A Computer-Aided Usability Testing Tool for In-Vehicle Infotainment Systems

Fred Feng [a, *], Yili Liu [b], and 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

1. Introduction

Usability testing is one of the key processes of user-centered design for interactive systems. The traditional approaches for usability testing typically rely on human test subjects, who are asked to perform certain tasks using the designs being tested. The subjects' interactions with the designs and their performances and comments are recorded and then analyzed. This approach is often considered of great value as it provides direct information about how people use and think of the designs. However, they are often time-consuming and costly to conduct. And thus the number of designs that can be afforded for testing is often very limited. Compared with this traditional approach, digital human models could be used as a supplemental method to test design concepts and prototypes with lower costs in both time and manpower (Chaffin, 2007). They allow the user interface (UI) designers to compare multiple design concepts, explore a larger design space, and address usability issues at the early stages of the design process.

Digital human models have been a valuable asset in many industries to analyze the physical ergonomics problems of product or biomechanical injury risk of workplace designs (Feyen, et al., 2000; Chaffin, 2007). One of the most successful digital human models is Jack which was originally developed by the University of Pennsylvania and later became commercial software (SIEMENS) to address the ergonomic aspects of manual operations. The University of Michigan developed Human Motion Simulation (HUMOSIM) framework that could simulate realistic human movements and be used for ergonomic analysis of products and workplaces (Reed, et al.,

[*] Corresponding author at: University of Michigan Transportation Research Institute, 2901 Baxter Road, Ann Arbor, MI, USA 48109
Email: fredfeng@umich.edu

2006). Despite the success of these digital models, they are largely based on human biomechanics and focused on the physical ergonomics (e.g., related to human posture and movements). In this paper we focus on modeling drivers simultaneously performing driving and other secondary tasks which are primarily perceptual and cognitive in nature.

Many driver models have been developed in the past with focuses on particular aspects of the driving task, such as car-following, lane-keeping, and overtaking. Several models have been developed based on task-independent cognitive architectures, including the ACT-R driver models (Salvucci, 2006) and the QN-MHP (Queuing Network-Model Human Processor) driver model (Wu & Liu, 2007a). QN-MHP simulates the human cognition as a queuing network of information processing servers derived from the psychological and neuroscience fields (Liu 1996, 1997, Liu, et al., 2006). The modeling framework has been successfully used to simulate a wide variety of human performance including transcription typing (Wu & Liu, 2008a), visual search (Feng & Liu, 2013), vehicle steering (Tsimhoni & Liu, 2003), driver performance and workload (Wu & Liu, 2007). Recent work has also fully implemented ACT-R as a special case of the queuing network model (Cao & Liu, 2013). Nonetheless, even for the generic cognitive architecture-based models, there is still a gap that prevents them from being deployed in practice. This is largely because using these models usually requires the designers to have a fairly deep understanding of their theoretical foundations and mechanisms as well as programming skills to setup simulations for any particular task at hand.

Pew (2008) listed three major challenges for a successful digital human model: (1) Simplified model development; (2) better capabilities for articulating and visualizing how the models work, and (3) model validation. With these three challenges in mind in this paper we aim at developing a Computer-Aided Engineering (CAE) software toolkit that could make quantitative predictions of the usability of in-vehicle infotainment systems to be tested. To achieve this goal, the core of the software needs to be built upon a model that is capable of simulating human multi-task performance that includes both driving and a wide variety of secondary tasks in an accurate and reliable way. In addition, the software also needs to allow the UI designers to set up the simulation with a user-friendly and easy-to-follow interface.

2. Methods

2.1. Software Development

2.1.1. Model structure

QN-MHP simulates human cognition as a queuing network of information processing servers that can be grouped into three subnetworks (i.e., perceptual, cognitive, and motor subnetwork). Each subnetwork is composed of individual servers that represent certain brain functions for processing information. More details on the QN-MHP cognitive architecture can be found in Liu, et al., (2006).

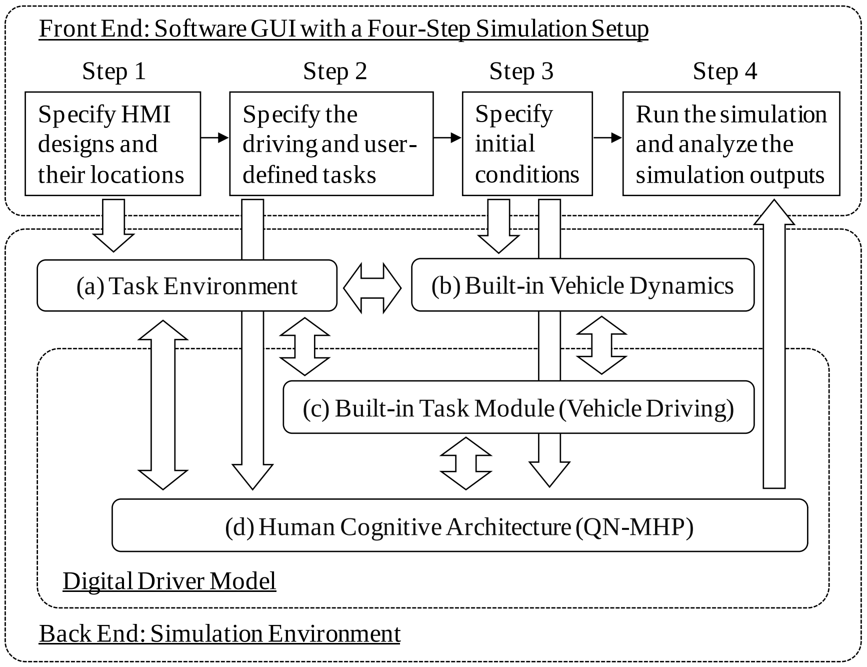

Front End: Software GUI with a Four-Step Simulation Setup
Step 1
Specify HMI designs and their locations
Step 2
Specify the driving and user-defined tasks
Step 3
Specify initial conditions
Step 4
Run the simulation and analyze the simulation outputs
(a) Task Environment
(b) Built-in Vehicle Dynamics
(c) Built-in Task Module (Vehicle Driving)
(d) Human Cognitive Architecture (QN-MHP)
Digital Driver Model
Back End: Simulation Environment

Fig. 1. Software structure and major components of the driver model

Figure 1 shows the software structure and major components of the driver model. At the front end is a graphic user interface (GUI) that allows the software users to setup and run the simulation in four steps. At the back end is the simulation environment with four major components. The task environment (box a in Figure 1) represents the environment with which the digital driver could interact with. It stores the information about the driving environments once they are specified during the simulation setup. During the simulation run it receives outputs from the QN-MHP's body part servers (e.g., the hand server to turn the steering wheel), and supplies updated input stimulus to the QN-MHP (box d)'s perceptual servers. The vehicle dynamics (box b) is a built-in module that receives input from the QN-MHP's driving related actions (e.g., steering), and generates the vehicle responses which are used to update the driving environment in box a. Currently a three-Degree-Of-Freedom (longitudinal, lateral and yaw) bicycle model is implemented for its simplicity. The QN-MHP (box d) represents the generic digital human. Its procedural long-term memory server stores the task information once it is specified in the simulation setup. During the simulation the task information is used as instructions to the digital driver on how to perform the tasks on the QN-MHP framework. During the simulation, the QN-MHP is able to generate the task performance based on the information about the status of the queuing network. The software toolkit was implemented in MATLAB/Simulink software. Figure 2 is a screenshot of the implementation of the back end described above.

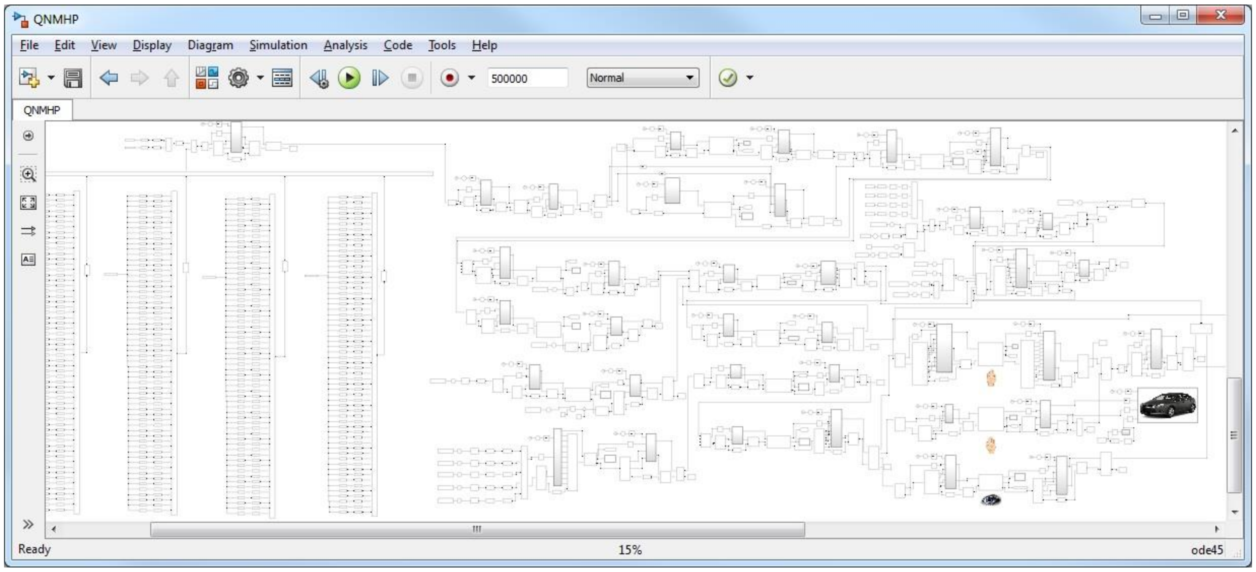
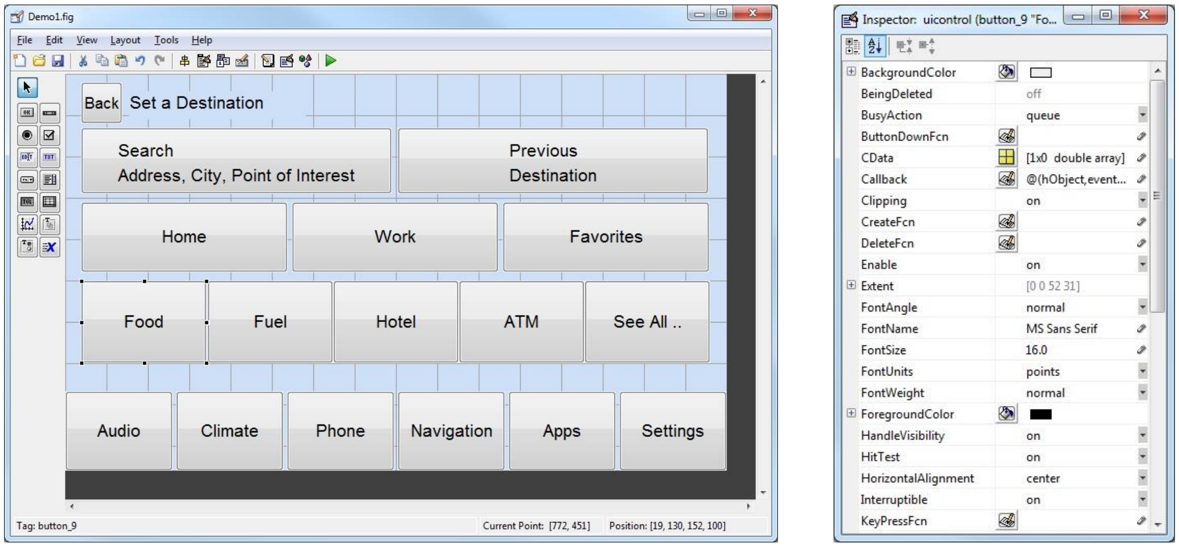

QNMHP
File Edit View Display Diagram Simulation Analysis Code Tools Help
500000
Normal
QNMHP
Ready
15%
ode45

Fig. 2. A screenshot of the model implementation in MATLAB

2.1.2. Design prototyping

To evaluate a design, first a prototyping tool is needed to create a digital mockup of the design that the driver model could interact with. The interaction may include the driver model perceiving information from the digital mockup (e.g., visual information from a display, auditory information from a warning sound), and the driver model generating motor actions upon the digital mockup (e.g., pressing a button). A prototyping tool was developed using MATLAB GUIDE (Graphical User Interface Design Environment, see Figure 3a). 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 by editing the property inspector of the selected object (see Figure 3b). A GUI object's behavior upon user actions (e.g., a button being clicked) could be specified in its callback functions. MATLAB GUIDE also supports WYSIWYG (What You See Is What You Get) and allows users to check the current design's look and behavior at any time during the prototyping process. Once a design mockup is created, it can be saved as a stand-alone file that can be imported in to the simulation environment during the simulation setup.


Demo1.fig
File Edit View Layout Tools Help
Back
Set a Destination
Search
Address, City, Point of Interest
Previous
Destination
Home
Work
Favorites
Food
Fuel
Hotel
ATM
See All ..
Audio
Climate
Phone
Navigation
Apps
Settings
Tag: button_9
Current Point: [772, 451]
Position: [19, 130, 152, 100]
Inspector: uicontrol (button_9 "Fo...

BackgroundColor	▭
BeingDeleted	off
BusyAction	queue
ButtonDownFcn	
CData	[1x0 double array]
Callback	@(hObject,event...
Clipping	on
CreateFcn	
DeleteFcn	
Enable	on
Extent	[0 0 52 31]
FontAngle	normal
FontName	MS Sans Serif
FontSize	16.0
FontUnits	points
FontWeight	normal
ForegroundColor	■
HandleVisibility	on
HitTest	on
HorizontalAlignment	center
Interruptible	on
KeyPressFcn	

(a) Main screen (b) Object property inspector

Fig. 3. Using MATLAB GUIDE for prototyping interface designs

2.1.3. Simulation setup

Once the digital mockups of the to-be-tested designs are created, the software user 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 a few steps. Figure 4 shows the main window of the software GUI.

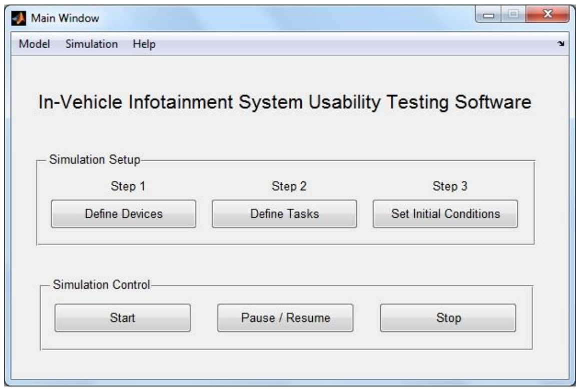
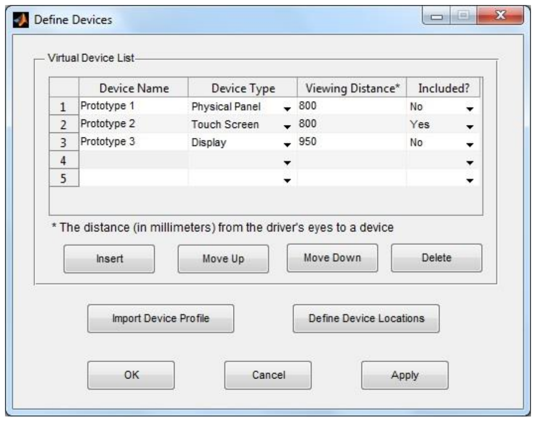


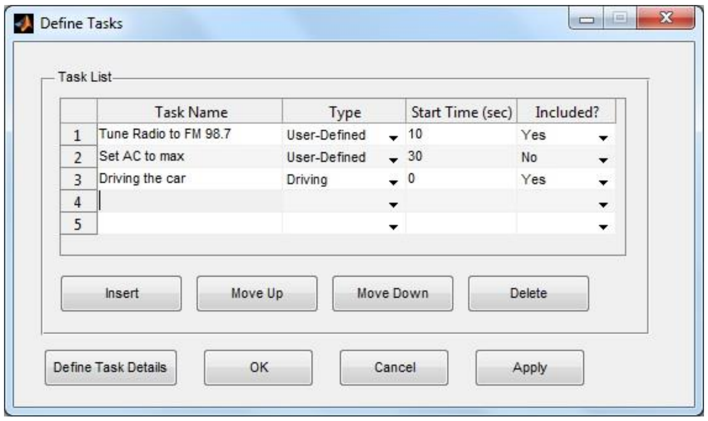
Fig. 4. Main window of the usability testing tool

Step 1: The setup starts with adding virtual devices (i.e., digital mockups created in Section 2.1.2 above) to the task environment. A virtual device could be added by filling out a single row in the virtual device table (see Figure 5a) with a device name, type (currently support pre-defined types of *physical panel*, *touch screen*, and *display*), viewing distance, and an option of whether the device shall be included in the simulation (note this option is to allow users to temporarily disable a device rather than having to delete it and add it back later). The device profile could be imported to the simulation environment from the design mockup file that is saved during the prototyping.



	Device Name	Device Type	Viewing Distance*	Included?
1	Prototype 1	Physical Panel	800	No
2	Prototype 2	Touch Screen	800	Yes
3	Prototype 3	Display	950	No
4				
5				

* The distance (in millimeters) from the driver's eyes to a device

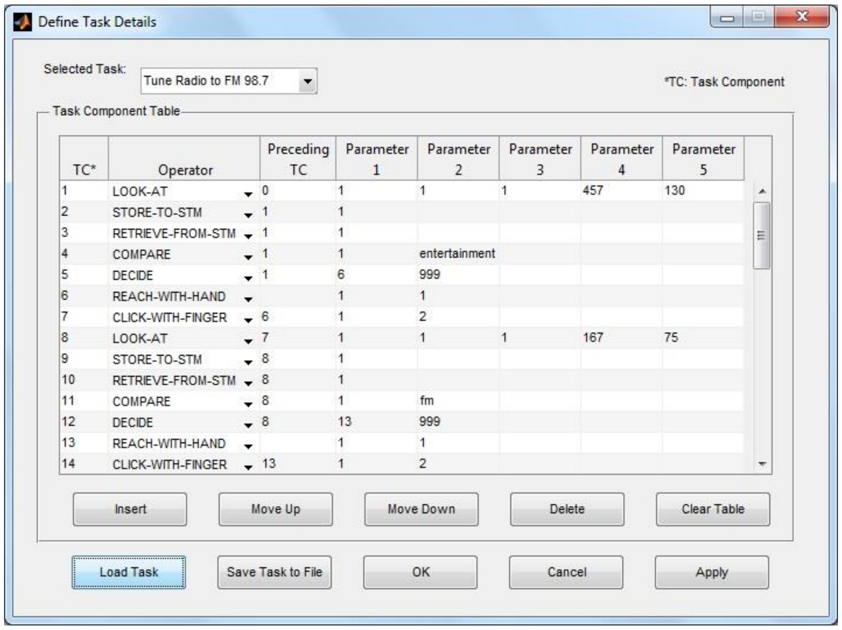


	Task Name	Type	Start Time (sec)	Included?
1	Tune Radio to FM 98.7	User-Defined	10	Yes
2	Set AC to max	User-Defined	30	No
3	Driving the car	Driving	0	Yes
4				
5				

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

Fig. 5. Define to-be-tested designs and tasks

Step 2: After the virtual devices are specified in Step 1, the related task information needs to be specified as instructions to the driver model on how to perform the task on the QN-MHP framework. Similar to adding a new device in Step 1, a new task could be added by filling out a single row in the task table (see Figure 5b) with a task name, type (either *user-defined* or *driving*), and task starting time. The details of a user-defined task can be specified in a new window (see Figure 6) in a NGOMSL-style (Natural Goals, Operators, Methods, and Selection rules Language, Kieras, 1996). The NGOMSL task analysis breaks down the task in a “top-down, breadth-first” manner into “atomic-level” Task Components (TC). Each task component is associated with a task independent context-free QN-MHP operator from the QN-MHP operator library. The operators are set with parameters either explicitly by the modeler when specifying the task. The sequential dependency of the task components is set by the modeler. More details on how to perform a NGOMSL-style task analysis for QN-MHP could be found in Liu et al. (2006). The built-in driving task could be included by simply creating a new task and selecting “driving” as its type.



Selected Task: Tune Radio to FM 98.7

*TC: Task Component

Task Component Table

TC*	Operator	Preceding TC	Parameter 1	Parameter 2	Parameter 3	Parameter 4	Parameter 5
1	LOOK-AT	0	1	1	1	457	130
2	STORE-TO-STM	1	1				
3	RETRIEVE-FROM-STM	1	1				
4	COMPARE	1	1	entertainment			
5	DECIDE	1	6	999			
6	REACH-WITH-HAND		1	1			
7	CLICK-WITH-FINGER	6	1	2			
8	LOOK-AT	7	1	1	1	167	75
9	STORE-TO-STM	8	1				
10	RETRIEVE-FROM-STM	8	1				
11	COMPARE	8	1	fm			
12	DECIDE	8	13	999			
13	REACH-WITH-HAND		1	1			
14	CLICK-WITH-FINGER	13	1	2			

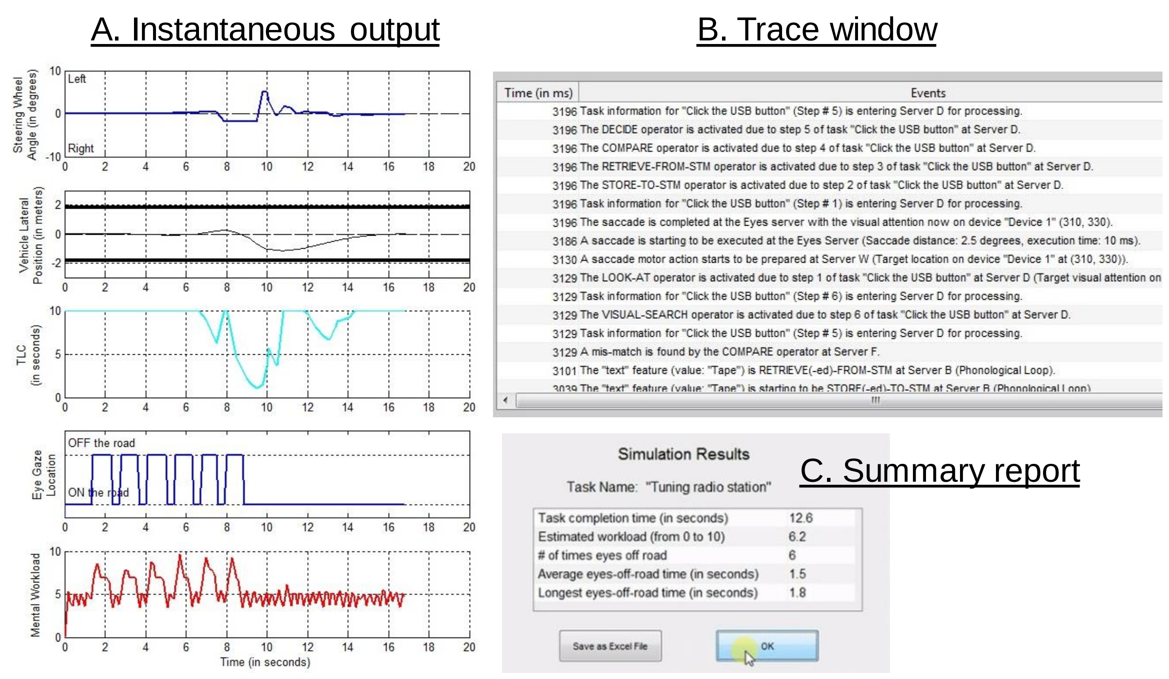
Fig. 6. Define details of a user-defined task

Step 3: Once the device and task information are specified in Step 1 and 2, the initial conditions of the simulation need to be specified. For example, where the digital driver's visual attention is initially located, what is the initial states of the devices (e.g., which screen a display is on) and the vehicle (e.g., initial speed). Once Step 3 is completed, the user can run the simulation.

2.1.4. Simulation output

During the simulation the software is able to generate instantaneous outputs of the vehicle states and the digital driver's performance. Figure 7 shows the simulation outputs including the steering wheel angle, vehicle lane position, and time-to-lane-crossing, as well as the driver eye gaze location (on or off the road) and estimated mental workload. 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 (2007a).

The software also generates a trace window that shows in chronological order the important event log that have occurred at the QN-MHP server level (e.g., at time 12.0 s an eye saccade is completed at the Eyes server with visual attention now on device X at location (100, 200) mm). This event log could be useful for the users to verify the simulation and examine the details of the task execution. When a simulation is completed, the software generates a summary report on the simulation results, as well as a time-series table containing detailed information of the driver and vehicle states during the course of the simulation.



Time (in ms)	Events
3196	Task information for "Click the USB button" (Step # 5) is entering Server D for processing.
3196	The DECIDE operator is activated due to step 5 of task "Click the USB button" at Server D.
3196	The COMPARE operator is activated due to step 4 of task "Click the USB button" at Server D.
3196	The RETRIEVE-FROM-STM operator is activated due to step 3 of task "Click the USB button" at Server D.
3196	The STORE-TO-STM operator is activated due to step 2 of task "Click the USB button" at Server D.
3196	Task information for "Click the USB button" (Step # 1) is entering Server D for processing.
3196	The saccade is completed at the Eyes server with the visual attention now on device "Device 1" (310, 330).
3186	A saccade is starting to be executed at the Eyes Server (Saccade distance: 2.5 degrees, execution time: 10 ms).
3130	A saccade motor action starts to be prepared at Server W (Target location on device "Device 1" at (310, 330)).
3129	The LOOK-AT operator is activated due to step 1 of task "Click the USB button" at Server D (Target visual attention on
3129	Task information for "Click the USB button" (Step # 6) is entering Server D for processing.
3129	The VISUAL-SEARCH operator is activated due to step 6 of task "Click the USB button" at Server D.
3129	Task information for "Click the USB button" (Step # 5) is entering Server D for processing.
3129	A mis-match is found by the COMPARE operator at Server F.
3101	The "text" feature (value: "Tape") is RETRIEVE(-ed)-FROM-STM at Server B (Phonological Loop).
3039	The "text" feature (value: "Tape") is starting to be STORE(-ed)-TO-STM at Server B (Phonological Loop)

Simulation Results

Task Name: "Tuning radio station"

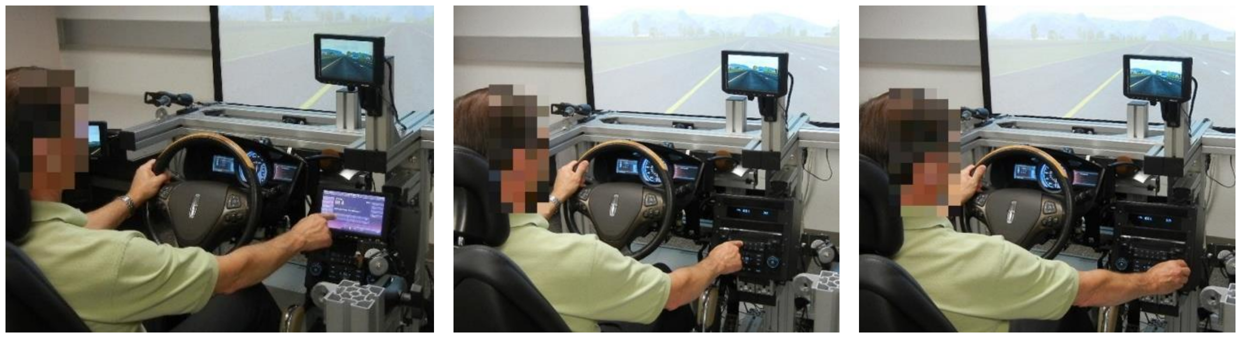
Task completion time (in seconds)	12.6
Estimated workload (from 0 to 10)	6.2
# of times eyes off road	6
Average eyes-off-road time (in seconds)	1.5
Longest eyes-off-road time (in seconds)	1.8

Fig. 7. Simulation outputs

2.2. Validation Experiment

2.2.1. Participants and tasks

An experiment with twenty human drivers (10 male and 10 female) was conducted to validate the software outputs. All the participants were employees from a company in the United States, and had gotten their driver license for at least one year. For the age distribution, 10 (50%) participants were 20-39 years old, 9 (45%) participants were 40-59 years old, and 1 (5%) participant was 60-69 years old. The time the participants spent during the experiment counted as their company work time. The experiment was conducted in a fix-based driving simulator (shown in Figure 8). The front road scene was projected on a flat screen in front of the driving buck.



(a) using the touch screen (b) using the physical buttons (c) using the knob

Fig. 8. Participants performing the radio-tuning task while driving the simulator

The participants were asked to drive the simulator on a virtual highway and maintain a speed of about 60-70 miles per hour (97-113 km/h). The highway is a square loop with two lanes in one direction. The participants were asked to stay in the left lane in which there is no other vehicle. The participants were asked to always put both hands on the steering wheel except when performing the radio-tuning task.

2.2.2. Experimental design

There are two independent variables in this experiment: (1) task condition (single or dual task), and (2) radio-tuning method (using the physical buttons, knob, or touch screen). In the single task condition, the participants were instructed to perform the radio-tuning task while the simulator is parked on road side. In the dual task condition, the participants were asked to drive the simulator, and at certain points, they were verbally instructed by the experimenter to start the radio-tuning task when they believe it is safe to do so. Three radio-tuning methods were tested in the experiment with the details described in Table 1.

Table 1. Task procedures of tuning the radio to FM 98.7

Steps	Method 1: physical buttons	Method 2: knob	Method 3: touch screen
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'	---	Press '9'
Step 5	Press '8'	---	Press '8'
Step 6	Press '7'	---	Press '7'
Step 7	---	---	Press "Enter"

There are three dependent variables in this study: (1) task completion time, (2) total eyes-off-road time (TEORT), and (3) subjective mental workload. One camera (sampling rate 30 Hz) facing the center console area was used to capture the participant's operations on the physical panel or touch screen. In the single task condition, task completion time is measured from the time when the subject started to move his/her right hand away from the steering wheel to the time when the correct radio frequency (e.g., FM 98.7) is reached. For the dual task condition, the task completion time is measured from the time when the participant started to move his/her eyes away from the road to the device, to the time when the correct radio frequency is reached. Another camera (sampling rate 30 Hz) was used to capture the participant's face, which was used to determine whether the participant was looking on or off the road. The total eyes-off-road time (TEORT) could then be calculated for each radio-tuning task. At the end of each session the subjects were asked to rate their workload using the standard NASA-Task Load Index (NASA-TLX, Hart & Staveland, 1988). A full factorial within-subject design was used in this experiment. Two independent variables and three trials for each combination give a total of 18 (= 2x3x3) trials for each participant.

2.2.3. Procedures

Once the participants arrive at the laboratory and complete the consent form, they were given an introduction using slides to ensure that they understood the tasks they were about to perform. This was followed by a practice session for both the radio-tuning and driving task. Then the data collection started. 10 (50%) participants started with the single task session, while 10 (50%) started with the dual task. 10 (50%) participants started with the method using the touch screen, 5 (25%) started with the method of using the physical buttons, and 5 (25%) started with us the method of using the knob.

2.3. Task Modeling

2.3.1. Radio-tuning task

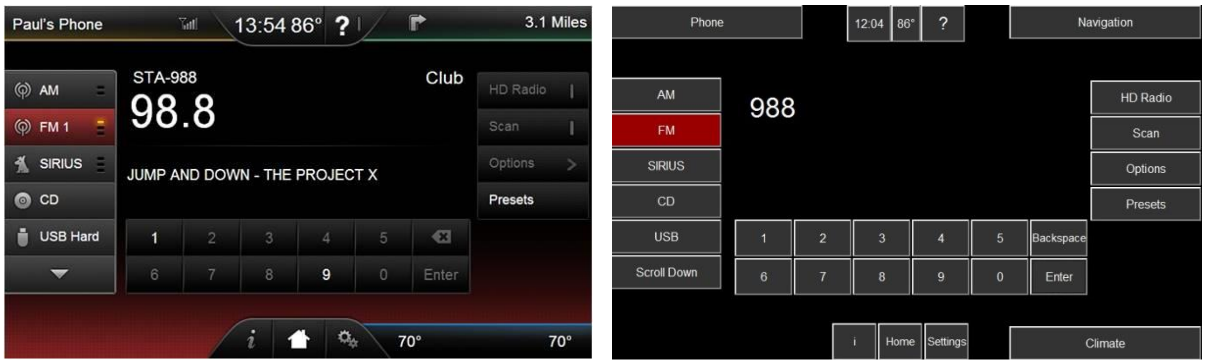


Fig. 9. The touch screen used in the experiment (left) and its digital mockup (right)

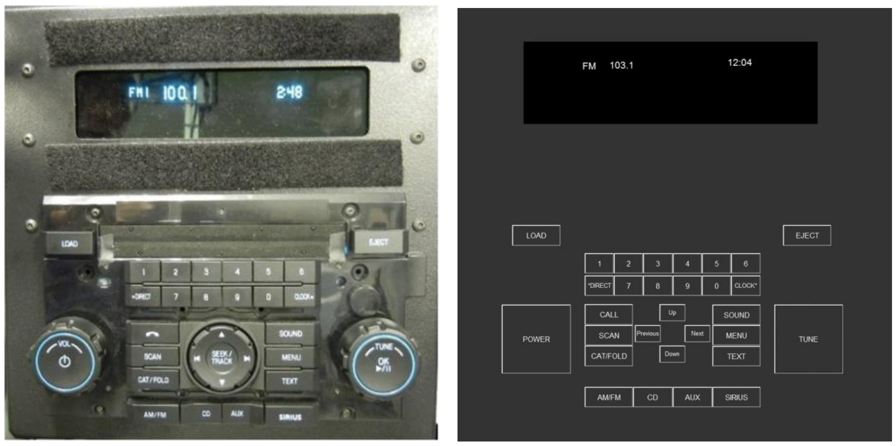


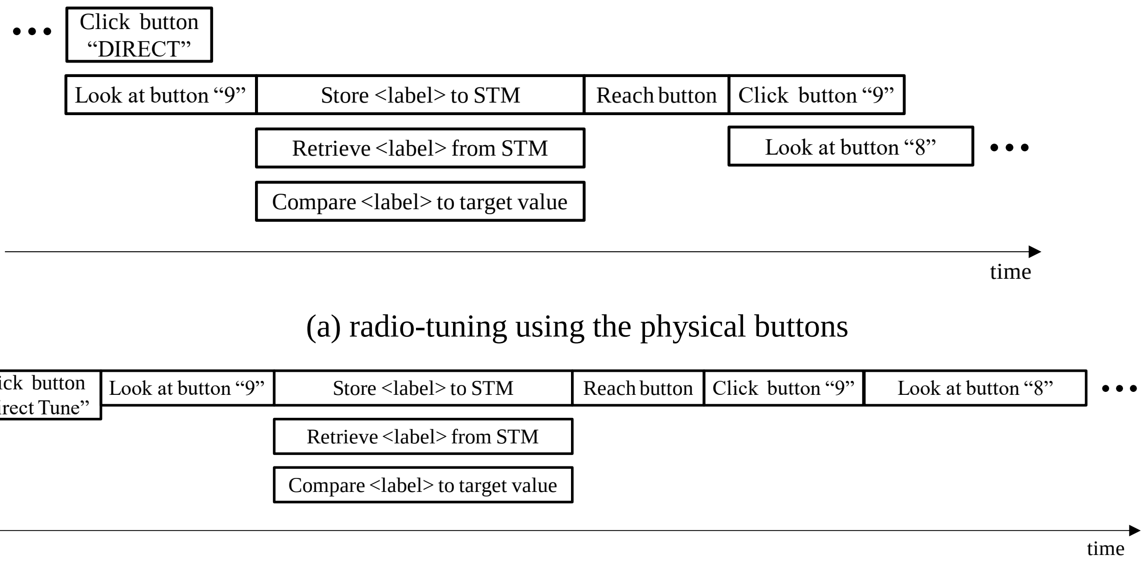
Fig. 10. The physical panel used in the experiment (left) and its digital mockup (right)

To simulate the radio-tuning task, A NGOMSL-style task analysis needs to be conducted. Figure 11 shows the NGOMSL-style task analysis of tuning the radio using the physical buttons. The task analyses of using the knob and touch screen follow a similar format.

```
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

The task analyses of tuning the radio using the knob and touch screen follow a similar format. However, there are differences on the sequential dependency of the task components (TCs) due to the inherent characteristics of the devices. For example, clicking a virtual button on a touch screen without tactile feedback requires visual attention to confirm that the button is successfully clicked (by using visual cues of, for example, a change of the button's color). And in some cases, the next button to click (e.g., the 'FM' button in the entertainment screen) would simply not appear before the previous button is clicked (e.g., the 'Entertainment' button in the home screen). For the physical buttons on the other hand, searching for the next button could start as soon as the previous button is reached by the hand, with the assumption that clicking on a physical button could be confirmed by the tactile feedback without the need for visual attention. We do assume that the driver needs the visual guidance for reaching the button (i.e., no 'touch typing'), so searching for the next button could start as early as the button is reached by hand, but not earlier.



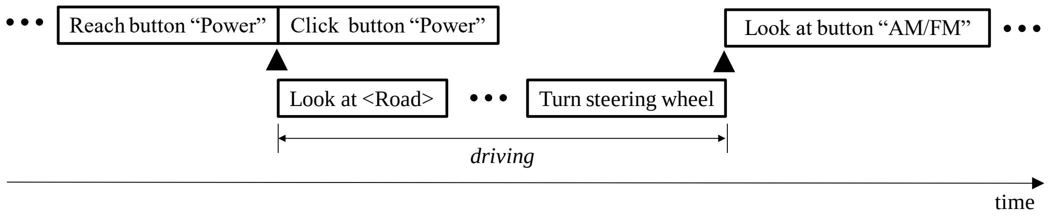
(a) radio-tuning using the physical buttons

(b) radio-tuning using the touch screen

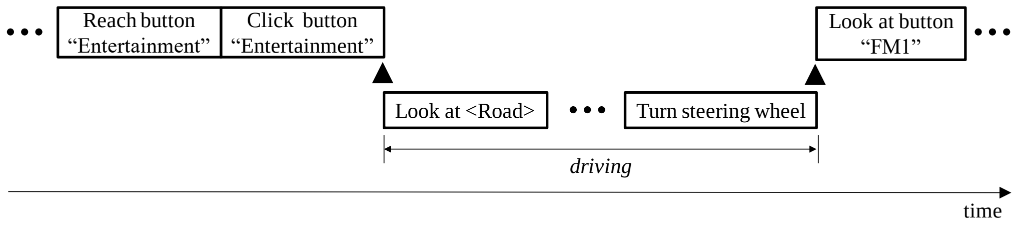
Fig. 12. Sequential dependency of the task components for single task

2.3.2. Multitasking modeling with driving included

Given that a secondary task by itself may take more than a few seconds to, when it is performed while driving, the driver is likely not able to complete the task with a single off-road glance. For this reason, the secondary task needs to be segmented so that the driver would operate the vehicle in a safer manner by not having his/her eyes off the road for too long. For the physical buttons, we assume the driver could use the tactile feedback to confirm the successful click of a button, and thus can start looking back to the road as soon as his/her hand reaches a button. However, when using the touch screen without the tactile feedback, the driver can only start looking at the road after the clicking is completed. Figure 13 illustrates the sequential dependency of the task components when tuning the radio using the physical buttons and touch screen while driving. These task specifications were loaded to the software as instructions on how to perform the task during the simulation.



(a) Using the physical panel



(b) Using the touch screen

Fig. 13. Sequential dependency of the task components of tuning radio while driving

3. Results

3.1. Task completion time

The task completion time in each experiment trial was calculated from the center-console-facing camera video using the method described in the above section. Wilcoxon signed-rank tests was conducted to test whether there is significant difference of task completion time between the two task conditions and among the three radio-tuning methods. The results show that by including the driving task, the task completion time is significantly longer when using the touch screen ($p < 0.001$), the physical buttons ($p < 0.05$), and the knob ($p < 0.050$). In both the parked and driving condition, the task completion time is significantly longer when using the touch screen than either the physical buttons or the knob (both $p < 0.001$ in both the parked and driving condition). However, there is no significant difference between the physical buttons and knob ($p = 0.500$ in the parked condition, and $p = 0.277$ in the driving condition). The model predictions of the task completion time are shown in Figure 14. The predictions are fairly similar to the empirical data. For the parked condition, the mean absolute percentage error (MAPE) is 5.7%. The root-mean-square error (RMSE) is 0.38 s. For the driving condition, the MAPE is 9.0%. The RMSE is 0.81 s.

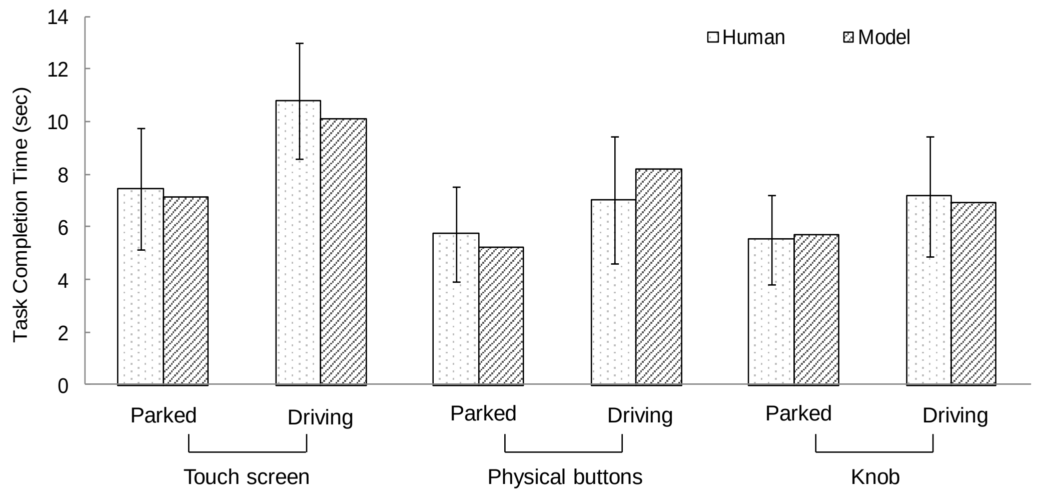


Fig. 14. Simulation results of task completion time in comparison to experiment results

3.2. Total Eyes-Off-Road Time (TEORT)

The TEORT in each experiment trial was calculated from the driver-facing camera video. Wilcoxon signed-rank tests show that the TEORT was significantly longer when using the touch screen than either the physical buttons or the knob (both p < 0.001). However, there is no significant difference between the physical buttons and knob (p = 0.709). The model outputs of the TEORT are shown in Figure 15. The predictions are fairly similar to the empirical data, with the MAPE of 12.1% and the RMSE of 0.71 s.

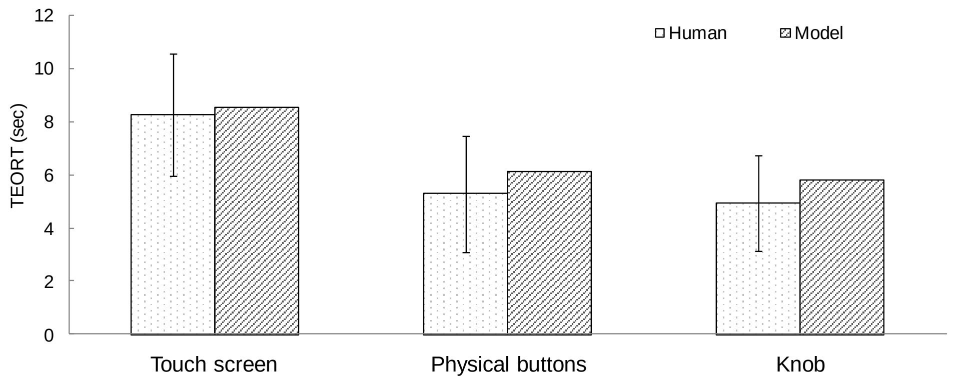


Fig. 15. Simulation results of TEORT in comparison to experiment results

3.2. Mental workload

The mental workload between the two task conditions and among the three radio-tuning methods were reported by the subjects using a 0-10 scale from NASA-TLX. Wilcoxon signed-rank tests show that by including the driving task, the mental workload is significantly higher when using any of the three radio-tuning method (all p < 0.001). In the parked condition, the mental workload when using the knob is significantly lower than both the physical buttons and touch screen (both $p < 0.05$), while there is no significant difference between the physical button and touch screen ($p = 0.088$). In the driving condition, the mental workload when using the knob is significantly lower ($p < 0.01$) than using the physical buttons, which is significantly lower ($p < 0.01$) than using the touch screen. The model predictions of the task completion time are shown in Figure 16. The predictions are fairly similar to the empirical data, except the model seems to over-estimate the workload when using the knob in the driving condition. For the parked condition, the MAPE is 14.1%, and the RMSE is 0.50. For the driving condition, the MAPE is 18.8%, and the RMSE is 1.13.

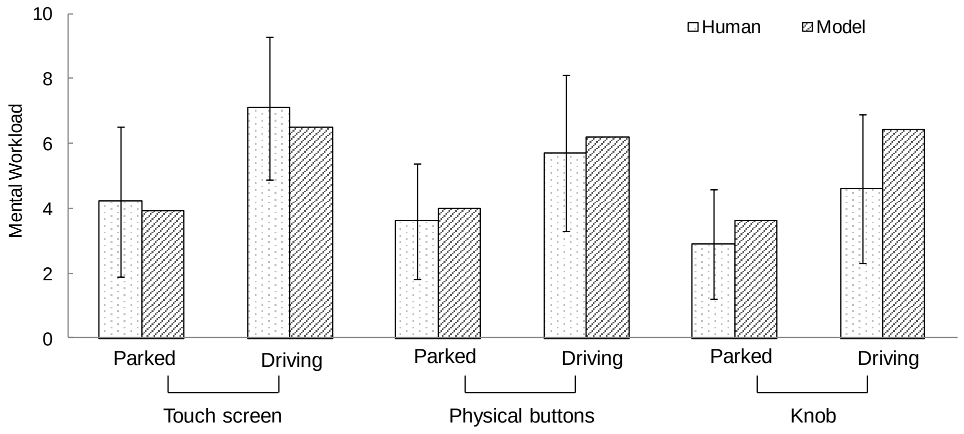


Fig. 16. Simulation results of mental workload in comparison to experiment results

4. Discussion

This paper describes the development and validation of a CAE software for designers of in-vehicle infotainment systems to predict and benchmark certain aspects of the system usability such as task completion time, eyes-off-road time, and mental workload. The software does not require the user to know any programming language to simulate tasks with specified designs, as the entire simulation can be set up within the software's GUIs by following a few rather simple steps. The software also adopts the MATLAB GUIDE as the design prototyping tool, which allows the users to create and edit design mockups graphically. These features enable designers to use the software as a usability testing tool with less training and reduced modeling time.

An experiment with human drivers was conducted to validate the software outputs. In general, the software could produce fairly similar outputs (task completion time, TEORT, and mental workload) to the results from the human drivers. The observed performance differences among the three radio-tuning methods and the two task conditions may be accounted for by the digital driver model in several aspects. The increased task completion time when driving is included, could be simply accounted by the fact that the radio-tuning task is partitioned into multiple eyes-off-road glances to allow the driver to move his/her visual attention back on the road. This essentially makes the task longer to complete. The task completion time using the physical buttons is shorter than the time using the touch screen, because the task using physical buttons is executed by the driver model in a more parallel fashion by, for example, clicking one button while simultaneously directing the visual attention to the next button. However, this is not possible for the touch screen, which requires the driver's visual attention to confirm if a button is successfully clicked before he/she could start searching for the next button. Similarly, the TEORT when using the physical buttons or knob is shorter than using the touch screen, because the secondary task and the driving task are executed in a more parallel fashion, so some of the task components (e.g., clicking a physical button) can be performed when the driver's eyes are on the road (see Figure 13 for details). The mental workload is higher when driving is included, because whenever the driver's visual attention is switched off the road while driving, the driver model initiates an internal clock to track the current eyes-off-road time, and switch the visual attention back to the road to make sure a single eyes-off-road glance never exceeds a certain safety value (e.g., 2 s). This essentially increases the QN-MHP server utilizations which are used to estimate the workload. These differences were naturally captured during setting up the simulation in the software. Lastly, the model seems to have over-estimated the workload when using the knob to turn the radio while driving. One possible explanation is that currently tuning a knob is modeled as a process that requires continuous visual attention at the radio frequency display to make sure the correct frequency is reached, while in reality a human driver may rely on the knowledge that each increment of the knob equals to 0.2 Hz to change the radio frequency to or near the target value without having to switch the visual attention off the road and increase the mental workload.

This work demonstrates the potential of using a CAE software toolkit as a usability testing tool for user interface designers. We are exploring several further developments of the software in several directions. First, currently the software requires the user to be able to conduct a NGOMSL task analysis (described in detail in Liu, et al, 2006). This may require some initial training time to the software user, and the accuracy of the model outputs may be affected by the accuracy of the task analysis. In the future we are planning to add "templates" for specifying typical in-vehicle operations at a level higher than the GOMS's perceptual, cognitive, and motor level. By doing so it may reduce the knowledge and skill required to conduct accurate task analysis, and thus reduce the required training time and the potential errors and discrepancy during the task analysis.

Second, we are currently using the MATLAB GUIDE for prototyping the design interfaces. In some applications, the designs may be already implemented on some other user interface languages (e.g., HTML, XML). To test these designs, the software users need to firstly recreate the designs in MATLAB GUIDE. In the future we may explore the techniques to support direct import of popular user interface formats into the software, so that the designs could be directly tested in their original formats. This would further reduce the time and efforts required to use the software.

Third, the software structure treats the driving task (box c in Figure 1) as a built-in extension module to the task-independent QN-MHP architecture (box d in Figure 1). This structure allows the software developers to add other extension modules in the future if the needs arise. For example, a potential module for flying an airplane could be added as another built-in task. This flexibility makes it possible to extend the scope of this software to the usability testing of other domains with human-in-the-loop systems.

5. Conclusions

This paper reports the development and validation of a computer-aided engineering (CAE) usability testing tool for user interface designers to predict and benchmark the usability of in-vehicle infotainment systems. A digital human model was built to simulate the driver cognition and performance based on the cognitive architecture of QN-MHP. At the front end of the software toolkit a graphical user interface (GUI) was developed that allows UI designers to create digital design prototypes and set up the tasks for simulation. The software is able to simulate a driver performing in-vehicle secondary tasks (e.g., tuning radios) while steering a vehicle, quantitatively correlate to driver's task performance such as task completion time, eyes-off-road time, and mental workload. To validate the software outputs, an experiment was conducted on a fix-base driving simulator with two typical in-vehicle infotainment systems (a physical panel and a touch screen). And a radio-tuning task were used as a test case. The results show that the software is able to generate task completion time, total eyes-off-road time, and mental workload estimates that are similar to the empirical data collected from human subjects. This toolkit has the potential to enable the UI designers to explore a larger design space and address usability issues at the early design stages with lower cost both in time and manpower.

References

Cao, S., & Liu, Y. (2013). Queueing network-adaptive control of thought rational (QN-ACTR): An integrated cognitive architecture for modelling complex cognitive and multi-task performance. *International Journal of Human Factors Modelling and Simulation*, 4(1), 63-86.

Chaffin, D. B., (2007), Human Motion Simulation for Vehicle and Workplace Design, Human Factors and Ergonomics in Manufacturing, Vol. 17 (5) 475 C484

Feng, F., & Liu, Y. (2013). Computational modeling of feature and conjunction visual search tasks using Queuing Network-Model Human Processor (QN-MHP), *2nd International Digital Human Modeling Symposium Proceedings*.

Feng, F., Liu, Y., Chen, Y., Filev, D., & To, C. (2014) Computer-aided usability evaluation of in-vehicle infotainment systems, *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 58, 2285-2289.

Feyen, R., Liu, Y., Chaffin, D., Jimmerson, G., & Joseph, B. (2000). Computer aided ergonomics: a case study of incorporating ergonomics analysis in workplace design, Applied Ergonomics, 31, 291-300.

Hart, S. G. & Staveland, L. E. (1988) Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In P. A. Hancock and N. Meshkati (Eds.) *Human Mental Workload*. Amsterdam: North Holland Press.

Kieras, D. E. (1996). A Guide to GOMS model usability evaluation using NGOMSL. In M. Helander, T. Landauer, and P. Prabhu (Eds.), *Handbook of Human-Computer Interaction*. (Second Edition). Amsterdam: North-Holland. 733-766.

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

Liu, Y. (1997). Queueing network modeling of human performance of concurrent spatial and verbal tasks. *IEEE Transactions on Systems, Man, Cybernetics*, 27, 195-207.

Liu, Y., Feyen, R., & Tsimhoni, O. (2006). Queueing Network-Model Human Processor (QN-MHP): a computational architecture for multitask performance in human-machine systems. *ACM Transactions on Computer-Human Interaction*, 13(1), 37-70.

Pew, R. W. (2008). More than 50 years of history and accomplishments in human performance model development. Human Factors: *The Journal of the Human Factors and Ergonomics Society*, 50(3), 489-496.

Reed, M., Faraway, J., Chaffin, D. B., and Martin, B. (2006). The HUMOSIM Ergonomics Framework: A New Approach to Digital Human Simulation for Ergonomic Analysis, Proceedings of the 2006 SAE Digital Human Modeling Conference.

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

Salvucci, D. D., & Lee, F. J. (2003). Simple cognitive modeling in a complex cognitive architecture, *CHI 2003 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 265-272

Salvucci, D.D. (2006). Modeling driver behavior in a cognitive architecture. Human Factors, 48(2), 362-80.

Tsimhoni, O., & Liu. Y. (2003), Modeling steering using the Queueing Network-Model Human Processor (QN-MHP), *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 47, 1875-1879.

Wu, C., & Liu, Y. (2007a). Queueing network modeling of driver workload and performance. *IEEE Transactions on Intelligent Transportation Systems*, 8(3), 528-537.

Wu, C., & Liu, Y. (2008b). Queuing network modeling of transcription typing, *ACM Transactions on Computer Human Interaction*, 15(1), 1-45.

Wu, C., & Liu, Y. (2009). Development and evaluation of a software package for predicting human performance and workload in human-machine interface design and evaluation. *Computers and Industrial Engineering*, 56(1), 323-333.