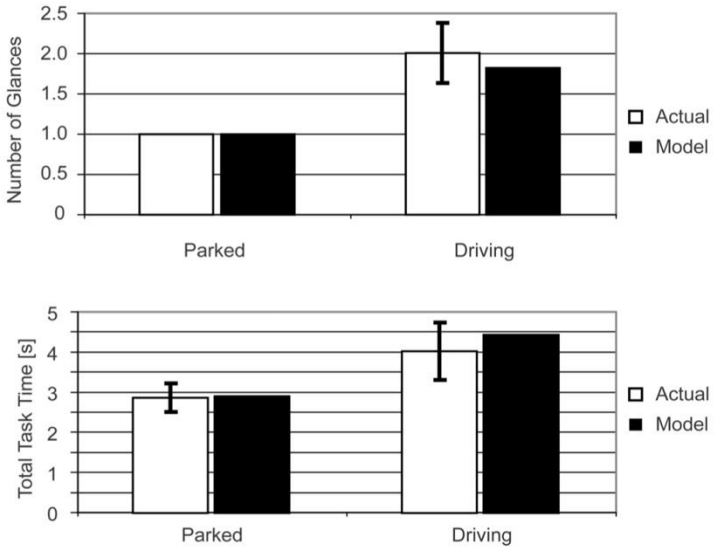


Fig. 9. Model predictions of (a) task timing and (b) number of glances.

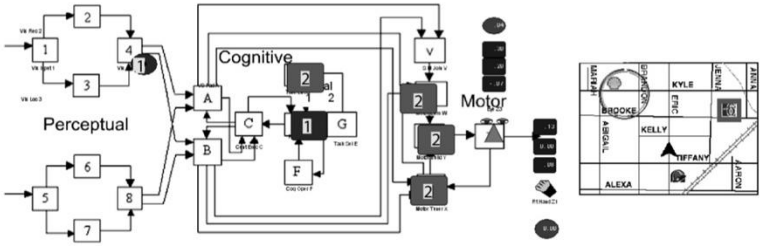


Fig. 10. Screenshot of QN-MHP in action. A visual entity is about to be processed by server A as two concurrent tasks are being processed in the cognitive subnetwork. The eye is looking at the map but a steering action is still underway.

task. A visual entity is about to be processed by server A (visuospatial sketchpad) as the two concurrent tasks (driving and reading a map) are processed concurrently in the cognitive subnetwork (servers D and E—goal procedures performance monitoring, respectively). The eye has already been moved away from the road scene to the map (represented by circle on top left of map) but a steering action is still underway in the motor subnetwork.

7. DISCUSSION AND CONCLUSIONS

As shown in this article, our modeling approach, called the Queueing Network-Model Human Processor (QN-MHP) is significantly different but very complementary to the existing approaches. As a model of multitask performance, a unique characteristic of the QN-MHP is its ability to model concurrent activities

INTL. JOURNAL OF HUMAN–COMPUTER INTERACTION, *25*(6), 554–581, 2009

ISSN: 1044-7318 print / 1532-7590 online
DOI: 10.1080/10447310902973182

Taylor & Francis
Taylor & Francis Group

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

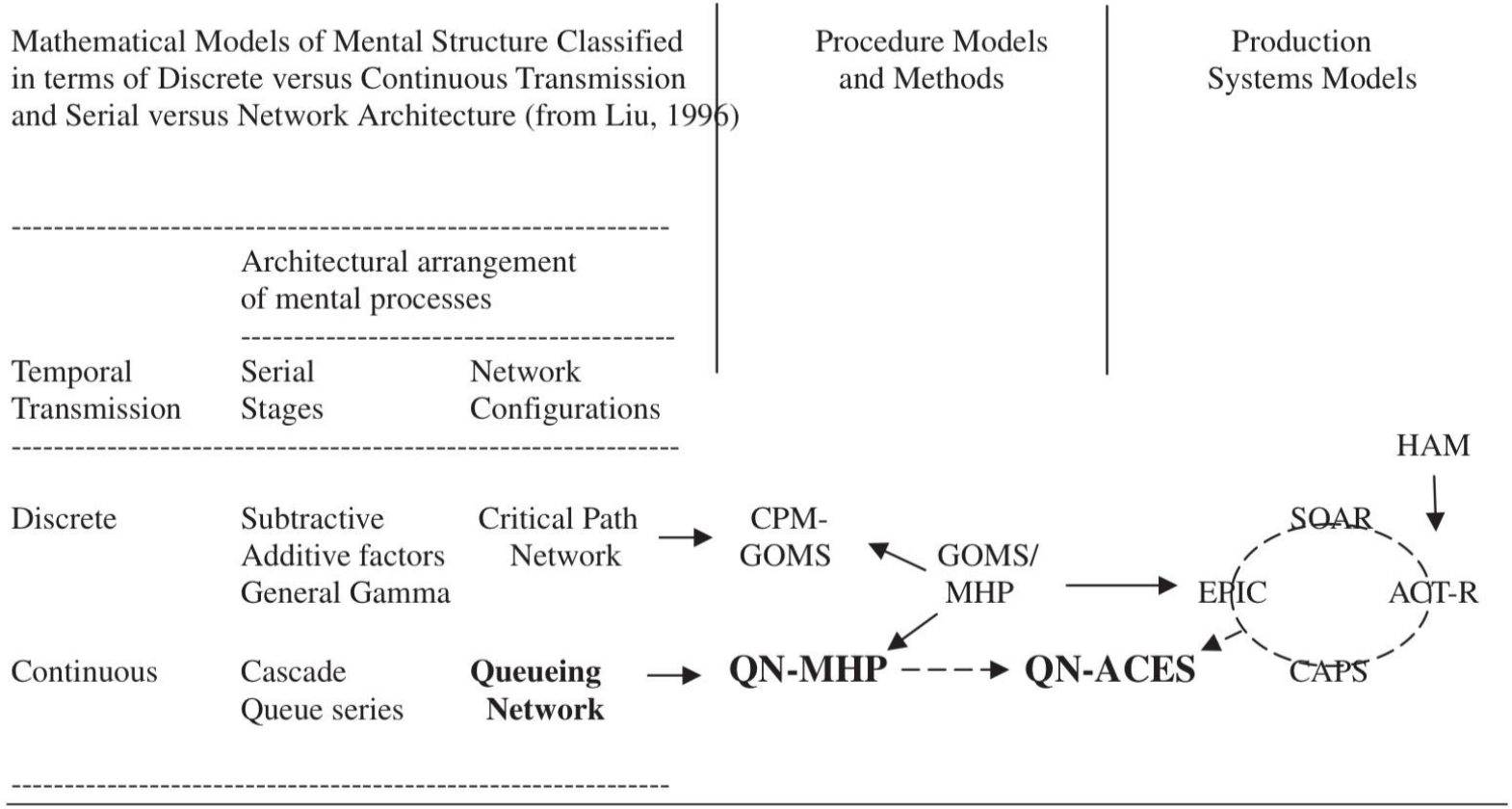


FIGURE 2 Mathematical and symbolic models of mental architecture (Liu, 2006; Liu et al, 2006) showing the relationship between QN, QN-MHP, QN-ACES and a sample of influential cognitive architectures. *Note*: By integrating the complementary schools of mathematical (left half of the figure) and symbolic (right half) models, QN-MHP supports both precise mathematical analysis and real-time generation of behavior, thus capitalizing on the strength and overcoming the weaknesses of either mathematical or symbolic modeling.

QUEUEING NETWORK MODELING OF VISUAL SEARCH

by

Ji Hyoun Lim

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Industrial and Operations Engineering)
in The University of Michigan
2007

Doctoral Committee:

Associate Professor Yili Liu, Chair
Associate Professor Bernard J. Martin
Associate Professor Jun Zhang
Adjunct Assistant Professor Omer Tsimhoni

QUEUEING NETWORK MODELING OF VISUAL SEARCH

by

Ji Hyoun Lim

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Industrial and Operations Engineering)
in The University of Michigan
2007

TABLE OF CONTENTS

LIST OF FIGURES.....vi
LIST OF TABLES.....ix
ABSTRACT.....x

CHAPTER 1
VISUAL SEARCH MODELS AND QUEUEING NETWORK ARCHITECTURE

1.1. Motivation.....1
1.2. Existing Computational Models of Visual Search.....2
1.2.1. Computational models specialized for visual search.....2
1.2.2. Visual search models based on unified cognitive architectures.....4
1.3. The QN-MHP: Computational Cognitive Model of Queueing Network.....7
1.4. Research Objective.....8

CHAPTER 2
CONVERSION OF VISUAL STIMULUS:
FROM NATURAL SCENES TO DATA ARRAYS

2.1. Introduction.....11
2.2. Computational Framework for Vision.....13
2.2.1. Basic vision and visual stimulus.....13
2.2.2. Existing computational frameworks of the early vision.....15
2.3. Representations of Visual Stimulus for Queueing Network Model.....17
2.4. Implementation of a Conversion System.....20
2.4.1. Size of a visual stimulus array.....20
2.4.2. Features for the early vision.....21
2.4.3. Data array for a moving visual stimulus.....23
2.5. Conclusion.....24

CHAPTER 3
QUEUING NETWORK MODELING OF
VISUAL SEARCH AND MENU SELECTION

3. 1. Introduction.....25

3. 2. Existing Experimental and Modeling Efforts on Menu Search 27

3. 3. Queueing Network Modeling of Menu Search.....36

3. 3. 1. Queueing Network – Model Human Processor (QN-MHP).....37

3. 3. 2. Modeling menu search task with QN-MHP.....39

3. 4. Simulation Results and Discussions.....50

3. 4. 1. Eye overshooting.....51

3. 4. 2. Varying length in saccade.....54

3. 4. 3. Target position effect.....59

3. 5. Conclusion.....61

CHAPTER 4
REINFORCEMENT LEARNING IN EYE MOVEMENTS:
MODELING THE INFLUENCES OF
CYCLIC TOP-DOWN AND BOTTOM-UP PROCESSES

4. 1. Introduction.....64

4. 2. Relationship between Attention and Saccades.....66

4. 3. Computational Models of Reinforcement Learning.....69

4. 4. Eye Movements and Reinforcement Learning.....74

4. 5. Effect of Top-Down Processes on Eye Movements
– Yarbus' Picture Viewing Task.....80

4. 6. Effect of Bottom-Up Processes on Eye Movements
– Findlay's Visual Search Task.....85

4. 7. Discussion and Conclusion.....91

CHAPTER 5
EXPERIMENTAL STUDY:
ANALYSIS OF GLANCE BEHAVIOR IN PEDESTRIAN DETECTION
USING A NIGHT VISION ENHANCEMENT SYSTEM WHILE DRIVING

5.1. Introduction.....94

5.2. Study 1: Relationship between the Intensity of Clutter
and the Subjective Rating of Clutter.....98

5.2.1. Method.....99

5.2.2. Clutter metrics for the visual stimulus.....100

5.2.3. Results.....103

5.2.4. Conclusion and discussion.....108

5.3. Study 2: Pedestrian Detection while Driving.....111

5.3.1. Tasks and stimulus.....111

5.3.2. Glance analysis.....113

5.3.3. Results.....116

5.3.4. Conclusion and discussion.....121

5.4. Summary.....124

CHAPTER 6
EFFECT OF A CONCURRENT TASK ON VISUAL SEARCH:
STUDY WITH COMPUTATIONAL MODEL OF
VISUAL SEARCH AND TRACKING DUAL TASK

6.1. Introduction.....126

6.2. Existing Studies on Dual Task.....128

6.3. Queueing Network Model
for Dual Task of Pedestrian Detection and Driving.....132

6.3.1. Pedestrian detection model.....132

6.3.2. Simple driving model.....141

6.3.3. Managing two tasks in the QN-MHP.....144

6.4. Simulation Results and Discussion.....146
6.4.1. Results from reinforcement learning: different strategies from different visual stimulus.....147
6.4.2. Single task: pedestrian detection.....150
6.4.3. Dual task: pedestrian detection and simple driving.....151
6.5. Conclusion and Discussion.....158

CHAPTER 7
CONCLUSIONS AND DISCUSSIONS

7.1. General Summary.....161
7.2. Contributions of This Work.....164
7.3. Limitations of This Work and Future Research.....169

REFERENCES.....172

Computational Modeling and Experimental Research on Touchscreen Gestures, Audio/Speech Interaction, and Driving

by

Heejin Jeong

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Industrial and Operations Engineering)
in the University of Michigan
2018

Doctoral Committee
Professor Yili Liu, Chair
Professor Melissa Gross
Associate Professor Bernard J. Martin
Professor Nadine B. Sarter

Computational Modeling and Experimental Research on Touchscreen Gestures, Audio/Speech Interaction, and Driving

by

Heejin Jeong

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Industrial and Operations Engineering)
in the University of Michigan
2018

Table of Contents

Dedication ii

Acknowledgements iii

Lists of Tables vii

Lists of Figures viii

Abstract xi

Chapter 1. Introduction 1

1.1. Overview 1

1.2. Research Objectives 4

1.3. Queueing Network – Model Human Processor 4

1.4. Contributions 9

1.5. Dissertation Structure 10

References 12

Chapter 2. An Experimental Investigation on the Effects of Touchscreen Gesture Type and Direction on Finger-touch Input Performance and Subjective Ratings 15

2.1. Introduction 15

2.2. Method 17

2.3. Results 24

2.4. Discussion 35

2.5. Implications and future studies 39

References 41

Chapter 3. Queuing Network Modeling of Touchscreen Drag Gestures using Motion Tracking 43

3.1. Introduction 43

3.2. Queueing-network modeling of human performance 47

3.3. Motion-tracking data collection for "Drag-with-finger" operator development 54

3.4. Model validation 59

3.5. Discussion 61

References 65

Chapter 4. An Experimental Study on the Effect of In-vehicle Secondary Task Modalities and Road Curvature on Eye Movements and Driving Performance 68
4.1. Introduction 68
4.2. Literature review 70
4.3. Method 75
4.4. Results 83
4.5. Discussion and conclusions 92
References 96
Chapter 5. Queuing Network Modeling of In-vehicle Secondary Task Performance 101
5.1. Introduction 101
5.2. Model development 102
5.3. Model validation 107
5.4. Conclusion and discussion 108
References 110
Chapter 6. An Experimental Study on the Effects of Road Geometry and Lead Vehicle on Driving Performance 112
6.1. Introduction 112
6.2. Methods 113
6.3. Results 116
6.4. Discussion 119
6.5. Conclusions 121
References 122
Chapter 7. Queuing Network Modeling of Driver Lateral Control on Curved Roads with Integration of Vehicle Dynamics and Reference Trajectory Tracking 124
7.1. Introduction 124
7.2. Computational modeling of lateral control 125
7.3. Validation results 130
7.4. Discussion 132
References 133
Chapter 8. Development and Evaluation of a Computational Cognitive Model for In-vehicle Direct/Indirect Manual and Speech Interactions 135
8.1. Introduction 135
8.2. Background 137
8.3. Model Development 139

8.4. Model Evaluation 144
8.5. Discussion 146
References 148
Chapter 9. Conclusion and Future Research 152
9.1. Conclusion 152
9.2. Summary of each chapter 153
9.3. Future research 156

Queuing Network Modeling of Human Multitask Performance and its Application to Usability Testing of In-Vehicle Infotainment Systems

by

Ruijia Feng

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Industrial and Operations Engineering)
in the University of Michigan
2015

Doctoral Committee:

Professor Yili Liu, Chair

Associate Professor Victoria Booth

Associate Professor Bernard J. Martin

Professor Nadine B. Sarter

Queuing Network Modeling of Human Multitask Performance and its Application to Usability Testing of In-Vehicle Infotainment Systems

by

Ruijia Feng

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Industrial and Operations Engineering)
in the University of Michigan
2015

TABLE OF CONTENTS

DEDICATION ii

ACKNOWLEDGEMENTS iii

LIST OF FIGURES vi

LIST OF TABLES ix

Abstract x

Chapter 1. Introduction 1

1.1 Computational Human Performance Modeling 1

1.2 Queuing Network-Model Human Processor 3

1.3 Scientific Merits and Broad Impact 7

1.4 Dissertation Structure 8

Chapter 2. A Driver Steering Model Using the QN-MHP 10

2.1 Introduction 10

2.2 Methods 12

2.3 Results 17

2.4 Discussions 20

Chapter 3. An Experimental Investigation on the Effects of Number of Buttons and Button Size on Visual Search Task Performance while Driving 22

3.1 Introduction 22

3.2 Methods 23

3.3 Results 28

3.4 Conclusions 43

Chapter 4. Queuing Network Modeling of Visual Search Task Performance while Driving 45

4.1 Introduction 45

4.2 Methods 46
4.3 Results 52
4.4 Discussions 54
Chapter 5. An Experimental Study on Task Performance and Mental Workload in Using Three Typical In-Vehicle Infotainment Systems while Driving 57
5.1 Introduction 57
5.2 Methods 60
5.3 Results 64
5.4 Discussions 68
Chapter 6. Queuing Network Modeling of Visual-Manual Secondary Tasks while Driving 70
6.1 Introduction 70
6.2 Methods 71
6.3 Results 79
6.4 Discussions 81
Chapter 7. A Computer-Aided Usability Testing Tool for In-Vehicle Infotainment Systems 83
7.1 Introduction 83
7.2 Software Development 86
7.3 Software Outputs 93
7.4 Discussions 96
Chapter 8. Conclusion 98
8.1 Dissertation Summary 98
8.2 Conclusions 100
8.3 Future Research 101
APPENDIX 104
BIBLIOGRAPHY 107

(a) Tuning radio using the knob

(b) A zoom-in view of the physical panel

Figure 31. Task Procedure using the knob: (1) press the power button, (2) press the AM/FM button, (3) turn the knob to decrease or increase the frequency shown on the display ("590" as in the picture)

(a) Tuning radio on the touch screen

(b) Click the "Entertainment" button

(c) Click the "FM" then "Direct Tune" button

(d) Enter the radio frequency

Figure 32. Task procedure using the virtual buttons

Table 4. Task procedures of tuning the radio to FM 98.7 using the three methods

Device	**Physical Buttons**	**Physical Knobs**	**Virtual Buttons**
Initial state	System OFF	System OFF	Home Screen
Step 1	Press 'Power'	Press 'Power'	Press 'Entertainment'
Step 2	Press 'AM/FM'	Press 'AM/FM'	Press 'FM1'
Step 3	Press 'DIRECT'	Turn the knob to 98.7	Press 'Direct Tune'
Step 4	Press '9'	*Done*	Press '9'
Step 5	Press '8'		Press '8'
Step 6	Press '7'		Press '7'
Step 7	*Done*		Press "Enter"
Step 8			*Done*

Driving task: The participants were asked to drive the simulated vehicle on a virtual highway. The virtual highway has two lanes in one direction, and the virtual course is a square loop with four straight sections connected by four curved corners. The driving environment is set as day time. The participants were asked to keep the vehicle in the left lane and maintain a speed of between 60-70 miles per hour. There is no other virtual vehicle in the left lane. The participants were asked to put both hands always on the steering wheel except when doing the radio-tuning task.

5.2.3 Experimental Design

There are two independent variables in this experiment: (1) Task condition (two levels: single or dual task), (2) Control modules (three levels: physical buttons, knobs, or virtual buttons).

Independent variables: Two task conditions were examined in the experiment: (1) single task, and (2) dual task. In the single task condition, the simulated vehicle is stopped on the side of the road. The participants were instructed to perform the radio-tuning task without driving the simulator. In the dual task condition, the participant was asked to drive the simulator in a highway scenario, and at given points, they were verbally instructed by the experimenter to start the radio-tuning task. The experimenter only instructs the participants to start the task when the

Table 7. Summary of some existing cognitive modeling tools

Tools	Theoretical framework	Modeling multitask	Software GUI	UI prototyping	Software requirements	References
QN-MHP MATLAB	QN-MHP	Yes	Yes	Yes (MATLAB GUIDE)	MATLAB/Simulink	Feng, et al, 2014
QN-MHP with VBA	QN-MHP	Yes	Yes	Rudimentary with static images only	ProModel + VBA in Excel	Wu & Liu, 2009
QN-ACTR	ACT-R, QN	Yes	Yes	Yes	Micro Saint	Cao & Liu, 2012
ADAT	SEEV and N-SEEV attention models	Yes	Yes	No	-	Sebok, et al., 2012
GLEAN	EPIC	No	Rudimentary menu interface	No	Macintosh Common Lisp	Kieras, et al., 1995
CRITIQUE	KLM	No	Yes	Yes (subArctic toolkit)	-	Hudson, et al., 1999
APEX	GOMS	Yes	-	-	Standalone software	Freed, et al, 2003
ACT-Simple	ACT-R	No	No	No	-	Salvucci & Lee, 2003
Distract-R	ACT-R	Yes	Yes	Yes	Standalone software	Salvucci, et al, 2005
CogTool	KLM + ACT-R	No	Yes	Import HTML	Standalone software	John, et al., 2004
E-GOMS	GOMS	No	Yes	-	-	Gil, 2010

Queueing Network Modeling of Human Performance in Complex Cognitive Multi-task Scenarios

by

Shi Cao

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Industrial and Operations Engineering)
in the University of Michigan
2013

Doctoral Committee:

Professor Yili Liu, Chair
Professor Benjamin J. Kuipers
Associate Professor Bernard J. Martin
Professor Nadine B. Sarter

Queueing Network Modeling of Human Performance in Complex Cognitive Multi-task Scenarios

by

Shi Cao

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Industrial and Operations Engineering)
in the University of Michigan
2013

Table of Contents

Dedication ii

Acknowledgements iii

List of Figures viii

List of Tables xi

Abstract xiii

Chapter 1. Introduction 1

Chapter Summary 1

1. Human Performance Modeling (HPM) 1

2. Queueing Network (QN) Architecture of Human Performance 4

3. Adaptive Control of Thought-Rational (ACT-R) 8

4. An Integrated Cognitive Architecture to Model Cognitive Multitasking Performance 10

5. Thesis Structure 13

Chapter 2. Framework and verification of Queueing Network – Adaptive Control of Thought Rational (QN-ACTR) 15

Chapter Summary 15

1. Queueing Network-Adaptive Control of Thought Rational (QN-ACTR) 15

2. Model Verification 20

3. QN-ACTR Simulation of Transcription Typing and Reading Comprehension Tasks 23

4. Discussion 32

Chapter 3. An Experimental Investigation of Effects of Concurrent Tasks on Diagnostic Decision Making 35

Chapter Summary 35

1. Introduction 36

2. Methods 41
3. Results 46
4. Discussion 49
5. Conclusions 52
Chapter 4. Modeling Cognitive Multitasking Performance in Diagnostic Decision Making Tasks 53
Chapter Summary 53
1. Introduction 53
2. Method 56
3. Results 58
4. Discussion 62
Chapter 5. An Experimental Investigation of Concurrent Processing of Vehicle Lane Keeping and Speech Comprehension Tasks 64
Chapter Summary 64
1. Introduction 65
2. Method 71
3. Results 76
4. Discussion 79
5. Conclusions 86
Chapter 6. QN-ACTR Modeling and Simulation of Lane Keeping and Speech Comprehension Dual-task Performance 87
Chapter Summary 87
1. Introduction 87
2. Method 90
3. Results 96
4. Discussion 100
Chapter 7. Usability Development of Queueing Network-ACTR for Cognitive Engineering Applications 101
Chapter Summary 101
1. Introduction 101
2. Method 103

3. Findings 110
4. Discussion 111
Chapter 8. Conclusions and Future Research 113
1. Summary of Thesis 113
2. Conclusions 116
3. Future Research 118
References 122

Modeling Dual-Task Concurrency and Effort in QN-ACTR and IMPRINT

by

Christopher Jason Best

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Industrial and Operations Engineering)
in The University of Michigan
2013

Doctoral Committee:

Professor Yili Liu, Chair
Assistant Professor Victoria Booth
John F. Locket III, US Army Research Laboratory
Professor Nadine B. Sarter

Modeling Dual-Task Concurrency and Effort in QN-ACTR and IMPRINT

by

Christopher Jason Best

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Industrial and Operations Engineering)
in The University of Michigan
2013

TABLE OF CONTENTS

DEDICATION ii

ACKNOWLEDGEMENTS iii

LIST OF FIGURES vii

LIST OF TABLES ix

CHAPTER

I. Introduction 1

1.1 QN 4
1.2 ACT-R 7
1.2.1 Declarative Module 8
1.2.2 Procedural Module 8
1.2.3 Goal Module 10
1.2.4 Vision Module 10
1.2.5 Device Module 12
1.2.6 Motor Module 12
1.2.7 Speech Module 13
1.2.8 Temporal Module 13
1.3 QN-ACTR 13
1.4 IMPRINT 14
1.4.1 IMPRINT model structure 15
1.4.2 Workload management 17
1.5 Soar 18
1.5.1 Problem state representation 19
1.5.2 Productions 19
1.5.3 Input and Output (IO) 22
1.5.4 Learning 22
1.5.5 Using Soar in cognitive models 22
1.6 EPIC 24

II. IMPRINT and Workload Management with Soar 26

2.1 Workload management extension theory 27
2.2 Extending IMPRINT 29
2.2.1 Plugin capability 29
2.2.2 Soar plugin 30
2.3 Soar agent 36
2.3.1 Release decision subgoal 36
2.3.2 Expire decision subgoal 37
2.3.3 Resume decision subgoal 37

2.3.4 Response selection . . . 38
2.4 UAV Study . . . 38
2.4.1 Release decision . . . 40
2.4.2 Resume decision . . . 42
2.4.3 Expire decision . . . 42
2.5 Results . . . 43
2.6 Conclusion . . . 44

III. Modeling concurrency on addition and targeting tasks in QN-ACTR and IMPRINT . . . 46

3.1 QN-ACTR additions . . . 46
3.2 Tasks . . . 47
3.2.1 Targeting task . . . 47
3.2.2 Addition task . . . 48
3.3 QN-ACTR task models . . . 49
3.3.1 Addition model . . . 49
3.3.2 Targeting models . . . 57
3.3.3 Concurrency model . . . 73
3.4 IMPRINT task models . . . 74
3.5 Method . . . 76
3.5.1 Scoring . . . 77
3.5.2 Procedure . . . 77
3.5.3 Apparatus . . . 78
3.5.4 Data collection . . . 79
3.6 Results . . . 79
3.6.1 Empirical results . . . 79
3.6.2 QN-ACTR Model validity . . . 80
3.6.3 IMPRINT . . . 83
3.7 Discussion . . . 85
3.7.1 Effect of speed . . . 85
3.7.2 Execution time . . . 87
3.7.3 Concurrency . . . 89
3.7.4 Behavioral modeling . . . 92
3.7.5 Application to IMPRINT . . . 94
3.8 Conclusion . . . 96

IV. The effect of effort on dual-task performance and concurrency . . . 99

4.1 Tasks . . . 100
4.1.1 Targeting task . . . 101
4.1.2 Addition task . . . 101
4.2 QN-ACTR models . . . 101
4.2.1 Addition model . . . 101
4.2.2 Targeting model . . . 102
4.3 Method . . . 103
4.3.1 Scoring . . . 104
4.3.2 Procedure . . . 104
4.3.3 Apparatus . . . 105
4.3.4 Data collection . . . 105
4.4 Results . . . 106
4.4.1 Single tasks . . . 106
4.4.2 Dual tasks . . . 106
4.4.3 Cluster analysis . . . 107

4.5 Discussion . . . 111
4.5.1 Single task incentivization . . . 111
4.5.2 Dual task performance . . . 112
4.5.3 Concurrency predictions . . . 115
4.6 Conclusion . . . 117

V. Conclusion . . . 118

5.1 Summary of models and their roles . . . 119
5.2 Scientific contributions and Future work . . . 120

APPENDICES . . . **123**

BIBLIOGRAPHY . . . **168**

Development and Evaluation of an Ergonomic Software Package for Predicting Multiple-task Human Performance and Mental Workload in Human-Machine Interface Design and Evaluation

Changxu Wu
Department of Industrial and Systems Engineering
State University of New York (SUNY)-Buffalo

Yili Liu
Department of Industrial & Operations Engineering
University of Michigan

Abstract

Predicting human performance and mental workload in multiple task situations at an early stage of system design can save a significant amount of time and cost. However, existing modeling tools either can only predict human performance or require users of tools to learn a new programming language. Queueing Network-Model Human Processor (QN-MHP) is a new cognitive architecture for modeling both human performance and mental workload in multiple tasks. This paper describes the development of a Visual Basic Application in Excel (VBA) software package and an illustrative case study to evaluate its effectiveness. The software package has an easy-to-use user interface for QN-MHP that assists users of the modeling tool to simulate a dual task including definition of the tasks and interfaces by clicking buttons to select options and filling texts in a table, with no need to learn a simulation language. It allows the model user to intuitively observe the information processing state of the model during simulation, and conveniently compare the simulated human performance and mental workload for different designs. The illustrative case study showed that naïve users without prior simulation language programming experience can model human performance and mental workload in a complex multitask situation within 3 minutes; and this software package can save 71% of modeling time and reduce 30% of modeling errors. Further developments of the VBA software package of QN-MHP are also discussed on how to make it a comprehensive proactive ergonomic design and analysis tool.

Keywords: Ergonomics, Software package, Queueing Network, Human performance, Mental workload

human performance and mental workload. The Promodel file reads the Excel file during the simulation process.

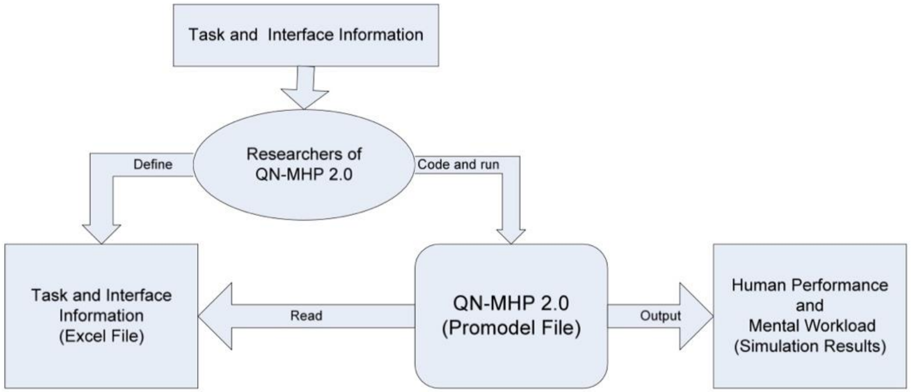


Figure 3. Original structure of the QN-MHP 2.0 without VBA software package

Overall Structure of QN-MHP 2.0 with VBA Software Package

Built on the original structure of QN-MHP 2.0, a Visual Basic Application in Excel (VBA) package was developed and this package was composed of two components (see Figure 4): an easy-to-use VBA user interface of the model and an ActiveX module (these two components together are called "VBA software package" or "modeling tool" of QN-MHP 2.0 in this paper). The role of the VBA user interface of the Model is to help users of the QN-MHP 2.0 define the task and interface information as well as automatically export the task and interface information to the Excel file. Then the information in the Excel file is exported to the ActiveX module by the software itself. The ActiveX module automatically codes the original QN-MHP 2.0 Promodel file according to the information from the Excel file and runs the Promodel file to generate the simulation results.

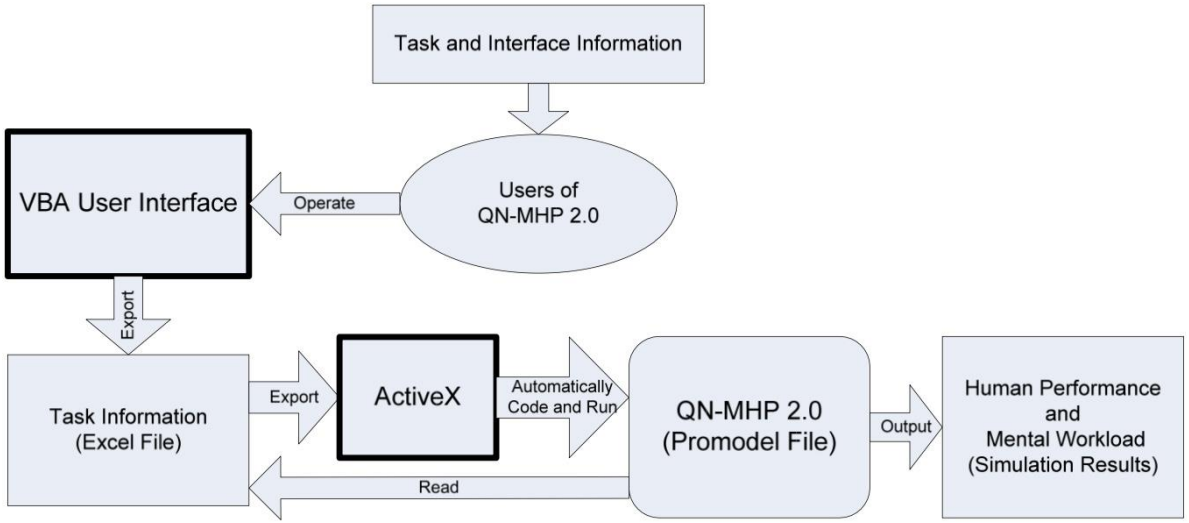


Figure 4. Overall Structure of QN-MHP 2.0 with VBA software package including a VBA user interface and an ActiveX modules (the two boxes highlighted with bold borders)

VBA User Interface

The user interface of QN-MHP 2.0 is developed using a widely used rapid UI design software: Visual Basic Application (VBA) in Excel. Figure 5 describes the flow chart in using this VBA user interface to define the task and interface information. To define a single or a dual task, users of the modeling tool need three steps to define the task and interface information using this VBA user interface. First, users have an option to choose whether the target task to be simulated is a single or dual task. After that, the users can define the single and dual tasks individually (see Figure 6 and 7).

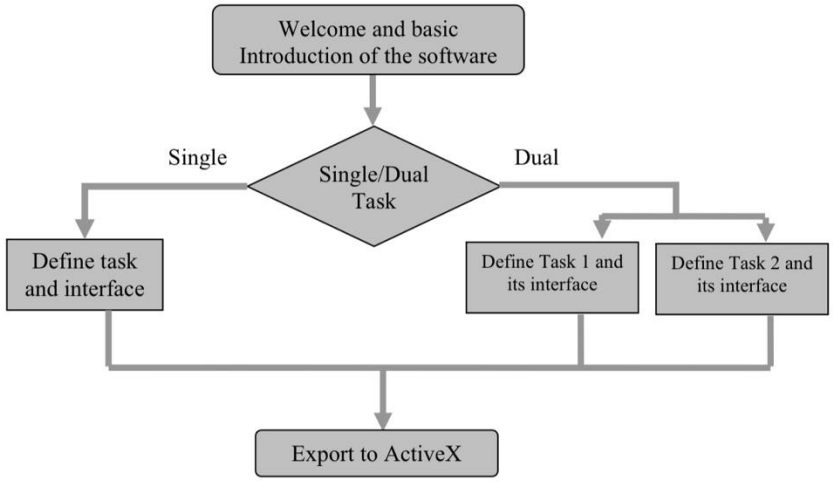


Figure 5. Flow Chart of the VBA user interface

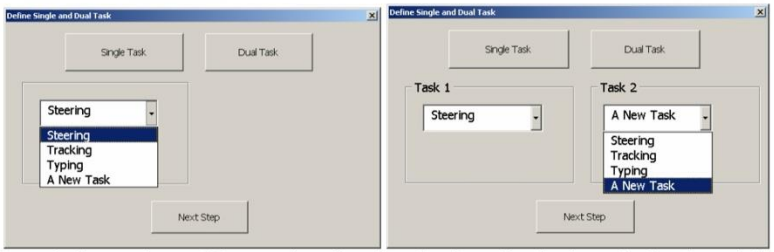


Figure 6. Screenshots of the user interface in selection of a single or dual task modeling

Second, in defining a single task, users have four options including three options to use the existing module: steering, typing and visual manual tracking and the fourth option to define the task as a new task. Similarly, in defining a dual task, users have these four options for both tasks (see Figures 6 and 7). If the "a new task" option is selected, a table automatically shows up so that the users can define the user interface mock-ups (the graphic images of the UIs and their path on the computer) and the name of objects in these user interface mock-ups (see Table 2).

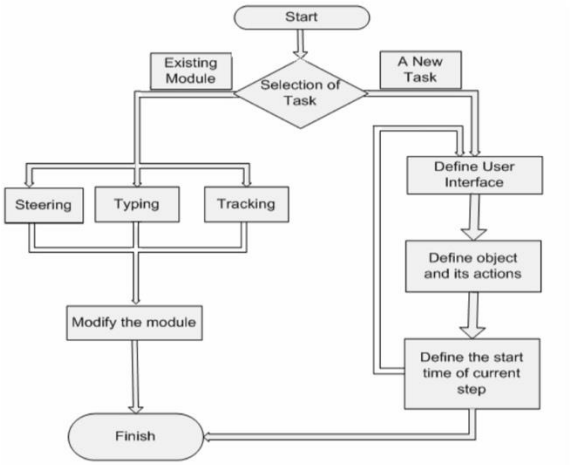


Figure 7. Flow chart of the VBA user interface in defining a task
(A single task or Task 1 or Task 2 in a dual task situation)

Table 2. A sample table for defining the user interfaces and objects in them

InterfaceID	Interface	Path	ObjectID	Object
1	UI_1	E:\UI_photos\GPS.jpg	1	Menu Button
			2	Direction Arrow
			3	Message
2	UI_2	E:\UI_photos\Plate.jpg	4	Address Button

Third, the interface automatically changes according to users' selection of the tasks (see Figures 8 and 9): Figure 8 shows the interface when users choose steering as Task 1 and a new task as Task 2 in a dual task condition. Users can define a change of UI in the image window at the left side of the dialogue box. Starting from the top on the right side of the dialogue box, users specify an object in the UI (e.g., the "menu" button) previously defined in the table (see Table 2), its sensory channel (visual, auditory, or tactile), and a series of actions or operators corresponding to that object (e.g., "look at," "reach to it (by hand)" etc.). Rather than typing a numerical code into an Excel sheet, when users demonstrate the tasks by choosing a certain action/operator, the VBA user interface automatically translates this

action/operator and store its numerical code to the Excel file. The definition of a series of actions corresponding to an object is called a step, and users specify the start time of these actions by: 1) waiting until the start of the previous step (start at the same time as the previous step), 2) waiting until several actions of the previous step occur, 3) waiting until the end of the previous step, or 4) starting independently of the previous step (users are allowed to define the start time by an event or absolute start time). Users can then click on the next step to define a series of actions corresponding to another object. In the dual task condition, after users define both tasks, users can define the priority of each task by clicking on the "Task Priority" button.

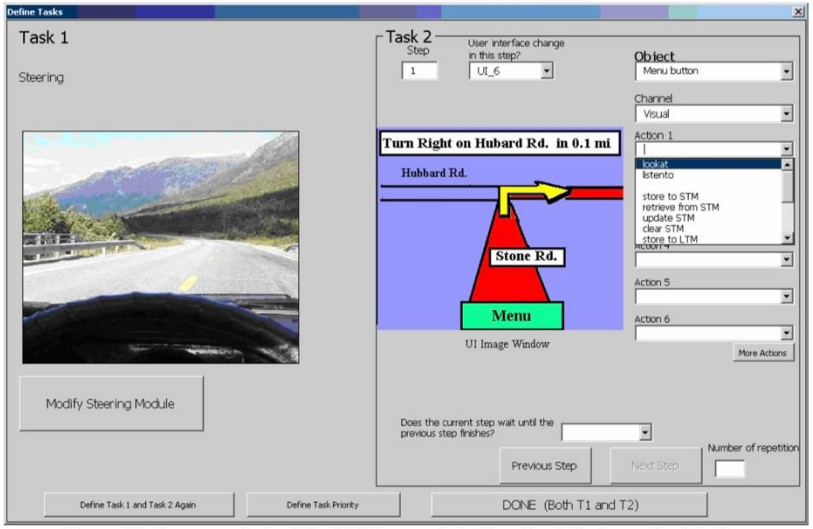


Figure 8. A screenshot of the VBA user interface in defining a dual task (Steering as Task 1 and a new task as Task 2 in a dual task condition)

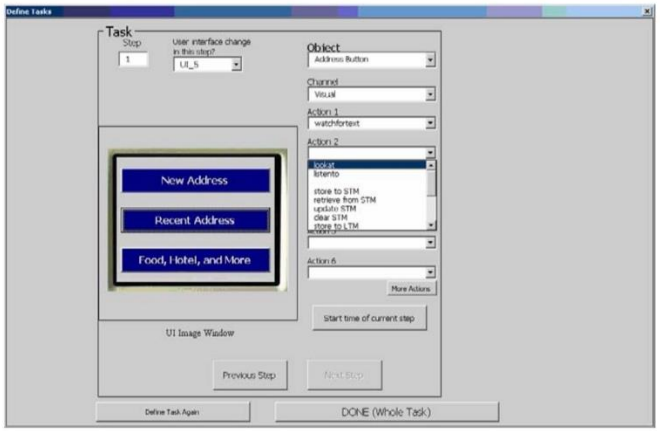


Figure 9. A screenshot of the VBA user interface of QN-MHP 2.0 in defining a single task (A new task in a single task condition)

All of the task and interface information is exported automatically to the Excel file when users press the "Done" button located at the bottom of the interface. At the same time, the information in the Excel file is also exported to the ActiveX module introduced in the following section.

ActiveX and Promodel Simulation

The ActiveX module in the software is developed based on the ActiveX module provided in the Promodel software package. The module is composed of: a) Several Excel macros and each of these macros is corresponding to a component in Promodel software (e.g., arrays, arrivals, and processing, etc.) and these macros are able to automatically communicate with the corresponding components in Promodel to update the Promodel codes (Promodel, 2003); b) A "Control" Excel sheet which manages all of these Excel macros and it contains subroutings (a set of VBA codes) controlling and activating all of the Excel macros. For example, a "Update Promodel" subrouting activates all the

macros to export codes to a Promodel file and this subrouting can be initiated by a "Update Promodel File" button on the "Control" sheet.

The function of this ActiveX module in the VBA software package is to automatically update three parts of the Promodel file (arrival, array and macro) corresponding to the information imported from the Excel file after users press the "Update Promodel File" button (see Figure 10). Once users click on the "Run Simulation" button on the "Control" sheet, the Promodel file is activated and run.

During the simulation, users can observe the dynamic activities of entities in the network and the changes of utilization of subnetworks on the dynamic plot (see Figure 11). The simulation results of human performance and index of mental workload (utilization of subnetwork) is reported after the Promodel file finishes running.

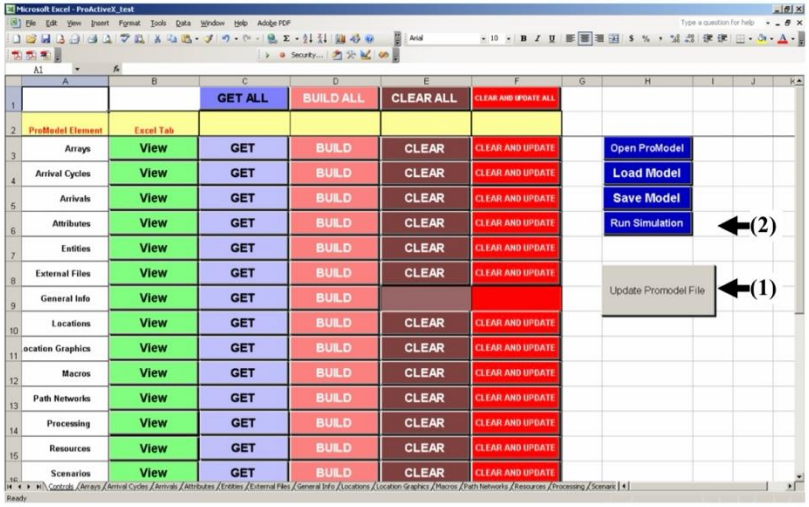


Figure 10. The ActiveX module in the new development of QN-MHP 2.0 (users of the modeling tool only need to click on the "Update Promodel File" (1) and "Run Simulation" button (2) on the "Control" sheet)

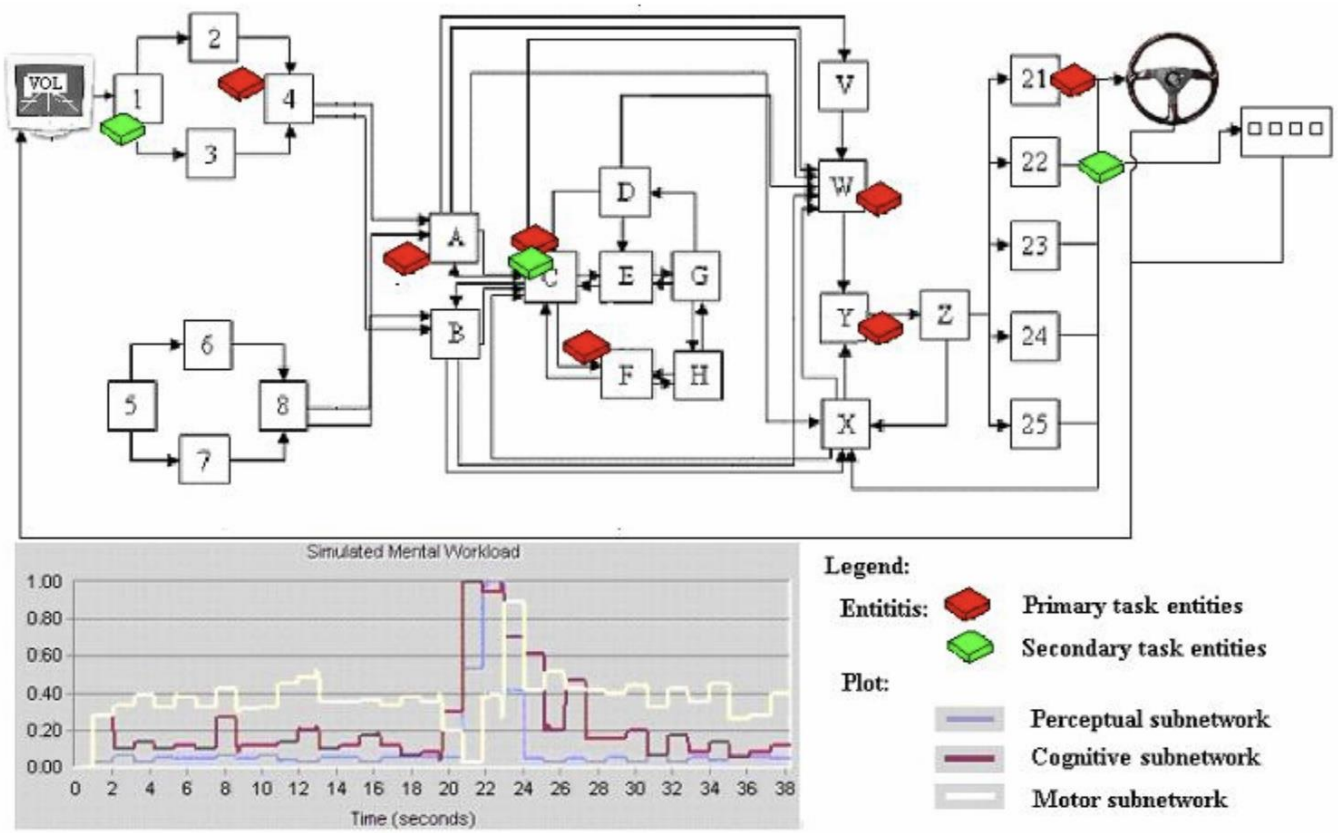


Figure 11. Dynamic change of subnetwork utilization as an index of mental workload during the Promodel simulation (see a short move clip on the website: http:www.acsu.buffalo.edu/changxu/) (Wu & Liu, 2007)

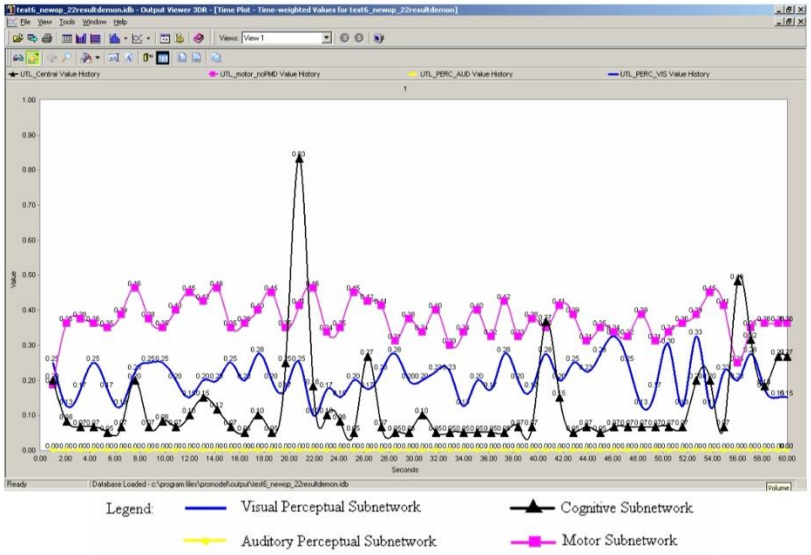


Figure 13. Sample simulation results in Promodel of subnetwork's utilization (X-axis is the simulation time and Y-axis is the utilization of the four subnetworks)

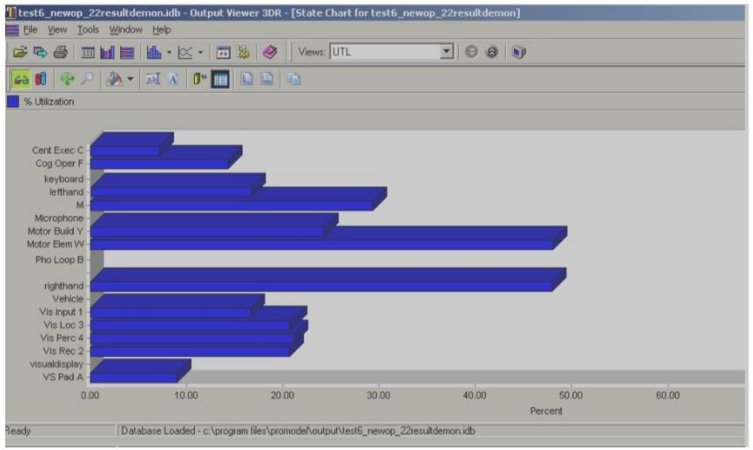


Figure 14. Sample simulation results of the averaged utilizations of servers in the queueing network (X-axis is the averaged utilization of servers and Y-axis is the name of servers described in Figure

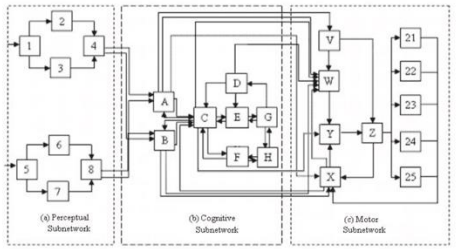


Figure 3. The general structure of the queuing network-model human processor (further developed from Liu, Feyen, & Tsimhoni, 2006; Wu & Liu, 2007a).

model human processor (MHP), which is widely adopted in the field of human–computer interaction (Card, Moran, & Newell, 1983). The queuing network model divides each stage into a subnetwork of a small number of servers and thus has a level of granularity that falls between the neural network and the RSB and MHP models.

In modeling human performance, mathematical models based on queuing networks have successfully integrated a large number of mathematical models in response time (Liu, 1996) and in multitask performance (Liu, 1997) as special cases of queuing networks. A computational cognitive architecture called the queuing network-MHP (QN-MHP; see Figures 3 and 4) has been developed as an integration of queuing networks and MHP for both mathematical modeling and real-time generation of psychological behavior (Liu et al., 2006). QN-MHP has been successfully used to generate human performance and mental workload in real time, including driver performance (Liu et al., 2006) and driver workload (Wu & Liu, 2007a, 2007b; Wu, Tsimhoni, & Liu, in press), transcription typing (Wu & Liu, 2008), and visual–manual tracking performance and mental workload measured by ERP techniques (Wu & Liu, 2008).

QN-MHP consists of three subnetworks: perceptual, cognitive, and motor subnetworks, as described in the following sections.

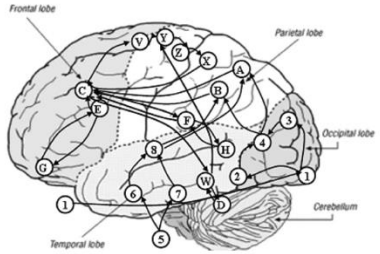


Figure 4. Approximate mapping of servers in the queuing network model onto human brain (Wu & Liu, 2007a). Letters and numbers in illustration correspond to the key in Figure 3.

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 (Bear, et al., 2001; Smith & Jonides, 1998; Ungerleider & Haxby, 1994). 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 GFi—integrate the information of these features from the two visual pathways and generate integrated perception of the objects (Bear et al., 2001). These brain regions also serve as the visual sensory memory storage for the visual information (Bear et al., 2001;

Psychological Review
2008, Vol. 115, No. 4, 913–954

0033-295X/08/$12.00 DOI: 10.1037/a0013123

Queuing Network Modeling of the Psychological Refractory Period (PRP)

Changxu Wu and Yili Liu
University of Michigan

The psychological refractory period (PRP) is a basic but important form of dual-task information processing. Existing serial or parallel processing models of PRP have successfully accounted for a variety of PRP phenomena; however, each also encounters at least 1 experimental counterexample to its predictions or modeling mechanisms. This article describes a queuing network-based mathematical model of PRP that is able to model various experimental findings in PRP with closed-form equations including all of the major counterexamples encountered by the existing models with fewer or equal numbers of free parameters. This modeling work also offers an alternative theoretical account for PRP and demonstrates the importance of the theoretical concepts of "queuing" and "hybrid cognitive networks" in understanding cognitive architecture and multitask performance.

Keywords: psychological refractory period (PRP), cognitive architecture, queuing network, multitask performance, serial and parallel processing

Performing multiple tasks at the same time is common in daily life; for example, drivers can steer a car and at the same time talk with friends in the car, and telephone operators can answer customer phone calls and type textual information into a computer. Among the wide range of multiple task situations, the psychological refractory period (PRP) is one of the most basic and simplest forms of a dual-task situation. In a PRP experiment, two reaction-time (RT) tasks are presented close together in time, and participants are asked to perform the two tasks as quickly as possible. Typically, participants' response to the second of the two RT tasks is performed more slowly than to the first when the interval between the presentation times of these two tasks is short. PRP has been studied in laboratories over 100 years, from the behavioral (Creamer, 1963; Kantowitz, 1974; Oberauer & Kliegl, 2004; Pashler, 1984, 1994b; Schumacher et al., 1999; Solomons & Stein, 1896; Welch, 1898; Welford, 1952) to the neurological level (Jiang, Saxe, & Kanwisher, 2004; Sommer, Leuthold, & Schubert, 2001). It is also the subject of extensive theoretical work and the focal point of an important theoretical controversy between several computational models of cognition. There are several important cognitive models of PRP, including the response-selection bottleneck (RSB) or central bottleneck model proposed by Pashler (1984, 1990, 1994a, 1994b, 1994c), the executive-process interactive control (EPIC) model proposed by Meyer and Kieras (1997a, 1997b), and the model based on the ACT-R/perceptual-motor system (ACT-R/PM) proposed by Byrne and Anderson (2001). Each of these models is able to account for some of the important aspects of PRP; however, each appears to encounter at least one experimental counterexample to its predictions (Jiang et al., 2004; Meyer & Kieras, 1997a, 1997b; Oberauer & Kliegl, 2004; Ruthruff, Pashler, & Klaassen, 2001). Therefore, the questions remain about how to model these experimental results, provide a unified account of the discoveries in behavioral and neuroscience studies, and gain further insights into the mechanisms of dual-task performance.

This article takes further steps toward addressing these important questions with a queuing network-based computational cognitive architecture (Liu, 1996, 1997; Liu, Feyen, & Tsimhoni, 2006; Wu & Liu, 2007a, 2007b, 2008; Wu, Liu, & Walsh, in press; Wu, Tsimhoni, & Liu, in press). First, we introduce the major experimental results in PRP studies and the major PRP effects. Second, the major existing models of PRP are described, including their advantages and their counterexamples. Third, we describe the queuing network model, including its major assumptions and components. In the fourth section, we describe how the queuing network model mathematically accounts for and theoretically explains the PRP phenomena. Finally, we discuss the implications of the model, as well as its extension in future research.

EXPERIMENTAL STUDIES IN PRP

In the following section, we introduce the major findings in experimental studies of PRP, including the basic PRP experiment paradigm and the major PRP effects that have been the subject of theoretical controversy—the subadditive difficulty effect, the response grouping effect, the practice effect, and brain imaging patterns in PRP.

Basic PRP Experiment Paradigm

The basic PRP experiment paradigm requires the participants to perform two tasks (Task 1 and Task 2 (T2) concur-

Changxu Wu and Yili Liu, Department of Industrial and Operations Engineering, University of Michigan.

Changxu Wu is now at the Department of Industrial and Systems Engineering, State University of New York at Buffalo.

We appreciate the support from the National Science Foundation (NSF) for this work (NSF Grant 0308000).

Correspondence concerning this article should be addressed to Changxu Wu, State University of New York (SUNY)—Buffalo, 414 Bell Hall, Buffalo, NY 14260-2050, or to Yili Liu, Department of Industrial and Operations Engineering, University of Michigan, 1205 Beal Avenue, Ann Arbor, MI 48109. E-mail: changxu@buffalo.edu or yililiu@umich.edu

Queueing Network-Model Human Processor (QN-MHP): A Computational Architecture for Multitask Performance in Human-Machine Systems

YILI LIU, ROBERT FEYEN, and OMER TSIMHONI
University of Michigan

Queueing Network-Model Human Processor (QN-MHP) is a computational architecture that integrates two complementary approaches to cognitive modeling: the queueing network approach and the symbolic approach (exemplified by the MHP/GOMS family of models, ACT-R, EPIC, and SOAR). Queueing networks are particularly suited for modeling parallel activities and complex structures. Symbolic models have particular strength in generating a person's actions in specific task situations. By integrating the two approaches, QN-MHP offers an architecture for mathematical modeling and real-time generation of concurrent activities in a truly concurrent manner. QN-MHP expands the three discrete serial stages of MHP, of perceptual, cognitive, and motor processing, into three continuous-transmission subnetworks of servers, each performing distinct psychological functions specified with a GOMS-style language. Multitask performance emerges as the behavior of multiple streams of information flowing through a network, with no need to devise complex, task-specific procedures to either interleave production rules into a serial program (ACT-R), or for an executive process to interactively control task processes (EPIC). Using QN-MHP, a driver performance model was created and interfaced with a driving simulator to perform a vehicle steering, and a map reading task concurrently and in real time. The performance data of the model are similar to human subjects performing the same tasks.

Categories and Subject Descriptors: H.1.2 [**Models and Principles**]: User/Machine Systems—*Human information processing, human factors*; I.6.5 [**Simulation and Modeling**]: Model Development—*Modeling methodologies*

General Terms: Human Factors

Additional Key Words and Phrases: Cognitive model, human-computer interaction, cognition, user interfaces, human information processing

R. Feyen is currently at School of Industrial Engineering, Purdue University.
Authors' address: Y. Liu, O. Tsimhoni, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI 48109; email: {yililiu,omert}@umich.edu; R. Feyen, School of Industrial Engineering, Purdue University, West Lafayette, IN 47907; email: rfeyen@purdue.edu.

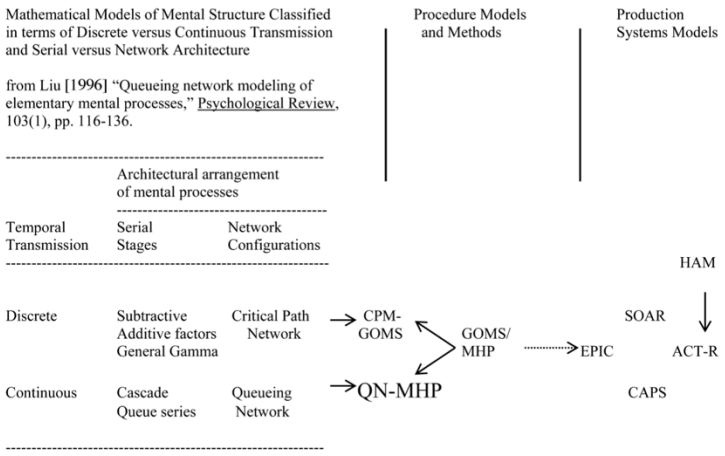


Fig. 1. Mathematical models of mental structure and procedure/production system models of cognitive architecture.

existing approaches. More specifically, we describe our current and proposed work in developing a complementary modeling approach that integrates the modeling philosophy and methods of the procedure-knowledge/production-systems models listed above and the mathematical/simulation theories and methods of queueing networks. Research on queueing networks is not only a major branch of mathematics and operations research but also one of the most commonly used methods for performance analysis of a large variety of real-world systems such as computer, communications, manufacturing, and transportation networks (e.g., Disney and Konig [1985]; Denning and Buzen [1978]; Boxma and Daduna [1990]). A large knowledge base on queueing networks exists, and some well-developed simulation and analysis software programs are widely used by engineers world-wide. Furthermore, from the psychological modeling perspective, as published in a *Psychological Review* article entitled "Queueing network modeling of elementary mental processes" [Liu 1996], we have successfully used queueing networks to integrate a large number of influential mathematical models of mental structure and psychological processes, such as Sternberg's serial stages model [Sternberg 1969], McClelland's cascade model [McClelland 1979], and Schweickert's critical path network model [Schweickert 1978] (see the left-half of Figure 1). From the systems engineering perspective, we have successfully used queueing networks to integrate single-channel one-server queueing models (e.g., Senders [1964]; Rouse [1980]) and the parallel processing models (e.g., Laughery [1989]; Wickens and Liu [1988]) as special cases [Liu 1994, 1997].

Mathematical Models of Mental Structure Classified in terms of Discrete versus Continuous Transmission and Serial versus Network Architecture

from Liu [1996] "Queueing network modeling of elementary mental processes," Psychological Review, 103(1), pp. 116-136.

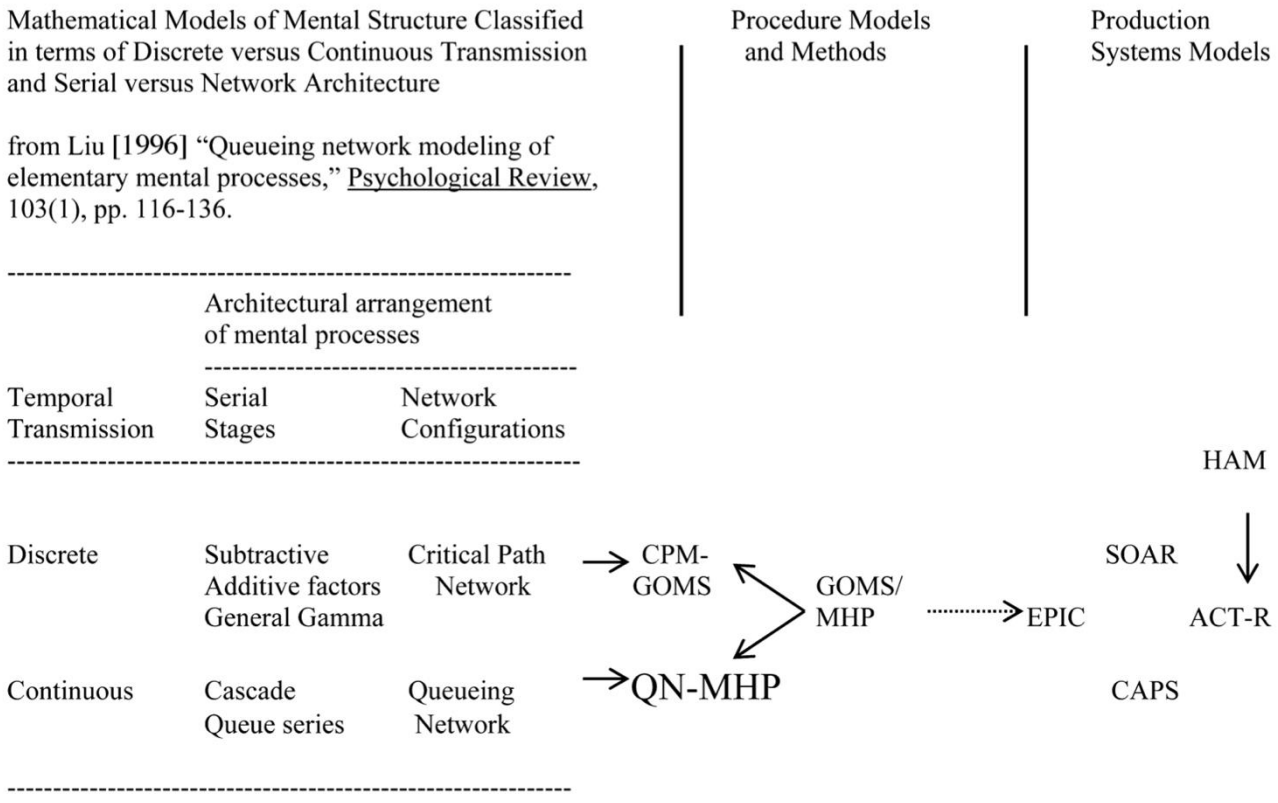


Fig. 1. Mathematical models of mental structure and procedure/production system models of cognitive architecture.

Mathematical Models of **RT** and Mental **Structure** Classified in terms of Discrete versus Continuous Information Transmission and Serial versus Network Architecture

(from Liu, 1996, "Queueing network modeling of elementary mental processes," Psychological Review, 103(1), pp. 116-136).

Temporal Transmission	Architectural arrangement of mental processes: Serial Stages	Architectural arrangement of mental processes: **Network Configurations**
Discrete	Subtractive Additive factors General Gamma	Critical Path Network (PERT)
Continuous	Cascade Queueing series	**Queueing Network (QN)**

- **Non-unidirectional flow**
- **Non-selective influence**
- **By-passing**
- **Fixed capacity**

Mathematical Models of Mental Structure Classified in terms of Discrete versus Continuous Transmission and Serial versus Network Architecture

(from Liu, 1996, "Queueing network modeling of elementary mental processes," Psychological Review, 103(1), pp. 116-136.

Temporal Transmission	Architectural arrangement of mental processes: Serial Stages	Network Configurations
Discrete	Subtractive Additive factors General Gamma	Critical Path Network
Continuous	Cascade Queue series	Queueing Network

Procedure Models and Methods

Production Systems

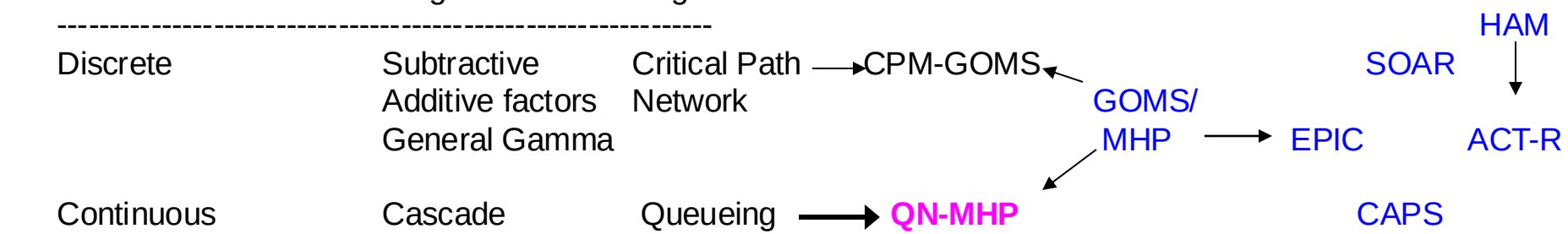


Figure 1: **Mathematical** (left side) and **Symbolic** (right side) models of mental architecture

Liu, Feyen, & Tsimhoni (2006) "QN-MHP: A computational architecture for multitask performance," ACM Transactions on Computer-Human Interaction, vol 13, pp. 37-70.

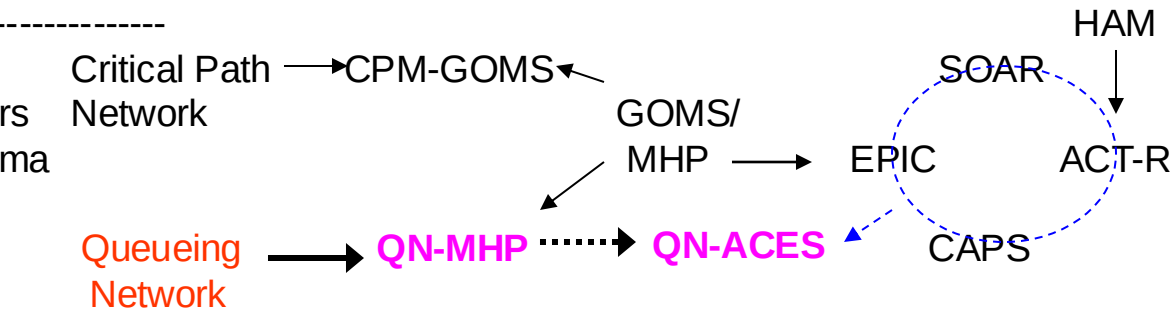
Mathematical Models of Mental Structure Classified in terms of Discrete versus Continuous Transmission and Serial versus Network Architecture

(from Liu, 1996, "Queueing network modeling of elementary mental processes," Psychological Review, 103(1), pp. 116-136.

Procedure Models and Methods

Production Systems

Temporal Transmission	Architectural arrangement of mental processes: Serial Stages	Network Configurations
Discrete	Subtractive Additive factors General Gamma	Critical Path Network
Continuous	Cascade Queue series	Queueing Network



INTL. JOURNAL OF HUMAN–COMPUTER INTERACTION, *25*(6), 554–581, 2009

ISSN: 1044-7318 print / 1532-7590 online
DOI: 10.1080/10447310902973182

Taylor & Francis
Taylor & Francis Group

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

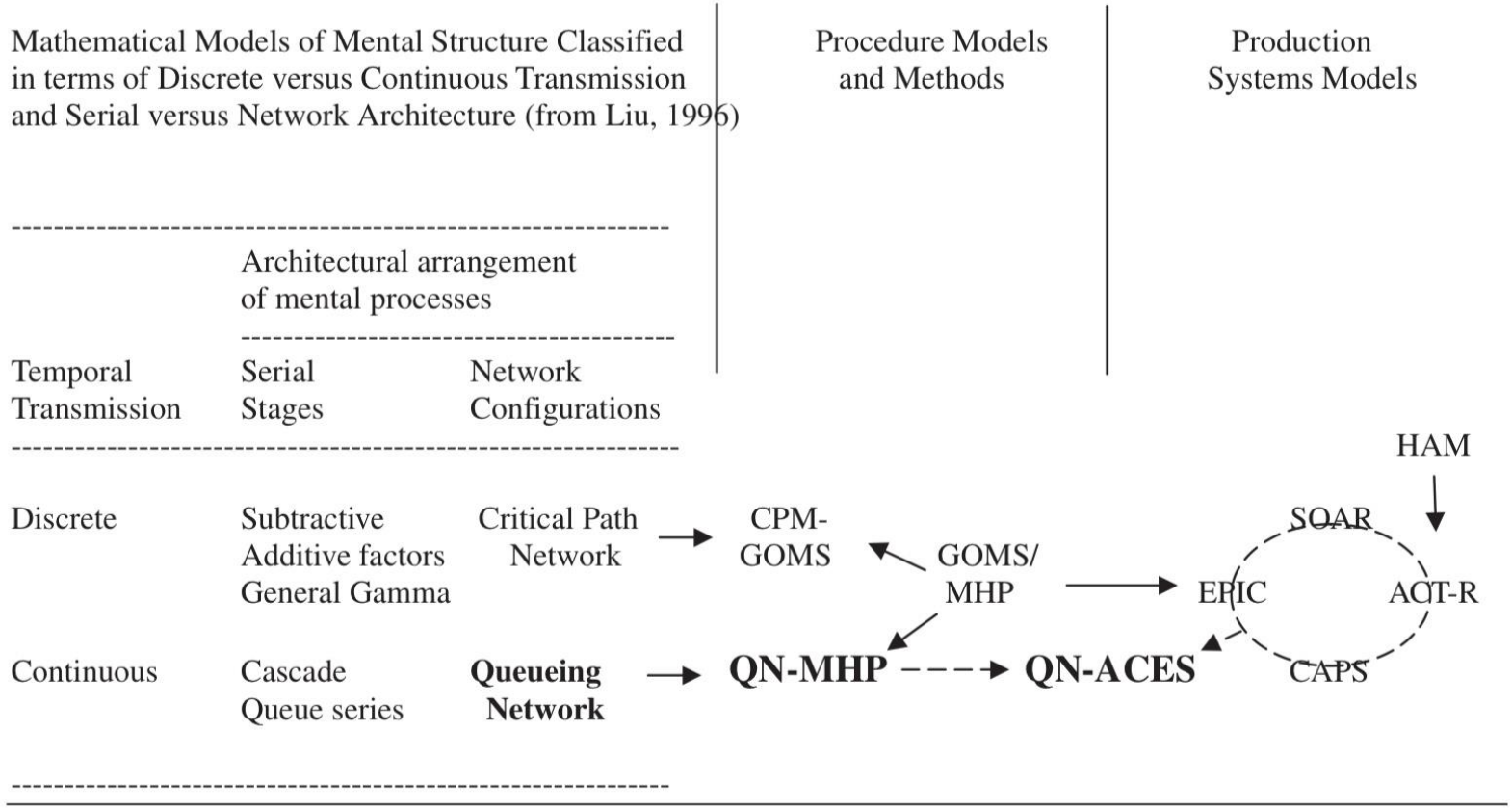


FIGURE 2 Mathematical and symbolic models of mental architecture (Liu, 2006; Liu et al, 2006) showing the relationship between QN, QN-MHP, QN-ACES and a sample of influential cognitive architectures. *Note*: By integrating the complementary schools of mathematical (left half of the figure) and symbolic (right half) models, QN-MHP supports both precise mathematical analysis and real-time generation of behavior, thus capitalizing on the strength and overcoming the weaknesses of either mathematical or symbolic modeling.

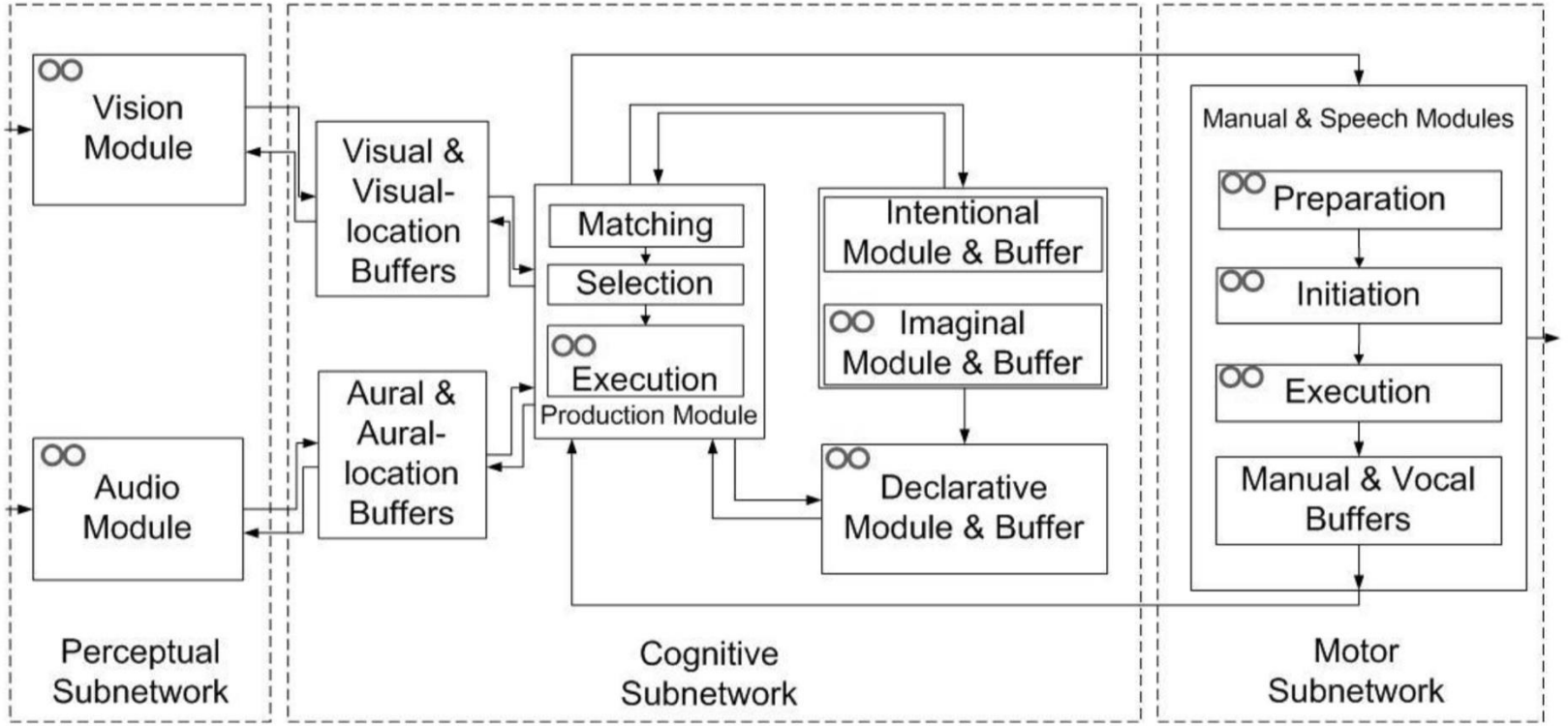


Figure 3. Server structure of QN-ACTR. Queue symbols (shown as two circles) mark the servers where queues are added from the QN's perspective. All the server processing logics in the QN-ACTR are identical to the corresponding algorithms in ACT-R (adapted from Cao & Liu, 2012c).

the queues of the cognitive subnetwork may provide sources of activation for the working memory. In the perceptual subnetwork, there may be a connection between the queues and short term sensory storages, including the visual sensory memory (Dick, 1974) and the auditory sensory memory (Darwin, Turvey, & Crowder, 1972), because they both have the functionality of temporarily storing information. In the motor subnetwork, queues are believed to temporarily store time-ordered motor commands (P. E. Roland et al., 1980). The correspondence between queues and cognitive or neurological constructs, however, is not the focus of the current study. Instead, I focus on the verification of QN-ACTR in terms of human performance modeling capabilities and demonstrate the importance of the theoretical concepts of "queueing" in understanding cognitive architecture and multi-task performance.

Table 1. Correspondence between ACT-R modules and QN servers (from Cao & Liu, 2012c).

ACT-R modules and buffers	**Corresponding QN servers. See (Wu & Liu, 2008a) for server details.**
Vision module	Server 1 – 4
Audio module	Server 5 – 8
Visual and visual-location buffers	Server A
Aural and aural-location buffers	Server B
Production module	Server C, D, F
Intentional and imaginal modules and buffers	Server E, G
Declarative module and buffer	Server H
Manual and speech modules and buffers	Server V, W, X, Y, Z, 21 – 25

At the implementation level, QN-ACTR is a computerized program built on a discrete event simulation platform Micro Saint® Sharp (http://www.maad.com). A full integration was implemented, which means that all the data structures and the functions in ACT-R have been ported from the original Lisp implementation to Micro Saint® Sharp. This programming platform is selected because of three major reasons. First, this platform provides graphical interfaces for easy QN construction and simulation and also supports the visualization of server network and task interaction. Second, this C# based platform supports C# programming plug-in functions and the connection with other C# applications, which help the development of QN-ACTR as an easy-to-use cognitive engineering tool with features such as model comparison and human experiment

their structures. The processing logics in each QN-ACTR server are identical to the algorithms in the corresponding ACT-R module, including the sub-symbolic computations in the production and the declarative modules. As previously described in Chapter 1, ACT-R modules do not have queueing mechanisms, but queues can be added from the QN perspective to support the scheduling of multi-tasks at the local server level. In QN-ACTR, queues are added to the modules that have non-zero processing time and a limited capacity. Currently, I do not apply any constraint to the capacity of queues and assume no time is needed for an information entity to enter or leave a queue. These assumptions are the same as the ones in previous QN modeling work (Liu et al., 2006; Wu & Liu, 2008a).

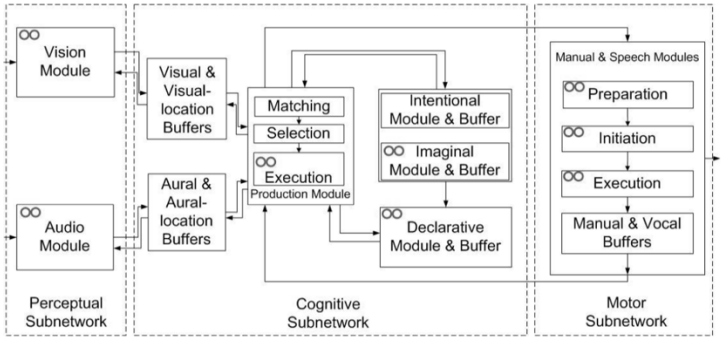


Figure 3. Server structure of QN-ACTR. Queue symbols (shown as two circles) mark the servers where queues are added from the QN's perspective. All the server processing logics in the QN-ACTR are identical to the corresponding algorithms in ACT-R (adapted from Cao & Liu, 2012c).

As shown in Figure 3, the servers are mainly based on ACT-R's modules and buffers. In ACT-R, there is no special module of working memory. Instead, "working memory can be equated with the portion of declarative memory above a threshold of activation" (p. 221, J. R. Anderson, Reder, & Lebiere, 1996), and the chunks that are temporarily held in the cognitive subnetwork (e.g., in the goal buffer) may provide a context and spread activation to the chunks in declarative memory, facilitating the retrieval of context-relevant declarative chunks. In this regard, entities (e.g., chunks) in

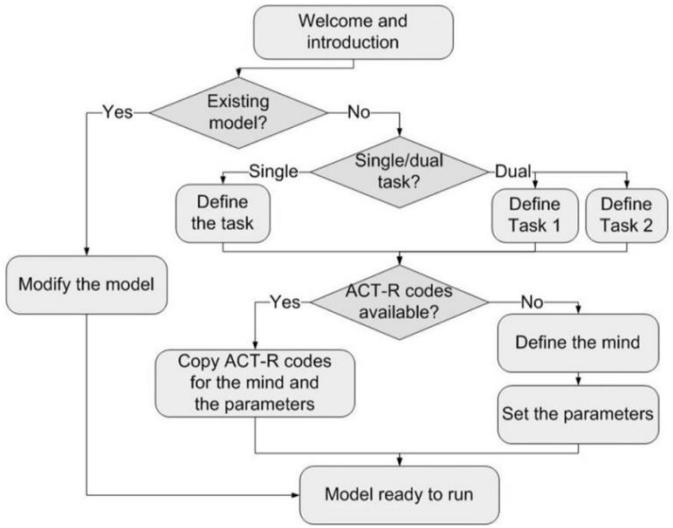


Figure 4. Flow chart showing the model development process in QN-ACTR (from Cao & Liu, 2012b).

QN-ACTR's modeling results include the trace of simulated mental activities, behavioral results, reaction times, correct rates, and mental workload. These results can be visualized while the model is performing the task and recorded for future analyses. QN-ACTR also has an integrated human experiment platform for human data collection. For example, a modeler can use QN-ACTR to conduct simulated driving experiments with steering wheels and pedals. This feature allows models and humans to perform and be compared in the same tasks with identical interfaces, with no need to replicate the real world experiment system in the modeling platform for models to interact with. Using the same experiment platform saves programming work and, more importantly, avoids any discrepancy between human and model tests caused by different experiment setups.

Table 2. Descriptions of the 20 tasks modeled to verify QN-ACTR (adapted from Cao & Liu, 2011a).

Model	Description
	ACT-R 6.0 (v1.3) tutorial models
Addition	Compute 5 + 2 by counting 5, 6, and 7.
Count	Count from 2 to 4.
Semantic	Judge if an animal belongs to a category.
Tutor-model	Compute 36 + 47 by first adding the single digits and then the ten digits.
Demo	See a letter. Press the key for the letter.
Unit-2-assignment	See three letters. Two are the same. Press the key of the single letter.
Sperling	Briefly see 12 letters in three rows. Press keys for letters in the target row, which is indicated by the pitch of a tone. Higher pitch = higher row.
Subitize	See several "x"s. Say how many "x"s there are.
Paired	Memorize and recall 20 word-number pairs, one pair a trial.
Zbrodoff	Judge alphabetic arithmetic problems by pressing keys. For example, A + 2 = C is correct, but B + 3 = F is not.
Fan	Remember a set of people-location facts. Answer queries such as "is the captain in the park?"
Group	Imperfect recall of 9 numbers in three groups, 123-456-789.
Siegle	Imperfect single-digit addition due to number similarity.
Bst-learn	Create a stick of a particular target length by selecting building sticks with three different lengths using the mouse.
Choice	Repetitively guess a biased coin.
Paired-learning	Same as the paired model except starting the task-specific knowledge from descriptive instructions instead of procedural rules.
Past-tense	Learn English past tenses from examples, demonstrating the overregularization of irregular verbs.
	Schumacher et al. (2001) experiments modeled by threaded cognition
Exp. 1	Psychological refractory period, visual-motor and auditory-vocal dual-task. No specific response order.
Exp. 2	Same as Exp. 1 except prioritizing on the auditory-vocal task.
Exp. 3	Same as Exp. 1 except using incompatible stimulus-response associations instead of compatible ones.

2.2 Results

Both the mental activity results recorded in the model output traces and the behavioral results such as reaction times and correction rates were compared between QN-ACTR and ACT-R models. Output traces were compared line by line to examine both the time and event contents. For example, the following line of trace,

222.423 DECLARATIVE RETRIEVED-CHUNK pair18-0

from the Paired model results in QN-ACTR showed that at clock time 222.423 second, the model's declarative module retrieved the chunk *pair18-0*. This trace was identical to the corresponding ACT-R trace. For the tasks with quantitative behavioral

results, mean absolute percentage error (*MAPE*), root mean square error (*RMSE*), and coefficient of determination (R^2) were computed between QN-ACTR and ACT-R results.

Table 3. Model code excerpts from a QN-ACTR model for the Demo task in ACT-R's tutorial. The syntaxes specifying task-specific knowledge and parameters are identical to the ones used in ACT-R (from Cao & Liu, 2013c).

Model setup step	Model codes
Task description	(use_task_dbt_template :method discrete_display_feedback_two_stages_method) (add_trials_from_discrete_display_feedback_two_stages_method :add_number_of_trials_per_block 1 :number_of_responses_per_trial 1 (:item_type display_item_visual_text :visual_text ("B" "C" "D" "F" "G" "H" "J" "K" "L" "M" "N" "P" "Q" "R" "S" "T" "V" "W" "X" "Y" "Z") :correct_response_to_each_visual_text (b c d f g h j k l m n p q r s t v w x y z) :text_randomization without_replacement :display_item_screen_location_x (125) :display_item_screen_location_y (150)))
Declarative knowledge description	(chunk-type read-letters state) (chunk-type array letter) (add-dm (start isa chunk) (attend isa chunk) (respond isa chunk) (done isa chunk) (goal isa read-letters state start))
Procedural knowledge description	(P find-unattended-letter =goal> ISA read-letters state start ==> +visual-location> ISA visual-location :attended nil =goal> state find-location)
Parameters	(sgp :v t :needs-mouse nil :show-focus t :trace-detail high)

As summarized in Table 4, the verification results showed that QN-ACTR models generated the same results as the original ACT-R models. Results from 15 of the 20 tasks were identical between QN-ACTR and ACT-R. For the other five models, results were very similar (*MAPE* < 5.0% and R^2 > 0.9). The sources of the remaining variances include the difference of built-in random number functions between Lisp and C#, which were used in randomly focusing visual attention on a visual item, and the difference in rounding digits between Lisp and C#. These results support the conclusion that the ACT-R functions built in QN-ACTR are accurate and complete.

Table 4. QN-ACTR verification results (adapted from Cao & Liu, 2011a).

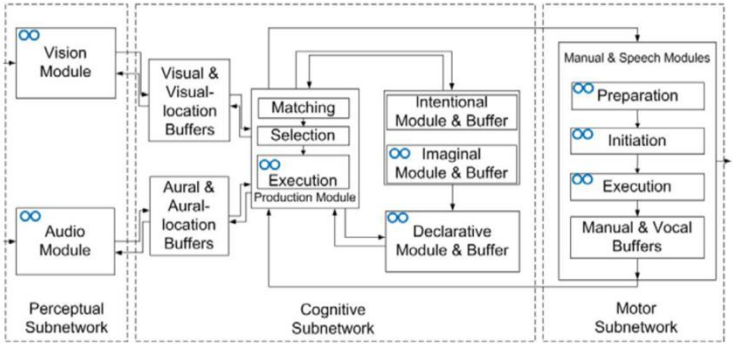
Model	Results
ACT-R 6.0 (v1.3) tutorial models	
Addition	Same output traces ended at 0.5 s.
Count	Same output traces ended at 0.3 s.
Semantic	Same output traces for test 1 (ended at 0.15 s), test 2 (0.25 s), and test 3 (0.35 s).
Tutor-model	Same output traces ended at 0.45 s.
Demo2	Same output traces for both home-location key condition (ended at 0.785 s) and other key condition (1.035 s).
Unit-2-assignment	Same output traces for both home-location key condition (ended at 1.055 s) and other key condition (1.305 s).
Sperling	1000 run results (numbers of correct responses): $MAPE = 1.4\%$, $RMSE = 0.032$, $R^2 = 0.997$.
Subitize	Same output traces. 1 run results (both reaction time and correction rate): $MAPE = 0\%$, $RMSE = 0$, $R^2 = 1$.
Paired	
Zbrodoff	
Fan	
Group	Same output traces ended at 18.535 s.
Siegle	Same output traces. 500 run results (response distribution): $MAPE = 0\%$, $RMSE = 0$, $R^2 = 1$.
Bst-learn	Same output traces. 5 run results (overshoot chances and rule utilities): $MAPE = 0\%$, $RMSE = 0$, $R^2 = 1$.
Choice	Same output traces. 1 run results (rate of guessing heads): $MAPE = 0\%$, $RMSE = 0$, $R^2 = 1$.
Paired-learning	Same output traces. 1 run results (both reaction time and correction rate): $MAPE = 0\%$, $RMSE = 0$, $R^2 = 1$.
Past-tense	Same production rules composed and same output traces over 10 trials, except for a 0.001% time difference.
Schumacher et al. (2001) experiments modeled by threaded cognition	
Exp. 1	1 run results (reaction time): $MAPE = 1.3\%$, $RMSE = 7.554$, $R^2 = 0.998$.
Exp. 2	30 run results (reaction time): $MAPE = 2.3\%$, $RMSE = 10.851$, $R^2 = 0.983$.
Exp. 3	3 run results (reaction time): $MAPE = 4.0\%$, $RMSE = 37.742$, $R^2 = 0.939$.

3. QN-ACTR Simulation of Transcription Typing and Reading Comprehension Tasks

The verification of QN-ACTR described in the previous section established the basis for further incorporating unique QN mechanisms into QN-ACTR. In this section, I demonstrate the improved modeling capability of QN-ACTR after adding QN mechanisms. A model was built using QN-ACTR to simulate transcription typing tasks involving dual-task performance and reading comprehension, illustrating the benefits of the integrated cognitive architecture in modeling complex cognition and multi-task scenarios and resolving the concurrent goal scheduling and the module jamming issues in ACT-R. The transcription typing tasks were selected, because (1) previous studies about transcription typing have accumulated numerous detailed empirical results that are very useful for comparing models, (2) the previous QN architecture (i.e., QN-MHP) has

that supports the integration of their structures. The processing logics in each QN-ACTR server are identical to the algorithms in the corresponding ACT-R module, including the sub-symbolic computations in the production and the declarative modules. As previously described, ACT-R modules do not have queueing mechanisms, but queues can be added from the QN perspective to support the scheduling of multi-tasks at the local server level. In QN-ACTR, queues are added to the modules that have non-zero processing time and a limited capacity. Currently, we do not apply any constraint to the capacity of queues and assume no time is needed for an information entity to enter or leave a queue. These assumptions are the same as the ones in previous QN modelling work (Liu et al., 2006; Wu and Liu, 2008a).

Figure 1 Server structure of QN-ACTR (see online version for colours)



Note: Queue symbols (shown as two circles) mark the servers where queues are added from the QN's perspective. All the server processing logics in the QN-ACTR are identical to the corresponding algorithms in ACT-R

Source: adapted from Cao and Liu (2012b)

Table 1 Correspondence between ACT-R modules and QN servers

ACT-R modules and buffers	*Corresponding QN servers. See (Wu and Liu, 2008a) for server details.*
Vision module	Server 1–4
Audio module	Server 5–8
Visual and visual-location buffers	Server A
Aural and aural-location buffers	Server B
Production module	Server C, D, F
Intentional and imaginal modules and buffers	Server E, G
Declarative module and buffer	Server H
Manual and speech modules and buffers	Server V, W, X, Y, Z, 21–25

Source: from Cao and Liu (2012b)

As shown in Figure 1, the servers are mainly based on ACT-R's modules and buffers. In ACT-R, there is no special module of working memory. Instead, "working memory can be equated with the portion of declarative memory above a threshold of activation"

together, these 20 tasks provide a thorough test bed to verify the implementation and programming of QN-ACTR.

Table 2 Descriptions of the 20 tasks modelled to verify QN-ACTR

Model	*Description*
	ACT-R 6.0 (v1.3) tutorial models
Addition	Compute 5 + 2 by counting 5, 6, and 7.
Count	Count from 2 to 4.
Semantic	Judge if an animal belongs to a category.
Tutor-model	Compute 36 + 47 by first adding the single digits and then the ten digits.
Demo	See a letter. Press the key for the letter.
Unit-2-assignment	See three letters. Two are the same. Press the key of the single letter.
Sperling	Briefly see 12 letters in three rows. Press keys for letters in the target row, which is indicated by the pitch of a tone. Higher pitch = higher row.
Subitize	See several 'x's. Say how many 'x's there are.
Paired	Memorise and recall 20 word-number pairs, one pair a trial.
Zbrodoff	Judge alphabetic arithmetic problems by pressing keys. For example, A + 2 = C is correct, but B + 3 = F is not.
Fan	Remember a set of people-location facts. Answer queries such as 'is the captain in the park?'
Group	Imperfect recall of 9 numbers in three groups, 123-456-789.
Siegle	Imperfect single-digit addition due to number similarity.
Bst-learn	Create a stick of a particular target length by selecting building sticks with three different lengths using the mouse.
Choice	Repetitively guess a biased coin.
Paired-learning	Same as the paired model except starting the task-specific knowledge from descriptive instructions instead of procedural rules.
Past-tense	Learn English past tenses from examples, demonstrating the overregularisation of irregular verbs.
	Schumacher et al. (2001) experiments modelled by threaded cognition
Exp. 1	Psychological refractory period, visual-motor and auditory-vocal dual-task. No specific response order.
Exp. 2	Same as Exp. 1 except prioritising on the auditory-vocal task.
Exp. 3	Same as Exp. 1 except using incompatible stimulus-response associations instead of compatible ones.

Source: adapted from Cao and Liu (2011a)

Models for the 20 tasks were developed strictly following their corresponding ACT-R models. The task displays and controls were modelled using QN-ACTR's syntaxes and templates. The codes specifying task-specific knowledge and parameters were directly copied from the codes used in the original ACT-R models. In addition, we also used the Common Random Numbers technique (McGeoch, 1992) to assign ACT-R and QN-ACTR models the same randomisation method and the same seeds in order to further control the sources of variance in the verification. Table 3 shows excerpts from the

QN-ACTR model codes for the Demo task. This task presents a random letter on the screen and requires a key press response corresponding to the displayed letter.

Table 3 Model code excerpts from a QN-ACTR model for the demo task in ACT-R's tutorial

Model setup step	*Model codes*
Task description	(use_task_dbt_template :method discrete_display_feedback_two_stages_method) (add_trials_from_discrete_display_feedback_two_stages_method :add_number_of_trials_per_block 1 :number_of_responses_per_trial 1 (:item_type display_item_visual_text :visual_text ("B" "C" "D" "F" "G" "H" "J" "K" "L" "M" "N" "P" "Q" "R" "S" "T" "V" "W" "X" "Y" "Z") :correct_response_to_each_visual_text (b c d f g h j k l m n p q r s t v w x y z) :text_randomisation without_replacement :display_item_screen_location_x (125) :display_item_screen_location_y (150)))
Declarative knowledge description	(chunk-type read-letters state) (chunk-type array letter) (add-dm (start isa chunk) (attend isa chunk) (respond isa chunk) (done isa chunk) (goal isa read-letters state start))
Procedural knowledge description	(P find-unattended-letter =goal> ISA read-letters state start ==> +visual-location> ISA visual-location :attended nil =goal> state find-location)
Parameters	(sgp :v t :needs-mouse nil :show-focus t :trace-detail high)

Note: The syntaxes specifying task-specific knowledge and parameters are identical to the ones used in ACT-R

3.2 Results

Both the mental activity results recorded in the model output traces and the behavioural results such as reaction times and correction rates were compared between QN-ACTR and ACT-R models. Output traces were compared line by line to examine both the time and event contents. For example, the following line of trace,

222.423 DECLARATIVE RETRIEVED-CHUNK pair18-0

from the paired model results in QN-ACTR showed that at clock time 222.423 second, the model's declarative module retrieved the chunk *pair18-0*. This trace was identical to the corresponding ACT-R trace. For the tasks with quantitative behavioural results, mean absolute percentage error (*MAPE*), root mean square error (*RMSE*), and coefficient of determination (R^2) were computed between QN-ACTR and ACT-R results.

Table 4 QN-ACTR verification results

Model	*Results*
	ACT-R 6.0 (v1.3) tutorial models
Addition	Same output traces ended at 0.5 s.
Count	Same output traces ended at 0.3 s.
Semantic	Same output traces for test 1 (ended at 0.15 s), test 2 (0.25 s), and test 3 (0.35 s).
Tutor-model	Same output traces ended at 0.45 s.
Demo2	Same output traces for both home-location key condition (ended at 0.785 s) and other key condition (1.035 s).
Unit-2-assignment	Same output traces for both home-location key condition (ended at 1.055 s) and other key condition (1.305 s).
Sperling	1,000 run results (numbers of correct responses): *MAPE* = 1.4%, *RMSE* = 0.032, R^2 = 0.997.
Subitize Paired Zbrodoff Fan	Same output traces. 1 run results (both reaction time and correction rate): *MAPE* = 0%, *RMSE* = 0, R^2 = 1.
Group	Same output traces ended at 18.535 s.
Siegle	Same output traces. 500 run results (response distribution): *MAPE* = 0%, *RMSE* = 0, R^2 = 1.
Bst-learn	Same output traces. 5 run results (overshoot chances and rule utilities): *MAPE* = 0%, *RMSE* = 0, R^2 = 1.
Choice	Same output traces. 1 run results (rate of guessing heads): *MAPE* = 0%, *RMSE* = 0, R^2 = 1.
Paired-learning	Same output traces. 1 run results (both reaction time and correction rate): *MAPE* = 0%, *RMSE* = 0, R^2 = 1.
Past-tense	Same production rules composed and same output traces over ten trials, except for a 0.001% time difference.
	Schumacher et al. (2001) experiments modelled by threaded cognition
Exp. 1	1 run results (reaction time): *MAPE* = 1.3%, *RMSE* = 7.554, R^2 = 0.998.
Exp. 2	30 run results (reaction time): *MAPE* = 2.3%, *RMSE* = 10.851, R^2 = 0.983.
Exp. 3	3 run results (reaction time): *MAPE* = 4.0%, *RMSE* = 37.742, R^2 = 0.939.

Source: adapted from Cao and Liu (2011a)

As summarised in Table 4, the verification results showed that QN-ACTR models generated the same results as the original ACT-R models. Results from 15 of the 20 tasks were identical between QN-ACTR and ACT-R. For the other five models, results were

module, the model can capture comprehension errors caused by forgetting. For the concurrent task phenomena, the secondary auditory-pedal task was modelled following the Sperling task in ACT-R's tutorial. When a tone is presented, the model first detects the sound and then responds by issuing a pedal-pressing action. As we previously described, the goal chunk of this secondary task co-exists with the goal chunk of the primary typing task in the goal buffer. The queueing mechanism in the production module coordinates multiple tasks without the need to define any multitask-specific knowledge or any executive process. Since the primary typing task was stressed by instructions in the original experiment, the typing task was assigned a higher priority than the auditory-pedal task in the queueing mechanism. All related parameters used values from previous studies (Table 6), with all other parameters at their default values. As previously described in equation (1), the parameters A_i, B_i, α_i, and N_i affect how the motor servers' processing time decreases with practice. The other parameters are integrated from the ACT-R architecture, and Table 6 provides a brief description (see ACT-R reference manual for details, ACT-R Group, 2011). The Nelson-Denny Reading Test was used in the simulation as in the empirical experiments.

Table 5 Summary of transcription typing phenomena modelled in QN-ACTR

Phenomena description	*Empirical human results*	*QN-ACTR simulation results*
	Phenomena involving complex cognition	
Typing is slower than reading.	Reading speed = 253 words-per-minute (wpm); typing speed = 58 wpm; comprehension accuracy is 58.1% for reading only and 44.7% for reading while typing. Results from 74 typists typing or reading passages with about 1,200 characters. (Salthouse, 1984)	Reading speed = 267 wpm (absolute percentage error, *APE* = 5.4%); typing speed = 69 wpm (*APE* = 19.0%); comprehension accuracy is 55.3% (*APE* = 4.9%) for reading only and 44.9% (*APE* = 0.4%) for typing and reading.
Typing skill and comprehension are independent.	No significant correlation (i.e., $r = -0.169$, $p > 0.15$). 74 typists. Passage with about 1200 characters. (Salthouse, 1986)	The correlation $r = 0.042$, $p = 0.80 > 0.15$.
	Dual-task phenomena	
A concurrent task does not affect typing performance.	The interkey time in the concurrent task situation (185 ms) was not significantly longer than that in single-task typing (181 ms). 40 typists. Passage with about 1,250 characters. (Salthouse and Saults, 1987)	Interkey time in the concurrent task situation was 177 ms (*APE* = 4.3%), similar to typing only (172 ms).
	Other phenomena	
Typing is faster than choice reaction time.	Typing interkey time (median) = 177 ms for skilled typists; choice reaction time = 560 ms. 74 typists. Passage with about 1200 characters. (Salthouse, 1984)	Typing interkey time (median) = 182 ms (*APE* = 2.8%); choice reaction time = 495 ms (*APE* = 11.7%).

Table 5 Summary of transcription typing phenomena modelled in QN-ACTR (continued)

Phenomena description	*Empirical human results*	*QN-ACTR simulation results*
	Other phenomena	
Typing rate is independent of word order.	Qualitative phenomena (Wu and Liu, 2008b)	Typing interkey time = 172 ms for normal order and 172 ms for randomised word order, t (6601) = 0.001, p = 0.999.
Typing speed is slower with random character order.	Interkey time in typing increased to 454 ms when typing materials composed of words with random characters. Five subjects (three typists). 220 words. (Hershman and Hillix, 1965)	Typing interkey time = 172 ms for normal order and 373 ms for random character order (*APE* = 17.8%).
Typing rate is impaired by restricted preview.	Typing rate decreases with smaller preview window of the material to be typed. Eight typists. Six passages each with about 74 words. (Inhoff and Wang, 1992)	R^2 of simulated interkey time is 0.98 (*APE* = 9.2%).
Alternate-hand advantage	Alternate-hand keystrokes are about 45 ms faster than the same-hand keystrokes (Wu and Liu, 2008b)	78 ms faster (*APE* = 73.6%).
Digram frequency effect	Digram (letter pairs) that occur more frequently in normal language are typed faster than less frequent digram. 45 typists. Passage with about 1,250 characters. (Salthouse, 1984)	Significantly faster, t (197) = –11.062, $p < 0.001$.
Interkey time is independent of word length.	No significant difference between long and short words. 74 typists. Passage with about 1,200 characters. (Salthouse, 1984)	No significance, t (387) = 0.381, $p = 0.70$.
Word initiation effect	The first keystroke in a word is about 45 ms slower than the subsequent keystrokes. 74 typists. Passage with about 1,200 characters. (Salthouse, 1984)	53 ms slower (*APE* = 17.1%).
Context phenomenon	The time for a keystroke is dependent on the specific context in which the character appears, especially keyboard topography (Wu and Liu, 2008b)	Interkey time for the same key can range from around 60 ms to 200 ms, depending on the context of the previous key.
Copying span	14.6 characters. 29 typists. Eight sentences, each about 75 characters. (Salthouse, 1985)	10.9 characters (*APE* = 25.5%).
Stopping span	2.16 characters. 12 secretaries. 300 sentences, each about 28 characters. (Logan, 1982)	1.8 characters (*APE* = 16.7%).

Table 5 Summary of transcription typing phenomena modelled in QN-ACTR (continued)

Phenomena description	*Empirical human results*	*QN-ACTR simulation results*
	Other phenomena	
Eye-hand span	5.25 characters, averaged from multiple studies (for details, see Salthouse, 1986).	6.1 characters (*APE* = 16.0%).
Eye-hand span is smaller for randomly ordered letters.	1.75 characters. 74 typists. Passage with about 1,200 characters. (Salthouse, 1984)	1.0 characters (*APE* = 42.9%).
Replacement span	2.8 characters. 85 typists. Passages with about 1,200 characters. (Salthouse and Saults, 1987)	4.5 characters (*APE* = 60.7%).
Detection span	8.1 characters. 85 typists. Passages with about 1,200 characters. (Salthouse and Saults, 1987)	9.1 characters (*APE* = 12.3%).
Two-hand digrams or two-finger digrams exhibit greater changes with skills than do one-finger digrams.	The slope of the regression equations relating the digram interval (ms) to typing speed of two-hand digrams (–2.08) and two-finger digrams (–2.38) were greater than that of one-finger digrams (–1.38). 74 typists. Passage with about 1,200 characters. (Salthouse, 1984)	Two-hand (–6.05) and two-finger (–4.06) are greater than one finger (–2.92 on average).
Repetitive tapping rate increases with skill.	Significant positive correlation between the tapping rate and the net typing speed ($p < 0.01$). 74 typists. Passage with about 1,200 characters. (Salthouse, 1984)	$r = 0.81$ ($p < 0.01$).
The variability of interkey time decreases with the skills.	Inter-keystroke variability correlated –0.69 with the net typing speed; intra-keystroke variability correlated –0.71 with the net typing speed. 74 typists. Passage with about 1,200 characters. (Salthouse, 1984)	Inter-key $r = -0.85$ ($p < 0.05$); intra-key $r = –0.90$ ($p < 0.05$)
Eye-hand span is larger with increased skills.	The correlation between the eye-hand span and net words-per-minute was significant with $p < 0.01$. 74 typists. Passage with about 1,200 characters. (Salthouse, 1984)	$r = 0.99$ ($p < 0.05$).
Replacement span is larger with more skills.	The correlation between net words per minute and the replacement span was 0.80 ($p < 0.01$). 29 typists. Eight sentences, each about 75 characters. (Salthouse, 1985)	$r = 0.61$ ($p < 0.05$).
Interkey time decreases with practice.	Qualitative phenomena (see Gentner, 1983)	$R^2 = 0.94$ with significant correlation, $p < 0.05$.
Eye gaze duration-per-character decreases with increased preview window size.	(see Figure 2 in Wu and Liu, 2008b).	R^2 of the simulated fixation time is 0.97 (*APE* = 21.6%).

Table 5 Summary of transcription typing phenomena modelled in QN-ACTR (continued)

Phenomena description	*Empirical human results*	*QN-ACTR simulation results*
	Other phenomena	
Eye saccade size	Four characters, averaged from multiple studies (for details, see Rayner, 1998).	4.1 characters (*APE* = 2.5%).
Eye fixation duration	400 ms, averaged from multiple studies (for details, see Rayner, 1998).	705 ms (*APE* = 76.3%).

Table 6 Descriptions, values, and sources of parameters used in the transcription typing model

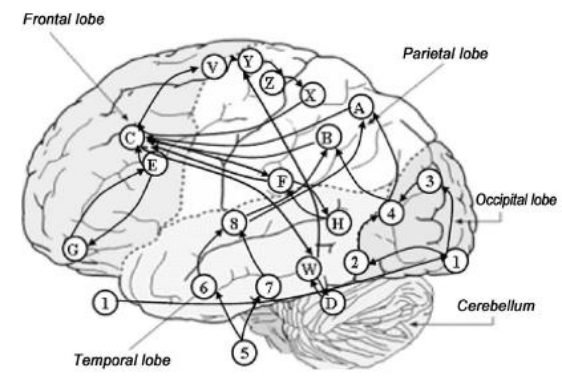
Parameter	*Description*	*Value and source*
α_i	Learning rate alpha	0.001 (Heathcote et al., 2000; Wu and Liu, 2008b)
A_i	Expected minimal processing time after intensive practice	21.5 ms (Anderson and Lebiere, 1998; Card et al., 1983)
B_i	Change of expected processing time from the beginning to the end of practice	50 ms (Anderson and Lebiere, 1998; Card et al., 1983)
N_i	Total number of digrams that have been processed	15,000,000 for skilled typists (Wu and Liu, 2008b)
:imaginal-delay	Determine the time for the imaginal module to form a chunk of imaginal representation.	0.100 s (Mehlhorn and Marewski, 2011)
:lf	Latency factor for declarative retrieval time. Larger values lead to longer memory retrieval time.	0.003 (Budiu and Anderson, 2004)
:bll	Base level learning parameter for chunk activation. Larger values lead to faster activation decay.	0.3 (Pavlik and Anderson, 2005)
:rt	Retrieval threshold. Set the minimum activation a chunk must have to be able to be retrieved.	–0.704 (Pavlik and Anderson, 2005)
:ans	Set the instantaneous noise added to chunk activation.	0.5 (Anderson and Matessa, 1997)
:tone-recode-delay	Determine the auditory perception time to recode a tone sound.	0.05 s (Byrne and Anderson, 2001)
saccade duration	Saccade movement duration	20 ms for saccade execution, plus an additional 2 ms for each degree of visual angle (Salvucci, 2001)

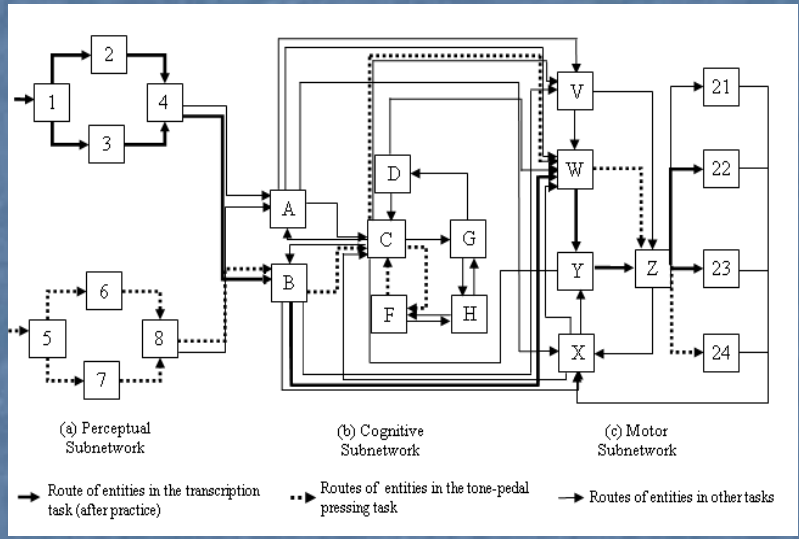
Note: The first four parameters are from the QN-MHP architecture, whereas the other parameters are from the ACT-R architecture

Source: adapted from Cao and Liu (2012b)

4.2 Results

The model simulated various transcription typing tasks and produced behavioural results. While the model is performing the task, the task visualisation feature in





Queueing Network

Perceptual Subnetwork	Cognitive Subnetwork	Motor Subnetwork
1. Common visual processing (eyes, lateral geniculate nucleus, superior colliculus, primary and secondary visual cortex) 2. Visual recognition (dorsal system) 3. Visual location (ventral system) 4. Visual recognition and location integration (distributed parallel area including the connections among V3 , V4 and V5, superior frontal sulcus, and inferior frontal gyrus) 5. Common auditory processing (middle and inner ear) 6. Auditory recognition (area from dorsal and ventral cochlear nuclei to the inferior colliculus) 7. Auditory location (area from ventral cochlear nucleus to the superior olivary complex) 8. Auditory recognition and location integration (primary auditory cortex and planum temporale)	A. Visuospatial sketchpad (right-hemisphere posterior parietal cortex) B. Phonological loop (left-hemisphere posterior parietal cortex) C. Central executive (dorsolateral prefrontal cortex (DLPFC), anterior-dorsal prefrontal cortex (ADPFC) and middle frontal gyrus (GFm)) D. Long-term procedural memory (striatal and cerebellar systems) E. Performance monitor (anterior cingulate cortex) F. Complex cognitive function: decision, calculation, anticipation of stimulus in simple reaction etc. (intraparietal sulcus (IPS), the superior frontal gyrus (SFS), the inferior frontal gyrus (GFi), the inferior parietal cortex and the ventrolateral frontal cortex, the intraparietal sulcus and the superior parietal gyrus) G. Goal initiation (orbitofrontal region and amygdala complex) H. Long-term declarative & spatial memory (hippocampus and diencephalons)	V. Sensorimotor integration (premotor cortex) W. Motor program retrieval (basal ganglia) X. Feedback information collection (somosensoy cortex) Y. Motor program assembling and error detecting (supplementary motor area (SMA) and the pre-SMA) Z. Sending information to body parts (primary motor cortex) 21-25: Body parts: eye, mouth, left hand, right hand, foot

(Queueing Network) Human Performance Modeling

Yili Liu

Dept of Industrial and Operations Engineering
University of Michigan
Ann Arbor, MI 48109

Changxu Wu

Dept of Industrial and Systems Engineering
State Univ of New York at Buffalo

Workshop Plan

- Importance of HPM
- Overview of HPM
- Overview of Queuing Network HPM
- QN Modeling of mental architecture, RT and accuracy
- QN Modeling of multitask performance and workload
- Q/A and Discussion

Importance of HPM

1. For systematic description and organization of knowledge and facts
2. For identification of gaps in knowledge, facts, and research
3. For extrapolation, prediction, and making inference
4. For theoretical explanation of facts and generation of hypothesis
5. For design evaluation of alternatives
6. For “virtual human” simulation in system development and analysis
7. For comprehensive and rigorous consideration of “all” related factors

Overview of HPM

Conceptual (qualitative) versus Computational (quantitative) HPMs

Mathematical, Statistical, and Symbolic HMPs

Need for Integration, Unification, Synthesis
(to reduce fragmentation)

Need for Strong Psychological and Neural Foundation
(as HUMAN models)

Overview of Queuing Network HMP

- WHY QN?

Queueing Network HPM

Philosophy:

1. Step-by-step HUMAN modeling:
   from basic (fundamental)
   to complex (composite behavior)

2. Synthesis, unification, integration
   of theories and methods

Related Journal Publications

- Liu, Y. (1996). "Queuing network modeling of elementary mental processes," *Psychological Review*, vol. 103, pp. 116-136.

- Wu, C., and Liu, Y. (2008). "Queuing network modeling of the Psychological Refractory Period (PRP)," *Psychological Review*, 115(4), pp.913-954.

Related Journal Publications (Continued)

- Liu, Y., Feyen, R., and Tsimhoni, O. (2006). "Queuing Network-Model Human Processor (QN-MHP): A Computational Architecture for multitask performance in human-machine systems," *ACM Transactions on Computer-Human Interaction*, vol. 13, no. 1. pp. 37-70.

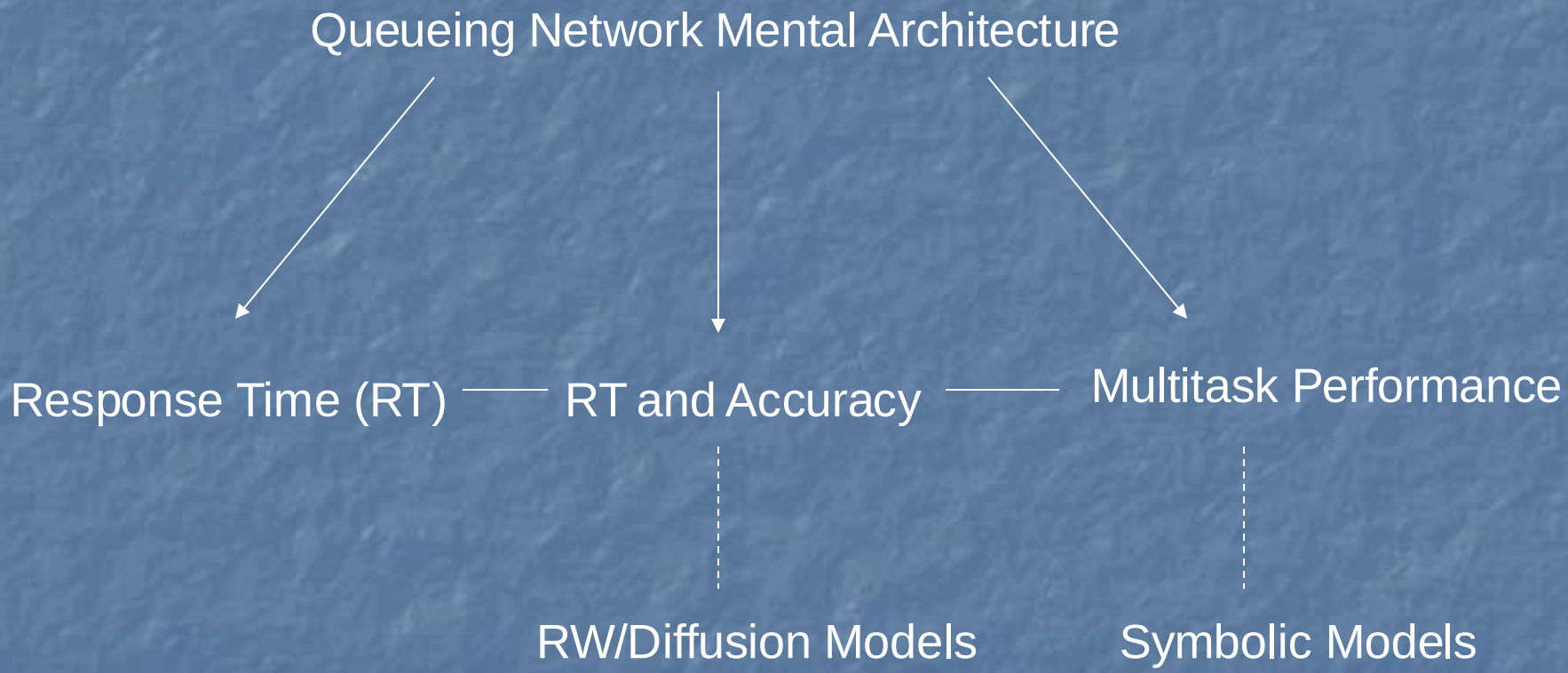
- Wu, C., and Liu, Y. (2008). "Queuing network modeling of transcription typing," *ACM Transactions on Computer Human Interaction*. 15(1), Article 6, pp. 1-45.

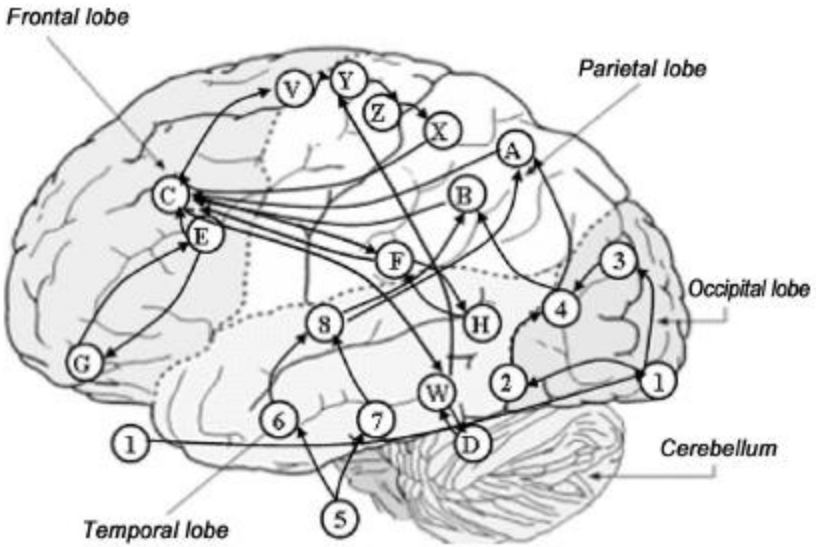
Related Journal Publications (Continued)

- **Liu, Y. (1997). "Queuing network modeling of human performance of concurrent spatial and verbal tasks," *IEEE Transactions on Systems, Man, Cybernetics*, vol. 27, pp. 195-207.**
- **Lim, J., and Liu, Y. (2008). "Reinforcement learning in eye movement: Modeling the influences of top-down and bottom-up processes," in print in *IEEE Transactions on Systems, Man, and Cybernetics.***
- **Wu, C., Liu, Y., and Walsh, C. (2008). "Queuing network modeling of a real-time psychophysiological index of mental workload--P300 in evoked brain potential (ERP)," *IEEE Transactions on Systems, Man, and Cybernetics,* 38(5), 1068-1084.**

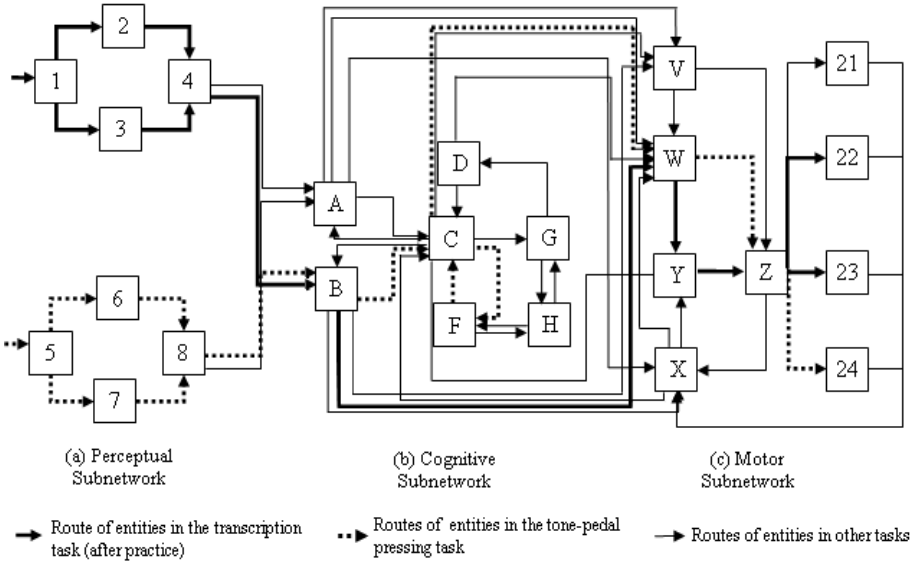
Related Journal Publications (Continued)

- **Bi, L., Tsimhoni, O., and Liu, Y. (2008). "Use image-based metrics to model pedestrian detection performance with night-vision systems," to appear in *IEEE Transactions on Intelligent Transportation Systems.***
- **Wu, C., Tsimhoni, O., and Liu, Y. (2008). "Development of an adaptive workload management system with the Queuing Network-Model Human Processor (QN-MHP)," *IEEE Transactions on Intelligent Transportation Systems*, 9(3), 463-475.**
- **Wu, C., and Liu, Y. (2007). "Queuing network modeling of driver workload and performance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 3, pp. 528-537.**
- **Lim, J. Tsimhoni, O., and Liu, Y. (2008a). "Investigation of driver performance with night vision and pedestrian detection systems, Part 1: Empirical study on visual clutter," to appear in *IEEE Transactions on Intelligent Transportation Systems.***
- **Lim, J. Tsimhoni, O., and Liu, Y. (2008b). "Investigation of driver performance with night vision and pedestrian detection systems, Part 2: Queuing network performance modeling," to appear in *IEEE Transactions on Intelligent Transportation Systems.***
- **Wu, C., and Liu, Y. (2007). *"Interface makeover of a software tool for modeling human performance and workload," Ergonomics in Design, 15(2), pp. 8-14.***
- **Wu, C., and Liu, Y. (2008). "Development and evaluation of a software package for predicting human performance and mental workload in interface design and evaluation," *Computers and Industrial Engineering*, in press.**


Queueing Network Mental Architecture
Response Time (RT)
RT and Accuracy
Multitask Performance
RW/Diffusion Models
Symbolic Models



Human Brain



Queueing Network

Queuing Network

1. A hybrid network
   (with serial and parallel components)

2. Analysis of both *queuing* and *processing*
   (queuing as a task coordination mechanism)

A screenshot of the current QN-MHP graphical interface for defining a dual task
(vehicle steering as Task 1 and a secondary task as Task 2, both specified graphically by a modeler)

in Wu and Liu, 2007, *Ergonomics in Design*, Winter 2007 issue